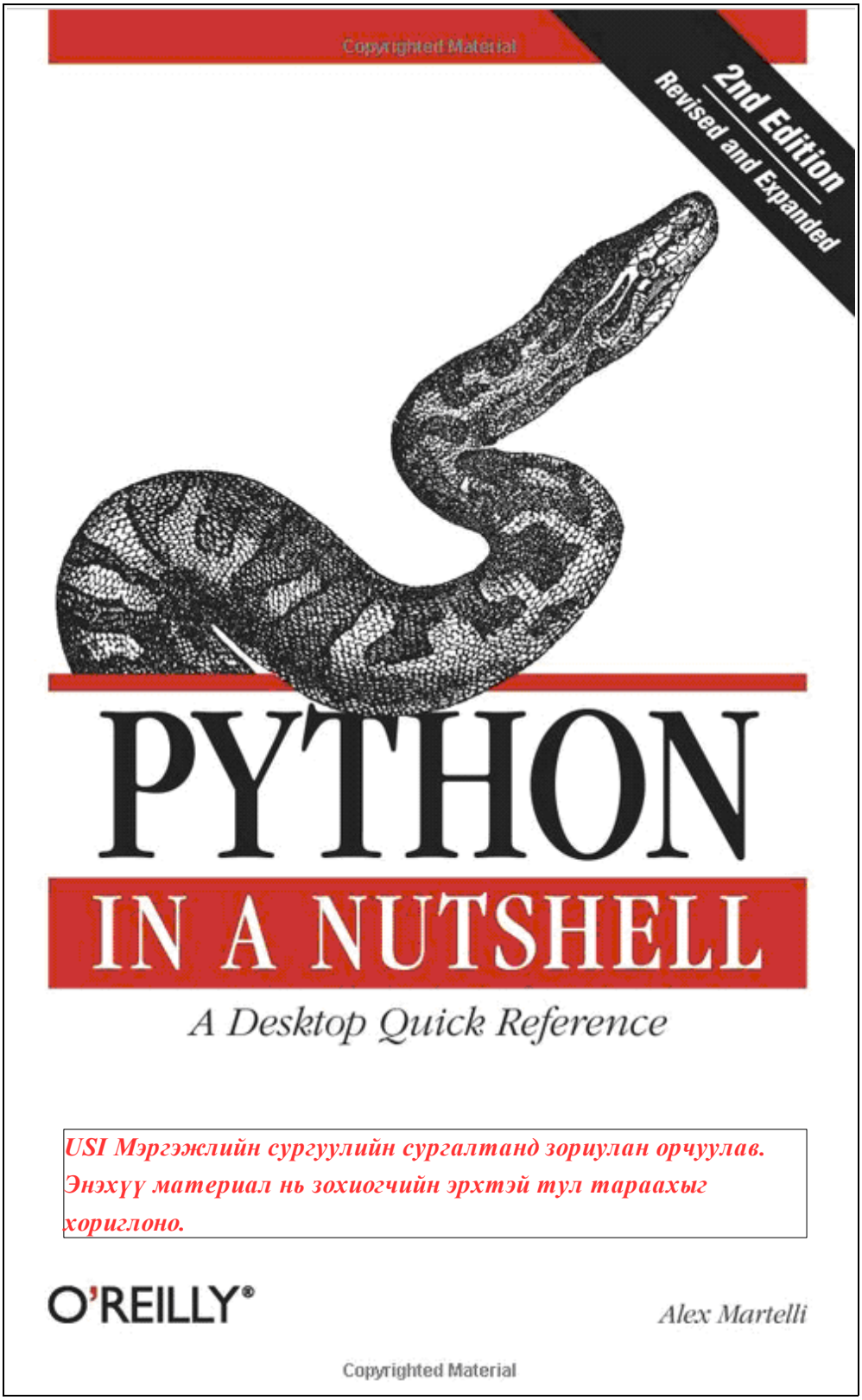

USI Мэргэжлийн сургуулийн сургалтанд зориулан орчуулав. Энэхүү материал нь зохиогчийн эрхтэй тул тараахыг хориглоно.

Python-ны онцлог

Python нь чанар, бүтээмж, зөөгдөх болон нэгтгэх боломж зэрэг онцлог зарчимтай. Эдгээр 4 шинж нь Python-ийг ашиглах гол шинж болох ба нарийвчлан тайлбарлая.

Программын чанар

Python-ийг хэрэглэн дахин ашиглагдах болон өөрчлөх боломж бүхий ойлгоход хялбар программыг бичдэг. Скрипт хэлний ертөнц дөх хөгжүүлэлтийн чанарыг дээшлүүлэх хүлээлтэнд зориулан зохиосон хэл бол Python юм. Python-ны ойлгомжтой зөв бичих зүй, уялдаа холбоотой загвар нь хэтдээ бусад хөгжүүлэгч өөрчлөх ба дахин ашиглах боломжтой үндсэн шинж бүхий ойлгомжтой кодыг бичихэд хүргэдэг.

Python хэл нь цуглуулж үүсгэсэн биш загварлаж үүсгэсэн хэлтэй адил байдаг. Энэ нь тодорхой, хамгийн бага хэмжээтэй загвартай байдаг ба энэ загвар нь программыг ойлгоход хялбар байх, урьдчилан тааварлах боломжийг олгодог. Программын тодорхой хэрэгслийг олон тооны (модуль) бүрэлдэхүүнд хувааж энгийн цөм хэлний хамт нийлмэл ярвигтай шинжийг Python хэл бүрдүүлдэг.

Python нь ‘таны тархинд зохисон’тархмал урианы баталгааны үр дүн нь үргэлж гарын авлагыг гүйлгэн уншилгүйгээр энэ хэлийг ашиглах боломж юм.

Энэ загвар нь Python өөрийн хүслээр өөрчлөх боломжтой мэргэжлийн бус хөгжүүлэгчдэд нэн тохиромжтой хэл юм. Perhaps most important is that by limiting the number of possible interactions in your code.

Python нь программын түвэгтэй байдал, алдаа үүсэх боломжийн аль алиныг нь багасгадаг.

Загвар сайтай хэлээс гадна Python нь кодыг 1 удаа бичээд олон удаа ашиглах боломжийг олгодог бүтэцлэг, модульт, обьект хандалттай зохиомж зэрэг орчин үеийн программчлалын технологиор тоноглогдсон байдаг.

Хөгжүүлэгчийн бүтээмж

Программыг хурдан хөгжүүлэхэд Python-ийг зориулсан байдаг. Илүү доод түвшиний, илүү түвэгтэй программчлалын хэлнээс илүү Python-ны интерпретатор нь таны тодорхойгоор бичсэн кодыг нарийвчлан боловсруулдаг учраас энэ хэл дээр программыг түргэн бичихэд хялбар байдаг. Төрлүүдийг зарлах, хадгалалтын схем, санах ойн зохион

байгуулалт, энгийн бодлогыг зохицуулах, компиляц хийх үйлдлийг Python-ны скриптээс олох боломжгүй.

Python-дээр бичигдсэн программ C++ , Java хэл дээр бичигдсэн бичигдсэн ижил программын 1/3-аас 1/5 –тэй тэнцэх хэмжээтэй байна. Энэ харьцаагаар хөгжүүлэгчийн хурд өснө. Python нь дээд түвшний хэл учраас Python-ны хөгжүүлэгч нь бага код бичиж, цөөн удаа зүгрүүлж, арчилгаа багатай байна.

Программын зөөгдөх чанар

Python дээр бичсэн ихэнх программ нь Windows, Linux, Macintosh систем ба IBM том машин ба Cray суперкомпьютер, гартаа барьдаг PDA зэрэг өнөөдрийн бүхий л компьютерийн систем дээр ямар 1 өөрчлөлтгүйгээр ажиллана. Python дээр бичсэн программ нь тэр ч байтугай гар утас, Apple-ийн iPod, тоглоом зэрэг чамин төхөөрөмж дээр ч ажиллана. Зарим орчин нь үл зөөгдөх нэмэлтүүдийг санал болгодог боловч Python хэлний цөм ба сангууд нь орчин болон техникээс үл хамаарах ба техникийн ялгаа гарч ирэх үед зохицуулах багажаар хангагдсан байдаг.

Бүрэлдхүүнийг нэгтгэх

Python нь хаалттай хайрцаг биш. Энэ нь өөр багажуудтай холбогдон ажиллахад зориулагдан бүтээгдсэн. Python-дээр бичсэн программуудыг хялбархан хольж нэгтгэх болон өөр системийн бүрэлдэхүүнийг ашиглах боломжтой. Энэ нь Python-ийг удирдах ба өөрийн хүслээр өөрчлөхөд нэн тохиромжтой хэл болгодог. Python-ны давхарга дээр программыг өргөтгөсөн бол бүх программын эх кодыг зөөлгүйгээр хэрэглэгч программыг тохируулан тэдгээрийг залгаж болно.

Хөгжүүлэгчдийг татдаг Python –ны өөр нэг хүрээ нь хөгжүүлэгч ба хэрэглэгчийн суралцах зам төстэй байдал, урьдаас бичсэн кодын өргөн санг ашигласнаар хөгжүүлэлтийг багасгах, хөгжүүлэлт ба зохион байгуулалтын өртөгийг бууруулахад нөлөөлдөг огт үнэгүй шинж зэрэг юм.

Python нь нээлттэй эх код шинжтэй (ямар нэг компаний мэдэлд биш хэрэглэгчид хянадаг) . Python-ийг хэрэгжүүлсэн хувилбар нь үнэгүй тараагддаг тул програмж боловсруулагч компанид барьцаалагддаггүй.Арилжааны хэрэгслүүдээс ялгаатай нь Python нь хэзээ ч хэн нэгний дур зоргоор хэрэглэгдэхээ болихгүй. Эх код руу хандах боломж нь хөгжүүлэгчдийг илүү чөлөөтэй болгох ба баримт бичгийн сүүлийн хувилбараар хангана.

Яагаад ердөө C буюу C++-ийг ашигладаггүй вэ?

Скрипт хэлийг өмнө нь огт ашиглаж үзээгүй бол энэ асуулт танд тавигддаг байсан нь дамжиггүй. C нь маш хурдан бөгөөд өргөн тархсан хэл юм. Гэвч яагаад Python

ингэж их дэлгэрсэн бэ? Python нь удаан хугацааны стратегийн үүрэгтэй, гэвч хөрвүүлэгддэг хэлнээс ялгаатай нь тактикийн хурдан горимд хэвийн ажиллана. Python-аар программыг байгуулж дахин ашиглах өөрчлөхөд хялбархан байдаг. Энэ нь web-ийн хувьд ялангуяа үнэн бөгөөд текст боловсруулалт нь гол, шинэчлэл нь тогтмол, хөгжүүлэлтийн хурдаас хамаарч төсөл тасрах, амжилттай болох үед илэрдэг . Ийм web-ийн хувьд:

- Python-ны тэмдэгт мөр ба хэв загвартай харьцуулах харьцуулалт нь тэмдэгт мөрийн хэмжээг хязгаарлах шаардлагагүй, тэмдэгт мөрийг хайх, хуваах, нэгтгэх, хэсэглэх үйлдлийг энгийн зүйл болгодог. C хэл дээр бүх зүйл төрөл ба хэмжээгээр хязгаарлагддаг тул эдгээр үйлдэл нь нуршуу байдаг.
- Python-ны дэмждэг өгөгдлийн бүтэц нь жишээлбэл нийлмэл толь бичгийн үг үсгийг шивж Python түүнийг байгуулдаг. Энд санах ойг хуваарилах, мэдээллийг байрлуулах, чөлөөлөх шаардлага байхгүй.
- Python-хэлээр программыг бичихэд их энгийн. Төрлийг зарлах шаардлагагүй тул жишээ нь программын код илүү богино болдог. Маш их өргөн хүрээнд хэрэглэгдэж дахин ашиглагдана. Кодын хэмжээ бага бол, программ нь хурдан болно. Python шиг скрипт хэлний ажиллаж байх үеийн алдааг шалгах боломж нь алдааг олох, засах үйлдлийг хялбар болгодог.
- web-тэй холбоотой үнэгүй программын ихээхэн цуглуулга Python-ны программ зохиогчдод зориулагдсан байдаг . Үүнд: стандарт модулийн клиент сервер протоколыг ашиглах, Zope, Plone, CherryPy, Django, ба Webware зэрэг web программын суурь хэрэгслүүдийг ашиглаж болно. Энэ нь үйлдвэрлэлийн түвшний web сайтыг байгуулах үйлдлийг хялбарчилдаг.

Бусад хэрэглээний хувьд, дээрхтэй ижил үзүүлэлтүүд хамаардаг. Үнэндээ, Python-ийг хэсэг хугацаанд хэрэглэсний дараа маш хэцүү учраас компиляц хийдэг хэлийг ашиглан хийх боломжгүй гэж үзэж байсан зүйлийг ч боломжтой болгоно. Сүлжээний скрипт, GUI, олон бодлого зэрэгцээ ажиллуулах зэрэг үйлдэл C дээр ярвигтай, Python дээр хялбар байдаг.

C нь ялангуяа сүлжээний ажлын хувьд нийлмэл, уян хатан биш, их удаан байдаг. Ийм динамик хэрэглээний хувьд Python шиг скрипт хэлний хурдан, уян хатан хөгжүүлэлт шаардлагатай. Компиляц хийдэг хэл нь хурдан ажилладаг хэдий ч хөгжүүлэлтийн хурд нь удаан байх нь Web –ийн ажиллах хурдад нөлөөлөх талтай. Танд анхааруулахад, нэг удаа л Python-ийг ашиглаж эхэлсэн бол та орхинао гэж байхгүй.

Jython-ийг суулгах

Jython-ийг суулгахын тулд Java 1.1 буюу түүнээс хойших хувилбараар компил хийсэн JVM хэрэгтэй. Jython-ийг http://www.jython.org-аас татаж авна. Жишээ нь :Python 2.2-тай адил хувилбар нь

http://prdownloads.sf.net/jython/jython-22.class-д байна.

To install Jython, you need a Java Virtual Machine (JVM) that complies with Java 1.1 or higher. See http://www.jython.org/platform.html for advice on JVMs for your platform.

To download Jython, visit http://www.jython.org and follow the link labeled Download. The latest version, which at the time of this writing (supporting some Python 2.3 features, as well as all of Python 2.2) is:

http://prdownloads.sf.net/jython/jython-22.class

Илэрхий байх үүднээс, C:\Jy нэртэй директорыг үүсгэж түүндээ jython-22.class-ийг татаж авсан гэж үзье. Unix төст суурийн хувьд ~/Jy-директорт байгаа гэж үзье.

Jython-ны суулгах класс нь өөрийгөө суулгадаг программ юм. Командын терминалыг нээж, дээрх директорт шилжиж Jython-ны суулгагчийг Java интерпретатор дээр ажиллуулна. Jython-ны суулгагч байгаа директорыг CLASSPATH –д бичиж өгнө. Дараах командаар ажиллуулна.

C:\Jy> **java -cp**
. jython-22
IronPython-ийг суулгах

IronPython-ийг суулгахын тулд Ажиллах үеийн Энгийн Хэл (Common Language Runtime-CLR)-ний сүүлийн хувилбарыг суулгасан байх шаардлагатай. Mono-гийн сүүлийн хувилбар (http://www.mono-project.com/Main_Page-ийг үзнэ үү)ба Microsoft .NET Framework 2.0 нь IronPython-тай сайн зохицон ажилладаг. IronPython-ийг http://www.codeplex.com/Wiki/View.aspx?ProjectName=IronPython хуудаснаас татан авч зааврын дагуу суулгана. IronPython-нь Common Language Runtime-ийг ашигласан программыг Python-дээр бичих боломжийг олгоно.

Python хэлний үндэс

Үгийн сангийн бүтэц

Программчлалын хэлний үгийн сангийн бүтэц нь тухайн хэлнээ хэрхэн бичихийг зохицуулсан үндсэн дүрмүүдийг олонлог юм. Энэ нь доод түвшний өгүүлбэр зүй ба хувьсагчийн нэрс ямар байх вэ? Тайлбарын эхлэл болгон ямар үсгийг хэрэглэх вэ зэргийг тодорхойлно. Python –ны эх код нь бусад текст файлын адил тэмдэгтийн дараалал юм. You can also usefully consider it as a sequence of lines, tokens, or statements. These different lexical views complement and reinforce each other. Python is very particular about program layout, especially with regard to lines and indentation, so you'll want to pay attention to this information if you are coming to Python from another language.

Мөр ба догол мөр

1 буюу олон физик мөрөөс тогтсон логик мөрийн дарааллаас Python-дээр бичсэн программ тогтдог. Физик мөр бүр тайлбараар төгсөж болно. Тэмдэгт мөрийн дотор бичигдээгүй чагт тэмдэгт(#)-ээр тайлбар эхэлнэ. Ийм чагтийн(#)- ард физик мөрийн төгсгөл хүртэлх бүх тэмдэгт нь тайлбарт тооцогдож Python-ний интерпретатор хэрэгсэхгүй орхино. Мөн тайлбартай тайлбаргүй бүх хоосон мөрийг хэрэгсэхгүй. In an interactive interpreter session, you must enter an empty physical line (without any whitespace or comment) to terminate a multiline statement.

Python-дээр физик мөрийн төгсгөл нь ихэнх командын төгсгөл гэж тооцогддог. Бусад хэлтэй адилгүй нь Python-ны мөрийг цэг таслал(;) зэрэг хязгаарлагчаар төгсгөх шаардлагагүй. Команд нь 1 мөрөнд багтахааргүй хэт урт байвал дараалсан 2 мөрийг 1 логик мөр болгохын тулд эхний мөр нь тайлбаргүй байх хэрэгтэй бөгөөд урагшаа налуу зураасаар(\) төгсгөнө. Гэвч (,[,{ зэрэг хаалтуудыг нээсний дараа хаагаагүй бол Python эдгээр зэргэлдээх физик мөрүүдийг 1 логик мөр болгодог.Энэ механизмын давуу тал нь олон физик мөрийн ард урагшаа налуу зураасаар(\) бичилгүй ойлгоход хялбар кодыг бичих юм. 3 ширхэг хашилт (“) нь физик мөрийг сунгадаг.

Python нь догол мөрийг програмын хэсэглэсэн бүтцийг илэрхийлэхэд хэрэглэнэ. Бусад хэлтэй адилгүй нь {,} шиг хэсэглэсэн бүтэц эхлэх, төгсөх хязгаарлагчийг хэрэглэдэггүй, зөвхөн догол мөр л хэсэглэсэн бүтцийг бүрдүүлэн илэрхийлэх арга болдог. Python программын логик мөр бүр нь эхэндээ догол мөртэй байдаг. Хэсэг нь дарааласан логик мөрүүдээс тогтох ба эдгээр мөр нь адил түвшингийн догол мөрөөр эхэлсэнн байх ба өмнөх түвшингийн догол мөр бүхий логик мөр нь тухайн хэсэг дууссаныг илэрхийлнэ.

Үсгийн олонлог

Ер нь Python эх код бүхий файл нь ASCII тэмдэгтээс(0-127) бүтнэ. ASCII-д багтаагүй өөр тэмдэгтүүд багтаах хэрэгтэй бол кодчилолыг илэрхийлсэн тайлбар мөрийг оруулж өгнө.

```
# -*- coding: utf-8 -*-
```

coding: ба the -*- ийн хооронд utf-8 or iso-8859-1 зэрэг Python-ны кодекийн нэрийг бичиж өгнө.

Хэлний нэгж хэсгүүд

Python нь логик мөрийг тэмдэгт хэсэг гэж нэрлэгдэх үгийн сангийн энгийн хэсгийн дараалал болгон хуваадаг. Тэмдэгт хэсгийн төрлүүд нь нэр, түлхүүр үг, оператор, хязгаарлагч, ба бусад үг юм.

Нэрс

Нэрийг хувьсагч, функц, класс, модуль болон бусад обьектийг нэрлэхэд хэрэглэнэ. Нэр нь үсэг(A-Z, a-z) буюу доогуур зураас (_)-аар эхлэх ба түүний хойноос 0 буюу түүнээс дээш үсэг, доогуур зураас, цифр байж болно.

Түлхүүр үг

Python нь 30 түлхүүр үгтэй. Түлхүүр үг гэдэг нь өгүүлбэр зүйд тусгайлан ашиглахаар нөөцөлсөн нэрс юм. Түлхүүр үгс нь бүгд жижиг үсгээр бичигдэнэ. Python нь дараах түлхүүр үгтэй:

and

assert

break

class

continue

def

del

elif

else

except

exec

finally

for

from

global

if

import

in

is

lambda

not

or

pass

print

raise

return

try

while

with (2.5)

yield

Оператор

Python нь тоо ба үсгэн ба үсгийн хослолыг оператор болгон ашиглана. Python нь дараах операторыг танина.

+

-

*

/

%

**

//

<<

>>

&

|

^

~

<

<=

>

>=

<>

!=

==

Хязгаарлагч

Python нь дараах тэмдэг ба тэмдэгийн хослолыг хязгаарлагч болгон хэрэглэдэг. Илэрхийлэл, жагсаалт, толь бичиг, команд ба тэмдэгт мөрийн төрөл бүрийн хэсэгт хязгаарлагчийг хэрэглэнэ.

(

)

[

]

{

}

,

:

.

'

=

;

+=

-=

*=

/=

//=

%=

&=

|=

^=

>>=

<<=

**=

Цэг (.) нь бас бутархай тоонд орж бичигдэнэ. Дараах тэмдэгтүүд бусад нэгжийн хэсэг болон тусгай утгыг илэрхийлнэ.

'

"

#

\

$ ба ? , сул зайнаас бусад удирдах тэмдэгтүүд болон 126-аас хойш кодтой бүх тэмдэгт нь Python программын тайлбар ба тэмдэгт мөрийн бичлэгээс бусад хэсэгт хэзээ ч орж болохгүй.

Үг(literal)

Программд бичигдэж болох тоо буюу тэмдэгт мөрийг үг гэнэ.Дараах нь Python-ны үгс юм.

```
          # Integer literal
3.14            # Floating-point literal
1.0j            # Imaginary literal
```

```
'hello'                 # String literal
"world"                   # Another string literal
"""Good night"""                 # Triple-quoted string literal
```

Үг ба хязгаарлагчийг ашиглан зарим үндсэн төрлүүдийн өгөгдлийг үүсгэж болно.

```
 [ 42, 3.14, 'hello' ]    # List
( 100, 200, 300 )         # Tuple
{ 'x':42, 'y':3.14 }      # Dictionary
```

Команд

Python-ны эх бүхий файлыг энгийн буюу нийлмэл командын дараалал гэж үзэж болно. Бусад хэлнээс ялгаатай нь хувьсагч зарлах зэрэг дээд хэсэгт бичих элементүүд Python-д байхгүй зөвхөн командууд л байдаг.

Энгийн команд

Энгийн команд нь өөр командыг агуулдаггүй. Энгийн команд нь бүтнээрээ 1 логик мөрөнд байдаг. Бусад хэлний адидаар 1-ээс олон командыг цэг таслалаар тусгаарлан 1 логик мөрөнд байрлуулж болно. Гэвч 1 мөрөнд 1 команд байх нь Python –ны энгийн хэв маяг бөгөөд энэ нь илүү ойлгомжтой болгодог.

Python нь обьект хандлагат хэл

Python нь обьект хандалттай программын хэл юм. Бусад обьект хандалттай хэлнээс ялгаатай нь Python нь зөвхөн обьект хандалттай загварыг хэрэглэхийг тулгадаггүй. Python нь бас модуль функц бүхий процедурын программчлалыг дэмждэг ба өөрийн программын хэсэг бүрт нийцсэн загварыг сонгож болно. Ер нь программын өгөгдлийн ба кодын хэсгийг хамт нь ажил үүргийн зорилгоор багцалж хэрэглэхэд обьект хандалтад программ тохиромжтой байдаг. Обьект хандалтад программын давуу тал онц шаардлагагүй хэсэгт процедурын программчлалын загварыг ашиглах боломжтой.

5.1. Класс ба классын жишээ

Өмнө нь обьект хандалттай программчлалын тухай мэдлэгтэй бол класс нь энэ хэрэглэгчийн тодорхойлдог төрөл бөгөөд энэ төрлийн обьектын жишээ нь маш чухал гэдгийг ойлгоно.

Python-ны класс

Класс нь дараах шинж бүхий Python-ны обьект юм.

-

Энэ нь функц байсан бол түүнийг дуудаж болно. Энэ дуудагдсан функц нь өөр обьектод утга буцаах ба үүнийг классын жишээ; тухайн классыг жишээний төрөл гэж нэрлэдэг.

- Классыг дураар нэрлэх боломжтой ба түүнийг зааж, холбож болно.
- Классын шинж чанарууд(функцыг оруулаад) нь тодорхойлогч байж болно.

  Классын шинж чанарууд классын метод гэж ойлгогддог функцтэй холбогдоно.
- Метод нь Python-ны тодорхойлсон тусгай нэртэй (урд хойноо 2 ширхэг доогуур зураастай) байж болно. Класс нь эдгээрийг дэмждэг, классын жишээнд янз бүрийн үйлдлийг багтаах гэж байвал Python нь ийм тусгай методыг дууддаг.
- Класс нь өөр классаас удамшиж болно. Энэ нь тухайн классд олдохгүй шинж чанаруудыг төлөөлүүлсэн өөр классаас хайх боломжтой гэсэн үг.

. *Тодорхойлогч(Descriptor)*

Тодорхойлогч нь _ _get_ _ гэсэн нэртэй тусгай метод бүхий шинэ маягийн обьект юм..

Өгөгдлийн төрлүүд

Python дээрх бүх өгөгдөл нь обьект бөгөөд обьект бүр нь утга ба төрөлтэй. Обьектийн төрөл нь обьектийн ямар үйлдлийг дэмжихийг, өөрөөр хэлбэл түүний өгөгдлийн утга нь ямар үйлдэлд оролцож болохыг тодорхойлно. Төрөл нь бас тухайн обьектийг өөрчилсөн ч элементүүд ба шинж чанарыг тодорхойлно. Өөрчлөгддөг обьектийг хувирамтгай гэдэг бол үл өөрчлөгддөгийг нь хувиршгүй гэдэг.

type(*obj*) функц нь дурын обьектыг параметр болгон хэрэглэж тухайн обьектийн төрлийг буцаадаг. *obj* обьект буюу түүнээс үүссэн дэд класс нь *type* төрөлтэй бол isinstance(*obj*, *type*)функц нь үнэн(true) утгыг эсрэг тохиолдолд худал (False) утгыг буцаадаг.

Python нь тоо, тэмдэгт мөр,tuple, жагсаалт, хэш зэрэг төрлүүдтэй.

Тоо

Python-ны тоон обьект нь бүхэл, бутархай, комплех тоог дэмждэг. Python дээрх бүх тоон обьект нь хувиршгүй юм. Энэ нь тоон обьект дээр ямар 1 үйлдэл хийхэд дандаа шинэ обьект үүсгэнэ гэсэн үг.

Тооны бичиглэлийн урдах + - тэмдгийг тоонд нэгтгэж ойлгохгүй тусад нь оператор болгон авдаг.

Бүхэл тоо

Бүхэл тоон бичиглэл нь аравт, наймт, 16-тынх байна. Аравтын бичиглэл нь эхний цифр нь 0 биш байх цифрүүдээс тогтоно. Харин 0-ээр эхэлбэл наймтын тоо гэж ойлгоно. 16-тын тоог илэрхийлэхдээ 0x-ээр эхэлсэн 16-тын цифрүүдийг бичнэ.Жишээ нь:

```
1, 23, 3493              # Аравт



01, 027, 06645           # Наймт
0x1, 0x17, 0xDA5         # 16-т
```

Энгийн болон урт бүхэл тоог ялгах шаардлага орчин цагийн Python-д байхгүй хэдий ч урт бүхэл тоо гэж тодорхойлохын тулд төгсгөлд нь L (буюу l) –ийг бичиж өгнө.Жишээ нь:

```
1L, 23L, 99999333493L       # Long decimal integers
01L, 027L, 01351033136165L  # Long octal integers
0x1L, 0x17L, 0x17486CBC75L  # Long hexadecimal integers
```

Бутархай тоо

Бутархай тоон бичиглэл нь аравтын таслал(.), зэргийн E буюу e,эерэг, сөргийг илэрхийлэх + , - зэргийг агуулсан аравтын цифрүүд байна. Эхний цифр нь E буюу e байж болохгүй аравтын таслал(.) ба аравтын цифрийн аль 1 нь байж болно.Жишээлбэл:

```
0., 0.0, .0, 1., 1.0, 1e0, 1.e0, 1.0e0
```

Python –ны бутархай тоо нь C-гийн double төрөлтэй адил орчин үеийн систем дээр 53 бит тоог илэрхийлэх боломжтой.

Комплекс тоо

Комплекс тоо нв бодит ба хийсвэр хэсэг гэсэн 2 бутархай тооноос бүтнэ. Бодит ба хийсвэр утга гэсэн зөвхөн уншигдах шинжтэй z.real ба z.imag гэж харьцана.

Хийсвэр тоо гэж ялгахдаа бутархай тооны ард j буюу J-г бичнэ.

```
0j, 0.j, 0.0j, .0j, 1j, 1.j, 1.0j, 1e0j, 1.e0j, 1.0e0j
```

Дараалал

Дараалал нь эерэг тоогоор индекслэгдсэн дараалсан итемүүдийг багтаадаг. Python нь тэмдэгт мөр, жагсаалт шиг дараалал гэсэн төрөлтэй.

Давтагдах төрөл(Iterables)

Бүх дараалал нь давтагдах төрөл бөгөөд жагсаалт зэрэг дарааллыг хэрэглэхдээ давтагдах төрлийг ашиглаж болно. Ер нь давтагдах төрлийг ашиглах гэдэгт эцэст нь элементүүдийг уншихаа зогсдог, хязгаартай давтагдах төрлийг хэлж байгаа юм. Бүх дараалал нь хязгаартай. Дараалал нь хязгааргүй байх боломжтой ч түүний хязгааргүй давтагдах төрөл нь хэзээ ч үл дуусах юм уу санах ойг дүүргэх программыг бичихэд хүргэдэг.

Тэмдэгт мөр

Python-ны тэмдэгт мөрийн обьект нь текст төрлийн мэдээллийг илэрхийлэх, хадгалахад хэрэглэгддэг тэмдэгтийн дараалал юм. Python дээрх тэмдэгт мөр нь хувирдаггүй тул тэмдэгт мөрөнд ямар нэг үйлдэл хийхдээ одоо байгаа тэмдэгт мөрийг өөрчлөлгүй үргэлж шинэ обьектийг үүсгэдэг.

Тэмдэгт мөр нь хашилттай буюу гурвалсан хашилттай байж болно. Хашилтан дах тэмдэгт мөр нь дан (') ба давхар (") хашилтаар хашигдсан 0 буюу түүнээс олон тэмдэгтийн дараалал юм.

```
'This is a literal string'
"This is another string"
```

Дан ба давхар хашилтын үүрэг нь адил. Хоёр төрлийн хашилт бүхий тэмдэгт мөрийн аль алинд нь нөгөө төрлийн хашилтыг урагшаа налуу зураас(\) ашиглалгүйгээр хэрэглэж болно.

```
'I\'m a Python fanatic'          # a quote can be escaped
"I'm a Python fanatic"           # this way is more readable
```

Дан хашилттай тэмдэгт мөр нь Python-ийн илүү ерөнхий хэлбэр гэж тооцогддог. Тэмдэгт мөрийг олон физик мөрөнд үргэлжлүүлэн бичихийн тулд урагшаа налуу зураасыг мөрийн сүүлийн тэмдэгт болгон бичих ба энэ нь дараагийн мөр өмнөх мөрийн үргэлжлэл болохыг илэрхийлнэ.

```
"A not very long string\
that spans two lines"            # comment not allowed on previous line
```

Тэмдэгт мөрийг 2 мөртэй гаргахын тулд түүний дунд шинэ мөр тэмдэгтийг оруулж өгнө.

```
"A not very long string\n\
that prints on two lines"        # comment not allowed on previous line
```

A better approach is to use a triple-quoted string, which is enclosed by matching triplets of quote characters (''' or """):

Тэмдэгт мөрийг бичих сайн арга нь гурвалсан хашилттай тэмдэгт мөрийг хэрэглэх юм. Гурвалсан хашилттай тэмдэгт мөр нь гурвалсан хашилт(''' буюу """) –аар хашигдсан тэмдэгтүүд юм.

```
"""An even bigger string that spans three lines"""                # comments not allowed on previous lines
```

Гурвалсан хашилттай тэмдэгт мөрийн мөр таслагч нь шинэ мөр тэмдэгт шиг нөөцлөгдсөн байна.

In a triple-quoted string literal, line breaks in the literal are preserved as newline characters in the resulting string object.

Урдаа урагшаа налуу зураасгүй дан урагшаа налуу зураасыг гурвалсан хашилттай тэмдэгт мөрөнд бичиж болохгүй.Мөн энэ тэмдэг мөрийг хааж болох мөрийн төгсгөл, хашилтыг агуулж болохгүй. Хүснэгт 4-1 нь урдаа урагшаа налуу зураастай тэмдэгтүүдийг дараалллыг үзүүлж байна. (Алгасдаг тэмдэгтийн дараалал)

Хүснэгт 4-1. Алгасдаг тэмдэгтийн дараалал

Дараалал	**Утга**
\<newline>	Мөрийн төгсгөлийг үл хэрэгсэх
\\	Урагшаа налуу зураас
\'	Дан хашилт
\"	Давхар хашилт
\a	Хонх
\b	Урагшаа арилгах(Back space)
\f	Маягтыг гүйлгэх(Form feed)
\n	Шинэ мөр
\r	Буцаан шилжүүлэх (Carriage return)
\t	Табуляц(Tab)

Хүснэгт 4-1. Алгасдаг тэмдэгтийн дараалал

Дараалал	**Утга**
\v	Хөндлөн табуляц(Vertical tab)
DDD	*DDD* 8-тын утга
\x*XX*	*XX* 16-тын утга
other	бусад тэмдэгт

Тэмдэгтийн бичиглэлийн 1 хэлбэр нь түүхий тэмдэгт мөр юм. Эхний хашилтын өмнө r буюу R тэмдэгтийг бичих ба бусад бичиглэл нь адилхан байна. Түүхий тэмдэгт мөрийн хувьд урдаа урагшаа налуу зураастай тэмдэгтүүдийг Хүснэгт 4-1 шиг хувиргахгүй урагшаа налуу зураас ба шинэ мөр тэмдэгтүүд ба бусад бүх тэмдэгтийг тэр чигт нь хуулна. Түүхий тэмдэгт мөрийн бичиглэл нь хэвийн илэрхийлэл шиг олон урагшаа налуу зураастай тул хэрэглэхэд эвтэйхэн байдаг. Түүхий тэмдэгт мөр нь сондгой тоотой урагшаа налуу зураасаар төгсөж болохгүй.Учир нь сүүлийнх нь арын хашилтыг тэмдэгт мөрийн доторх хашилт гэж уншихад хүргэнэ.

Unicode тэмдэгт мөрийн эхний хашилтын өмнө r буюу R тэмдэгтийг бичих ба энэ тэмдэгт мөрийн дотор 4 оронтой 16-тын тоо буюу 1 Unicode тэмдэгтийг бичиж болно.

For example, \N{Copyright Sign} indicates a Unicode copyright sign character (©). Raw Unicode string literals start with ur, not ru. Note that raw strings are not a different type from ordinary strings: raw strings are just an alternative syntax for literals of the usual two string types, plain (a.k.a. byte strings) and Unicode.

Multiple string literals of any kind (quoted, triple-quoted, raw, Unicode) can be adjacent, with optional whitespace in between. The compiler concatenates such adjacent string literals into a single string object. If any literal in the concatenation is Unicode, the whole result is Unicode. Writing a long string literal in this way lets you present it readably across multiple physical lines and gives you an opportunity to insert comments about parts of the string. For example:

```
marypop = ('supercalifragilistic'  # Open paren -> logical line continues
        'expialidocious')       # Indentation ignored in continuation
```

The string assigned to marypop is a single word of 34 characters.

. Tuples

Tuple нь хувирахгүй дэс дараатай элементийн дараалал юм:

tuple-ийн элементүүд нь өөр өөр төрөлтэй байж болох хүслээрээ өөрчлөх боломжтой обьект юм. Tuple-ийг тодорхойлохдоо tuple-ийн элементүүд болох илэрхийллийн цувааг ашиглана. Сүүлийн элементийн ард илүүдэл таслалыг сайн дураар нэмж болно. tuple-ийн элементүүдийг (ба) хаалтаар хашиж бүлэглэж болно. Гэхдээ таслал нь өөр утгатай байх буюу хоосон буюу давхар tuple-ийг тодорхойлох үед хаалт хэрэг болдог. Яг 2 элементтэй tuple-ийг хос гэж нэрлэнэ. 2 элементтэй tuple- үүсгэхийн тулд илэрхийллийн төгсгөлд таслал нэмнэ. Хоосон tuple-ийг илэрхийлэхдээ хоосон хаалтыг хэрэглэнэ. Заавал биш хаалтаар хүрээлэгдсэн зарим tuple-ийг бичье.

```
(100, 200, 300)          # Tuple with three items
(3.14,)                  # Tuple with one item
( )                       # Empty tuple (parentheses NOT optional!)
```

tuple үүсгэхдээ Python-ны tuple –ийг хэрэглэж болно.Жишээ нь:

```
tuple('wow')
```

Энэ tuple нь дараахтай адил.

```
('w', 'o', 'w')
```

tuple( )гэж tuple -ийг параметргүй дуудвал хоосон tuple үүсгэж буцаана. *x* нь давтах обьект бол tuple(*x*) нь *x*–ийн элементтэй адил элементтэй tuple -ийг буцаана.

Жагсаалт

Жагсаалт нь хувирах дэс дараалалтай элементүүдийн дараалал юм. Жагсаалтын элементүүд нь өөр өөр төрөлтэй байж болох хүслээрээ өөрчлөх боломжтой обьект юм . Жагсаалтыг тодорхойлохдоо жагсаалтын элементүүд болох таслалаар тусгаарлагдаж, [ба] хаалтаар хашигдсан илэрхийллийн цувааг ашиглана. Хоосон жагсаалтыг хоосон хаалтаар([ ]) илэрхийлнэ. Сүүлийн элементийн ард илүүдэл таслалыг сайн дураар нэмж болно. Жагсаалтын жишээг бичье.

```
[42, 3.14, 'hello']      # List with three items
[100]                    # List with one item
[]                       # Empty list
```

list-ийг дуудаж жагсаалт үүсгэж болно.

```
list('wow')
```

Энэ нь дараах жагсаалттай адил:

```
['w', 'o', 'w']
```

list( ) нь параметргүйгээр хоосон жагсаалтыг үүсгэнэ. *x* нь давтах обьект бол list(*x*) нь *x*–ийн элементүүдтэй адил элементүүдтэй жагсаалтыг буцаана.

Олонлог

Python нь үл давхардах элементүүдтэй дурын дэс дараалалтай цуглуулгыг илэрхийлэх set and frozenset төрөлтэй. Энэ төрлүүд нь Python 2.3-ийн sets модулийн Set ба ImmutableSet классуудтай ижил. Дараах код нь Python-ны сүүлийн хувилбар ба Python 2.3-ийн аль алин дээр олонлогийг импортлох боломжийг олгоно.

```
try:
  set except NameError:
  from sets import Set as set, ImmutableSet as frozenset
```

Олонлогийн элементүүд нь өөр өөр төрөлтэй байж болох ба гэхдээ эдгээр нь хэш шиг байх ёстой. set төрлийн жишээ нь хувирдаг бөгөөд хэш шиг биш байдаг. frozenset төрлийн жишээ нь хувирдаггүй бөгөөд хэш шиг байдаг. Олонлог нь олонлог (set) төрөлтэй элементтэй байж болохгүй боловч frozenset төрлийн элемэнттэй байж болно. Олонлог ба frozenset нь дэс дараалалгүй байдаг.

Олонлогийг үүсгэхдээ set –ийг параметргүй эсвэл 1 давтагч обьект бүхий параметртэй дуудна. (энэ олонлогийн элементүүд нь тухайн давтагч обьектын элементүүд байна гэсэн үг.)

Толь бичиг

Энэ харгалзаа нь түлхүүр гэж нэрлэгддэг бараг л дурын байх утгаар индекслэгдсэн обьектүүдийн дурын цуглуулга юм. Энэ харгалзааны нь дараалласс ялгаатай тал нь хувиран өөрчлөгддөг ба энэ нь дэс дараагүй байдаг.

Python нь харгалзааны цор ганц төрөл болох толь бичгийн төрөлтэй. Сан ба нэмэлт модулиуд нь өөрөө харгалзааны төрлийг үүсгэх боломжийг бүрдүүлнэ. Толь бичгийн түлхүүрүүд нь өөр өөр төрөлтэй байж болох боловч эдгээр нь хэш шиг байх ёстой. Толь

бичгийн утгууд нь өөр өөр төрөлтэй байх дурын обьектүүд байна. Толь бичгийн элемент бүр нь түлхүүр ба утгын хос юм. Толь бичгийг нэгдмэл массив гэж ойлгож болно. Толь бичгийг тодорхойлохдоо, {} хаалтан дах таслалаар тусгаарлагдсан илэрхийллийн хосын цувааг ашиглана. Хосууд нь толь бичгийн элементүүд юм. Сүүлийн элементүүдийн ард нэмэлт таслал байж болно. *key* нь элементийн түлхүүрт олгогддог илэрхийлэл, *value* нь элементийн утгад олгогддог илэрхийлэл байх *key*:*value* хэлбэрээр толь бичгийн элемент бүр нь бичигдэнэ. Толь бичгийн бичиглэлд түлхүүрийн утга 1-ээс олон бичигдсэн бол энэ түлхүүр бүхий элементүүдийн зөвхөн 1 л толь бичигт хадгалагдах ба түлхүүр нь давхцах ёсгүй байдаг. Хоосон толь бичгийг {} хос хаалтаар илэрхийлнэ. Энд зарим толь бичгүүдийг тодорхойльё.

```
{'x':42, 'y':3.14, 'z':7 }   # Dictionary with three items and string keys
{1:2, 3:4 }                  # Dictionary with two items and integer keys
{}                           # Empty dictionary
```

You can also call the built-in type dict to create a dictionary in a way that, while less concise, can sometimes be more readable. For example, the dictionaries in this last snippet can also, equivalently, be written as, respectively:

Толь бичгийг үүсгэх өөр 1 арга нь dict функцийг дуудах бөгөөд энэ нь хураангуй биш ч заримдаа илүү ойлгомжтой байдаг. Жишээ нь : өмнөх жишээнд өгүүлсэн толь бичгийг, харгалзан дараах тэнцүү бичлэгээр илэрхийлж болно.

```
dict(x=42, y=3.14, z=7)      # Dictionary with three items and string keys dict([[1, 2], [3, 4]])      #
Dictionary with two items and integer keys dict( )                          # Empty dictionary
```

dict( ) –ийг параметргүйгээр дуудвал хоосон толь бичгийг буцаана. dict –д дамжуулж байгаа параметр нь харгалзаа бол dict нь *x* –тэй ижил түлхүүр ба ижил утгатай шинэ толь бичгийг буцаана. *x* нь давтагч обьект бол *x* –ийн элементүүд нь хос байх ёстой ба *x* –ийн хос утгыг түлхүүр ба утгад олгосон элементүүд бүхий толь бичгийг буцаана. *x* –д түлхүүр утга нь 1 –ээс олон удаа байвал, зөвхөн сүүлийн нь утга л толь бичгийн элементийн утга болон хадгалагдана.

x гэсэн байрлалт параметрийн оронд буюу нэмж нэрлэсэн параметртэйгээр dict -ийг дуудаж болно. *name* нь элементийн түлхүүрийн нэр, *value* нь элементийн утгад олгогдох илэрхийлэл болж уншигдах *name=value* бичиглэлээр параметр бүр нь бичигдэнэ. dict –ийг байрлалт ба нэрлэсэн параметрийн аль алинтай нь дуудах буюу түлхүүр нь байрлалт ба нэрлэсэн параметрийн аль алинаар нь бичигдсэн байвал Python нь тухайн түлхүүрийг утгатай нь нэрлэсэн параметраар уялдуулна.

dict.fromkeys-ийг дуудаж толь бичгийг үүсгэж болно. Эхний параметр нь толь бичгийн түлхүүр болох давтагч обьект, 2-р параметр нь түлхүүр бүрт харгалзах утга байна.

(Түлхүүрүүд нь анхнаасаа ижил тооны харгалзах утгуудтай байна.) 2-р параметр нь байхгүй бол эдгээр түлхүүр бүрт харгалзах утга нь None байна. Жишээ нь :

```
dict.fromkeys('hello', 2)   # same as {'h':2, 'e':2, 'l':2, 'o':2}
dict.fromkeys([1, 2, 3])    # same as {1:None, 2:None, 3:None}
```

None

None нь null обьектийг илэрхийлнэ. None –обьект нь нэг ч шинж чанар, методгүй байна. Хаашаа ч хамаагүй зааж байх заагч хэрэгтэй эсвэл энд хоосон обьект байгаа газрыг гэж илэрхийлэхэд None -ийг ашигладаг.Өөр утгыг буцаах тодорхой return команд байхгүй бол функц нь None-ийг буцаана.

Дуудагдах төрөл

Python-д функцийг дуудах үйлдлийг дэмжих жишээ болгон дуудагдах төрлийг ашигладаг. Функцүүд нь дуудагдах төрлийнх юм. Python нь өөрийн хэдэн функцүүдтэй ба функцийг хэрэглэгч тодорхойлох боломжтой. Үүсгэгч(Generator) нь бас дуудагдах төрөлтэй. Төрлүүд бас дуудагдах шинжтэй. Хэрэглэгчийн тодорхойлсон класс обьект нь мөн дуудагдах шинжтэй. Төрлийг дуудсанаар тухайн төрөлтэй обьектийг (шинэ жишээг) үүсгэнэ.

Бусад дуудагдах төрөл ба методууд нь _ _call_ _.нэртэй тусгай метод бүхий классын жишээ буюу шинж чанартай холбогддог.

Логик утга(Boolean)

Python дээрх бүх утга үнэн зөв байдлыг илэрхийлсэн үнэн худал утгыг авч болно. Дурын 0 биш утгатай тоо, хоосон биш обьект нь (тэмдэгт мөр, жагсаалт, олонлог, толь бичиг гэх мэт) нь үнэн утгатай байна. бүх төрлийн 0 , None,ба хоосон обьект нь худал гэсэн утгатай байна. Бутархай тооны хувьд үнэн зөв байдлын утгыг болгоомжтой хэрэглээрэй. Энэ үед 0-той яг тэнцүү утгатай харьцуулах ба бутархай тоо нь нарийвчлал ихтэй тул бараг яг тэнцэнэ гэж байдаггүй.

Python-ны bool нь int –ийн дэд класс юм. bool төрөл нь True and False гэсэн 2 утгатай ба тэмдэгтээр бол 'true' ба 'False', тоон хэлбэрээр бол харгалзан 1 гэсэн 0 утгатай. Python-ны харьцуулалт хийдэг зарим функц нь bool үр дүнг буцаадаг. bool()-ыг дурын *x* параметртэйгээр дуудаж болно. *x* нь үнэн бол энэ функцийн үр дүн үнэн, худал бол мөн худал байна. Энэ нь илүү байвал ийм дуудлагыг Python сонгодог хэлбэрт ашигладаггүй.Хэзээ ч if bool(x): if x==True: if bool(x)==True гэж бичдэггүй.

Модуль обьект

Модуль нь Python хэлний дурaарaа нэрлэх боломжтой, өөр обьектоос заах, холбох боломжтой обьект юм. *aname* нэртэй модуль нь ерөнхийдөө aname.py нэртэй файлд байрлана.

Python дээр модуль нь бусад обьекттой адил боловсруулагддаг обьект юм. Иймээс функцийг дуудах үед модулыг параметрт нь оруулж болно. Үүнтэй төстэйгээр функц нь модулийг буцааж болно. Бусад обьекттой адилаар модулийг хувьсагч, обьектийн шинж чанар, агууламжийн элемент рүү холбож болно.

import команд

Python-ны 1 программын дотроос import командыг ажиллуулан Python-ны эх код бүхий файлыг ашиглаж болно. Import-ийн өгүүлбэр зүй нь дараах хэлбэртэй байна.

import *modname* [as *varname*][,...]
import түлхүүр үгийн араас 1 буюу түүнээс олон модулийн нэрийг таслалаар тусгаарлан бичнэ.

Жишээ нь:

```
import MyModule
```

Модулын их бие

Модулийн их бие нь эх файл дах командын дараалал юм. Эх файл нь модуль гэдгийг илэрхийлэх ямар нэгэн бичлэг байхгүй. Бүх зөв Python эх файл нь модуль хэлбэрээр ашиглагдаж болно. Программ эхний удаа модулийг импортолж авах үед энэ модулийн их бие ажиллана.

Модулийн баримтын тэмдэгт мөр

Модулийн эх биеийн эхний команд нь үгчлэн зөв бичих мөр байвал компилятор нь _ _doc_ _ нэртэй модулийн баримтын тэмдэгт мөр гэсэн тэмдэгт мөртэй холбодог. Үүнийг docstrings буюу баримтын тэмдэгт мөр гэж нэрлэдэг.

Модуль ба хувийн хувьсагч

Модулийн 1 ч хувьсагч үнэхээр дотоод байж чадахгүй. Зөвшлийн _secret дагуу шиг 1 доогуур зураасаар эхэлсэн нэр бүхий хувьсагчийг хувийн гэж үздэг

from команд

Python-ны from команд нь одоогийн нэрийн олонлог руу аль 1 модулийн тодорхой шинж чанаруудыг импортлон авах боломж олгодог. Үүний өгүүлбэр зүй нь 2 хэлбэртэй.

from *modname* import *attrname* [as
varname][,...]
from *modname* import *

from командад модулийн нэрийг тодорхойлох ба түүний араас 1 буюу түүнээс олон шинж чанаруудыг таслалаар тусгаарлан бичнэ.Ихэнх энгийн тохиолдолд *attrname* нэр бүхий шинж чанарыг *modname* нэртэй модулиас авч ижил нэртэй хувьсагчтай холбоэ өгнө гэсэн үг.Жишээлбэл:

from MyModule import f

modname нв таслалаар тусгаарлагдсан нэрийн дараалал байж болох ба энэ нь багц программын доторх модулийг заана.

Шинж чанарын тодорхойлолтын хэсэг болсон *varname*- хувьсагчид модулийн *attrname* шинж чанарын утгыг холбож өгнө.Жишээлбэл:

from MyModule import f as foo

attrname ба *varname* нь энгийн нэрс байна.

The from...import * statement

from MyModule import *

* тэмдэг нь *modname* нэртэй модулийн бүх шинж чанарыг авч энэ модулийн нийтийн хувьсагчид холбоно.

Хэвийн илэрхийлэл ба re модуль

Хэвийн илэрхийлэл (RE) нь хэв загварыг илэрхийлэх тэмдэгт мөр юм. Хэвийн илэрхийлэл (RE)-ийг ашигласнаар, дурын тэмдэгт мөрийг хэв загвараар шалгаж тэмдэгт мөрийн аль 1 хэсэг нь хэв загварт таарч байгааг үздэг.

Python хэлний хэвийн илэрхийлэл (RE)-ийн үүргийг re модуль гүйцэтгэнэ. Хэв загварын тэмдэг ба заавал биш флагаас хэвийн илэрхийлэл (RE) обьектийг compile функц үүсгэнэ. Хэвийн илэрхийлэл (RE)-ийн обьектийн метод нь RE нь тэмдэгт мөрөнд таарч байгааг шалгах ба орлуулалтыг хийнэ. re модуль нь хэвийн илэрхийлэл (RE)-ийн методуудтай ижил функцүүдтэй бөгөөд хэвийн илэрхийлэл (RE)-ийн хэв загварыг эхний параметр болгон авдаг.

Хэв загварын мөрийг бичих бичиглэл

Хэв загварын мөр нь дараах бичиглэлээр хэвийн илэрхийлэл (RE)-ийг илэрхийлнэ.

- Тоо ба үсгэн тэмдэгтүүд нь өөрсдийгөө шууд илэрхийлнэ. Хэв загвар нь үсэг ба тооноос бүтсэн хэвийн илэрхийлэл (RE) нь зөвхөн ижил тэмдэгт мөртэй нийцнэ.
- Урд нь урагшаа налуу зураас(\) орсон үед олон тоо буюу үсгэн тэмдэгт нь тусгай утгыг авна.

- Цэг тэмдэглэл нь өөр замаар үйлчилнэ. (Урдаа урагшаа налуу зураастай бол өөрийн утгаар, эсрэг тохиолдолд тусгай утгатай байна.) self-matching when escaped, special meaning when unescaped.
- Хос урагшаа налуу зураас нь урагшаа налуу зураастай таарна.

Хүснэгт 9-2 -д хэвийн илэрхийлэл (RE)-ийн бичиглэл дэх тусгай элементүүдийг жагсаажээ. Хэв загварын тэмдэгт мөр ба заавал биш флагийг өөрчлөх замаар зарим элементийн утгыг өөрчилж болно.

Хүснэгт 9-2. Хэвийн илэрхийлэл (RE)-ийн хэв загварын бичиглэл

Элемент	**Утга**
.	\n –ээс бусад дурын тэмдэгт (Хэрэв DOTALL бол бас \n тэмдэгтийг илэрхийлнэ)
^	Тэмдэгт мөрийн эхнээс харьцуулна (Хэрэв MULTILINE бол \n тэмдэгтийн дараагаас харьцуулна. If MULTILINE also matches after \n)
$	Тэмдэгт мөрийн төгсгөлөөс харьцуулна. (Хэрэв MULTILINE бол \n тэмдэгтийн өмнөхөөс харьцуулна if MULTILINE, also matches before \n)
*	Өмнөх хэвийн илэрхийлэл (RE) –ийг 0 буюу түүнээс олон удаа таарахыг зөвшөөрнө. Боломжтой бол хэдэн ч удаа таарч болно.Хомхой хувилбар
+	Өмнөх хэвийн илэрхийлэл (RE) –ийг 1 буюу түүнээс олон удаа таарахыг зөвшөөрнө. Боломжтой бол хэдэн ч удаа таарч болно. Хомхой хувилбар
?	Өмнөх хэвийн илэрхийлэл (RE) –ийг 0 буюу 1 удаа таарахыг зөвшөөрнө. (боломжтой бол 1 удаа таарч болно. Хомхой хувилбар
*?, +?, ??	*, +, ба ?- ний ховдог биш хувилбар (боломжийн хэрээр цөөн таарах)
{*m,n*}	Өмнөх хэвийн илэрхийлэл (RE) нь *m* -ээс *n* удаа таарч болно.

Хүснэгт 9-2. Хэвийн илэрхийлэл (RE)-ийн хэв загварын бичиглэл

Элемент	**Утга**
	Хомхой хувилбар
{*m*,*n*}?	Өмнөх хэвийн илэрхийлэл (RE) нь *m* -ээс *n* удаа таарч болно. Хомхой биш хувилбар
[...]	Хаалтан доторх тэмдэгтүүдээс аль нэг нь таарч болно.
|	Босоо зураасны өмнөх ба хойнох илэрхийллийн аль 1 нь таарч болно.
(...)	Хаалтан дах хэвийн илэрхийлэл (RE) нь таарах ба ( ) хаалт нь бүлэглэх үүргийг гүйцэтгэнэ.
(?iLmsux)	Заавал биш флагийг тохируулах өөр нэг арга. Харьцуулалтад нөлөөлөхгүй
(?:...)	(...)-тай адил, гэхдээ бүлэглэх үүрэггүй.
(?P<*id*>...)	(...)-тэй адил гэхдээ бүлэг нь *id* гэсэн нэрийг авна.
(?P=*id*)	*id* нэртэй бүлэгт өмнө юу ч таарсан байсан энэ бүлгийн илэрхийлэлтэй таарна.
(?#...)	Ийм хаалтан дах илэрхийлэл нь тайлбар болно. Харьцуулалтад нөлөөлөхгүй
(?=...)	Lookahead assertion: Хэрэв ийм хаалтан дах хэвийн илэрхийлэл (RE) таарч байвал өмнөх хэвийн илэрхийлэл (RE) таарах ба илэрхийллийн аль нэг хэсгийг зарцуулахгүй. Жишээ нь: Isaac (?=Asimov) гэсэн хэвийн илэрхийлэлд 'Isaac '-ийн араас 'Asimov' дагалдан орсон тохиолдолд л 'Isaac ' таарна.
(?!...)	Negative lookahead assertion: Хэрэв ийм хаалтан дах хэвийн илэрхийлэл (RE) таарахгүй байвал өмнөх хэвийн илэрхийлэл (RE) таарах ба илэрхийллийн аль нэг хэсгийг зарцуулахгүй. Жишээ нь: Isaac (?!Asimov) гэсэн хэвийн илэрхийлэлд 'Isaac '-ийн

Хүснэгт 9-2. Хэвийн илэрхийлэл (RE)-ийн хэв загварын бичиглэл

Элемент	**Утга**
	араас 'Asimov' дагалдан ороогүй тохиолдолд л 'Isaac ' таарна.

(?<=...)	Lookbehind assertion: Хэрэв хаалтан дах хэвийн илэрхийлэл (RE) –ийн одоогийн байрлалд төгсгөл нь давхацсан илэрхийлэл олдож байвал таарна (хаалтан дах хэвийн илэрхийлэл (RE)-д таарсан илэрхийлэл нь тогтмол урттай байна.) Жишээ нь
	`>>> import re`
	`>>> m = re.search('(?<=abc)def', 'abcdef')`
	`>>> m.group(0)`
	`'def'`

(?<!...)	Lookbehind assertion: Хэрэв хаалтан дах RE –ийн одоогийн байрлалд төгсгөл нь давхацсан илэрхийлэл олдохгүй байвал таарна (хаалтан дах хэвийн илэрхийлэл (RE)-д таарсан илэрхийлэл нь тогтмол урттай байна.)
number	*number* нэртэй бүлгийн тоотой тэнцүү удаа өмнө нь олдсон бүлэг таарна. (бүлэг нь 1 –ээс 99 хүртэл дугаарлагдан)
\A	Бүх тэмдэгт мөрийн эхнээс харьцуулна. Илэрхийллийн эхэнд байх хоосон мөртэй таарна.
\b	Үгийн эхэн буюу төгсгөлд байх хоосон тэмдэгт мөртэй таарна. Үг тоон ба үсгийн дараалал(\w -г үз)
\B	Үгийн эхэн буюу төгсгөлөөс бусад байрлалд байх хоосон тэмдэгт мөртэй таарна.

Хүснэгт 9-2. Хэвийн илэрхийлэл (RE)-ийн хэв загварын бичиглэл

Элемент	**Утга**
\d	[0-9]илэрхийлэл шиг 1 ширхэг тоотой таарна.
\D	[^0-9]илэрхийлэл шиг 1 ширхэг тоо биш тэмдэгттэй таарна.
\s	[\t\n\r\f\v] илэрхийлэл шиг энтэр, мөрийн төгсгөл, табуляц зэрэг тэмдэгтийн нэгтэй таарна.
\S	[^\t\n\r\f\v] илэрхийлэл шиг энтэр, мөрийн төгсгөл, табуляцаас бусад нэг тэмдэгттэй таарна.
\w	LOCALE , UNICODE –ийг тодорхойлоогүй бол тоон буюу үсгэн тэмдэгттэй таарна. \w нь [a-zA-Z0-9_] илэрхийлэлтэй адил
\W	\w ийн эсргээр тоон буюу үсэг биш тэмдэгттэй таарна.
\Z	Бүх тэмдэгт мөрийн төгсгөлөөс харьцуулна. Илэрхийллийн төгсгөлд байх хоосон мөртэй таарна.
\\	1 урагшаа налуу зураастай таарна.

Хэвийн илэрхийллийн энгийн хэллэгүүд

'.*' гэсэн хэвийн илэрхийлэлийн хэв загвар нь “дурын тэмдэгт дурын удаа давтагдан орно ” гэсэн утгатай. Өөрөөр хэлбэл '.*' нь хоосон тэмдэгт мөрийг оролцуулаад харьцуулж байгаа тэмдэгт мөрийн бүх дэд тэмдэгт мөрүүдтэй таарна. '.+' нь өмнөхтэй ижил боловч хоосон биш тэмдэгт мөрүүд таардаг.Жишээ нь:

'pre.*post'
'pre' –ийг агуулсан бөгөөд түүний дараагийн тэмдэгтүүд нь 'post'–ийг агуулсан бүх тэмдэгт мөрүүдтэй таарна. Сүүлийнх нь өмнөхтэйгөө дараалан орсон байсан ч бас тохирно. (Жишээ нь: 'prepost' ба 'pre23post' –хоёулаа таарна.)

Өөр 1 хэв загвар

'pre.+post'

matches only if 'pre' and 'post' are not adjacent (e.g., it matches 'pre23post' but does not match 'prepost'). Both patterns also match strings that continue after the 'post'. To constrain a pattern to match only strings that end with 'post', end the pattern with \Z. For example:

'pre' ба 'post' –ийг энэ дарааллаар агуулахдаа энэ 2 нь залгаа биш байх тэмдэгт мөрийг агуулна. (Жишээ нь: 'pre23post'таарах ба 'prepost' таарахгүй.) 'post'- ийн ард үргэлжлэх тэмдэгтүүд энэ 2 хэв загварт хоёуланд нь таарна. Зөвхөн 'post'-оор төгссөн тэмдэгт мөр таардаг болгон хязгаарлахын тулд \Z-ийг нэмнэ.

r'pre.*post\Z'
'prepost'таарах боловч нь 'preposterous' таарахгүй. Урагшаа налуу зураас агуулсан боловсроогүй тэмдэгт мөрийн бичиглэлээр хэв загварыг илэрхийлэх хэрэгтэй. Урагшаа налуу зураасыг алгасалгүй бичих үүднээс rE хэв загварын боловсроогүй тэмдэгт мөрийн бичиглэлийг ашиглана уу!

RE хэв загварын өөр нэг байнга ашигладаг элемент нь \b бөгөөд энэ нь үгийн эхлэл төгсгөлийг илэрхийлнэ. Та 'his' тэмдэгт мөр 'this' , 'history' зэрэг үгийн хэсэг болон орсон тохиолдлуудыг авахгүй тусдаа үг болон орсоныг тааруулья гэвэл RE хэв загвар нь дараах хэлбэртэй байна.

r'\bhis\b'

Эхлэл ба төгсгөлд үгийн заагийг бичсэн. 'her' –ээр эхэлсэн 'her','hermetic' зэрэг үгийг тааруулахын тулд дараах хэв загварыг ашиглана. Энэ хэв загварт дундаа буюу төгсгөлдөө 'her'-ийг агуулсан үгс таарахгүй.

r'\bher'
Харгалзах тэмдэгт мөрөнд үгийн төгсгөлд биш, өмнө нь үгийн заагийг бичсэн. 'its'-ээр төгссөн 'its', 'fits' зэрэг үгийг тааруулахын тулд дараах хэв загварыг ашиглана. Энэ хэв загварт эхэндээ буюу дундаа 'its'-ийг агуулсан 'itsy', 'jujitsu'зэрэг үгс таарахгүй.

r'its\b'

Харгалзах тэмдэгт мөрөнд үгийн эхэнд биш, төгсгөлд нь үгийн заагийг бичсэн. Үгийн эхлэл буюу төгсгөлийг тааруулж авахын оронд бүтэн үгээр нь авахын тулд гэсэн хэв загварын элемэнтийг нэмж өгнө. 'her'–ээр эхэлсэн бүтэн үгийн тааруулахын тулд дараах хэв загварыг ашиглана.

r'\bher\w*'
'its'-ээр төгссөн бүтэн үгийг тааруулья гэвэл:

r'\w*its\b'

Заавал биш флагууд

(? ба)-ийн хооронд iLmsux үсгүүдийн 1 буюу хэд хэд нь орсон элементтэй хэв загвар бүхий хэвийн илэрхийлэл (RE) нь re.модулийн compile функцийг *flags* параметртэйгээр ажиллуулалгүйгээр хэвийн илэрхийлэл (RE) –ийн сонголтыг өөрчлөх боломжийг олгоно.

Хэвийн илэрхийлэл (RE)-ийг бүхэлд хамардаг сонголт нь энэ илэрхийллийн хэв загварын хаана ч таарч болно. Илэрхий байх үүднээс сонголтуудыг үргэлж эхэнд нь байрлуулдаг. x нь сонголтуудын дунд байх үед заавал эхэнд байрлуулах ба ингэснээр энэ хэв загварыг задлах Python-ны арга зам өөрчлөгдөнө. *flags* гэсэн илэрхий параметрийг ашиглах нь хэв загвар дотор сонголтын 1 элемент байрлуулахаас илүү ойлгомжтой байдаг. compile функцийн *flags* параметр нь битийн OR-оор ялгах боломжтой бүхэл тоо байдаг ба re модулийн нэг буюу хэд хэдэн шинж чанарыг илэрхийлнэ.Шинж чанар нь богино нэр ба тохиромжтой байхын үүднээс урт нэртэй(Том үсэгээр бичсэн олон үсэгтэй) байна. Урт нэр нь илүү ойлгомжтой илүү хэрэгтэй байдаг.

I буюу IGNORECASE

Том жижиг үсгийг ялгахгүй харьцуулдаг болгоно.

L буюу LOCALE

Тухайн үйлдлийн системийн хэл нь үсэг буюу тоонд ямар тэмдэгтүүдийг хамааруулж байгаагаас \w, \W, \b, and \B-ийн харьцуулалт шалтгаална.

M буюу MULTILINE

^ and $ тусгай тэмдэгтүүдийг мөр бүрийн эхлэл, төгсгөлтэй таардаг байсныг бүх тэмдэгт мөрийн эхлэл төгсгөлтэй таардаг болгоно.

S буюу DOTALL

. тусгай тэмдэгт нь шинэ мөр(\n) тэмдэгтийг оруулан бүх тэмдэгт илэрхийлэх эсэхэд нөлөөлнө.

U буюу UNICODE

Unicode-д ямар үсгүүдийг үсэг буюу тоо гэж үзэхээс шалтгаалан\w, \W, \b, and \B харьцуулалт хийгдэнэ.

X буюу VERBOSE

Илэрхийллийн хэв загварт байгаа тэмдэгтийн олонлогт багтаагүй энтэр, мөрийн төгсгөл, табуляц зэрэг тэмдэгтийг үл хэрэгсэх ба # -аар хэлсэн тайлбарыг мөрийн төгсгөл хүртэл үргэлжилдэг болгоно.

Дараах 3 мөр нь "hello" үгийг том жижиг үсэг ялгалгүй харьцуулна.

```
import re r1 = re.compile(r'(?i)hello')
r2 = re.compile(r'hello', re.I)
r3 = re.compile(r'hello', re.IGNORECASE)
```

The third approach is clearly the most readable, and thus the most maintainable, even though it is slightly more verbose. The raw-string form is not necessary here, since the patterns do not include backslashes; however, using raw strings is innocuous, and is the recommended style for clarity.

re.VERBOSE сонголт нь энтэр, мөрийн төгсгөл, табуляц зэрэг тэмдэгт ба тайлбарыг зохистой ашигласнаар илүү уншихад ойлгомжтой хэв загварыг бичих боломж олгоно. Хэвийн илэрхийлэл(RE) ийн түвэгтэй нуршуу хэв загвар нь нэг мөрөнд багтахгүй тул ийм хэв загварыг бичихдээ гурван хашилттай тэмдэгт мөрийг хэрэглэхийг хүсэх нь зайлшгүй.Жишээ нь:

```
repat_num1 = r'(0[0-7]*|0x[\da-fA-F]+|[1-9]\d*)L?\Z'
repat_num2 = r'''(?x)          # pattern matching integer numbers
          (0 [0-7]*       | # octal: leading 0, then 0+ octal digits
           0x [\da-fA-F]+  | # hex: 0x, then 1+ hex digits
           [1-9] \d*       ) # decimal: leading non-0, then 0+ digits
           L?\Z             # optional trailing L, then end of string
          '''
```

Энэ 2 хэв загвар ижил, гэхдээ сүүлийнх нь сул зай төрлийн тэмдэгт ба тайлбарыг чөлөөтэй хэрэглэсэн тул илүү ойлгомжтой байна.

Хэвийн илэрхийллийн обьект

Хэвийн илэрхийллийн обьект *r* нь хэрхэн үүсгэснийг нэгбүрчлэн илэрхийлэх зөвхөн уншигдах дараах шинж чанаруудтай.

flags

flags нь байхгүй бол compile функц руу *flags* параметрийн дамжуулна.

groupindex

A dictionary whose keys are group names as defined by elements (?P<*id*>); the corresponding values are the named groups' numbers

Энэ нь (?P<*id*>)элементээр тодорхойлогддог бүлгийн нэрсээр түлхүүр хийсэн хэш толь юм. Харгалзах утга нь бүлгийн дугаарууд юм.

pattern

r - ийн эмхтгэх илэрхийллийн хэв загвар

Эмхтгэгдсэн RE обьектоос эдгээр шинж чанаруудыг уншихад хялбар ба харин тэднийг тусад нь өөрчлөх боломжгүй.

r RE обьект нь тэмдэгт мөрнөөс *r* –д таарсан элементийг олж энэ элементүүдийг хувьсагчаар орлуулах үйлдлийг гүйцэтгэнэ.

Findall

r.findall(*s*)

r обьект нь бүлэггүй бол энэ нь *r* обьекттой үл давхцан таарч байгаа *s* –ийн бүх дэд тэмдэгт мөрийн жагсаалтыг буцаана. Жишээ нь : файл дах бүх үгийг хэвлэе.

For example, to print out all words in a file, one per line:

```
import re reword = re.compile(r'\w+')
for aword in reword.findall(open('afile.txt').read( )):
   print aword
```

r обьект нь 1 бүлэгтэй бол *r* обьектийн бүлэгтэй таарч байгаа *s* –ийн бүх дэд тэмдэгт мөрийн жагсаалтыг буцаана. Жишээ нь сул зай төрлийн тэмдэгтээр төгссөн хэвлэхийн тулд дээрх жишээний 1 командыг өөрчилнө.

```
reword = re.compile('(\w+)\s')
```

r обьект нь 1-ээс олон бүлэгтэй бол *r* обьекттой үл давхцан таарч байгаа tuple-ийн жагсаалтыг findall функц буцаана. tuple бүр нь *r* обьектийн бүлэг бүрийн хувьд 1, 1-ээр *n*

элементтэй байх ба эдгээр нь энэ бүлэгт таарч байгаа *s* –ийн дэд тэмдэгт мөрүүд юм. Жишээ : доод тал нь 2 үгтэй мөрийн эхний ба сүүлийн үгийг хэвлэе.

```
import re first_last = re.compile(r'^\W*(\w+)\b.*\b(\w+)\W*$',re.MULTILINE)
for first, last in \
first_last.findall(open('afile.txt').read( )):
   print first, last
```

Finditer

r.finditer(*s*)

finditer нь findall –тай адил боловч тэмдэгт мөр(буюу tuple) ийн жагсаалтыг буцаахын оронд элементүүд нь таарсан обьектүүд байх давтагчийг буцаана. Ихэнх тохиолдолд finditer нь findall аас илүү уян хатан байдаг.

Match

r.match(*s,start*=0,*end*=sys.maxint)

start эхлэх индексээс *end* эцсийн индекс хүртэлх *s*-ийн дэд тэмдэгт мөр нь *r* обьекттой таарч байгаа харгалзах тохирлын обьектийг буцаана. Эсрэг тохиолдолд None -ийг буцаана. Үнэхээр *s* тэмдэгт мөрийн эхлэх байрлалаас match нь эхлэх цэгээ тогтооно. *r* обьекттой таарсан тохирол(match)-д зориулан *s* тэмдэгт мөрийн *start*-аас өмнөх дурын байрлалаас хайхын тулд *r*.match-ийг биш *r*.search-ийг дуудна.Жишээ нь: файл дахь цифрээр эхэлсэн мөрүүдийг хэвлэе.

```
import re digs = re.compile(r'\d+')
for line in open('afile.txt'):
if digs.match(line): print line,
```

search

r.search(*s,start*=0,*end*=sys.maxint)

start индексээс өмнө эхлээгүй, *end* индексээс хойш дуусаагүй *s*-ийн хамгийн зүүн талын дэд мөр нь *r* обьекттой таарч байх харгалзах тохирлын обьектийг буцаана. Ийм дэд мөр олдохгүй үед search нь None –ийг буцаана. Файл дахь цифрээр эхэлсэн мөрүүдийг хэвлэх жишээ нь дараах байдалтай байна.

```
import re digs = re.compile(r'\d+')
for line in open('afile.txt'):
   if digs.search(line): print line,
```

split

r.split(*s,maxsplit*=0)

r обьектоор *s* –ийг хуваасан тэмдэгт мөрүүдийг жагсаалтыг буцаана. (*r* обьекттой таарч байгаа хоосон биш, үл давхцах *s* –ийн дэд мөрүүд) Returns a list *L* of the splits of *s* by *r* (i.e., the substrings of *s* separated by nonoverlapping, nonempty matches with *r*). Жишээ нь: 'hello'дэд мөрийн бүх тохиолдлыг устгах 1 арга нь дараах байдалтай байна.

```
import re rehello = re.compile(r'hello', re.IGNORECASE)

astring = ''.join(rehello.split(astring))
```

r обьект нь *n* бүлэгтэй бол хуваалтын хосуудын хооронд *n* ширхэг илүү элементүүд орно. Энэ *n* ширхэг нэмэлт элементийн элемент бүр нь тухайн тохирол дохь *r*-обьектийн харгалзах бүлэгт таарах *s* -ийн 1 дэд мөр байна. Энэ тохиролд тухайн бүлэг байхгүй бол энэ элемент нь None утгатай байна. Сул зай төрлийн тэмдэгт нь : (2 цэг) ба цифрийн дунд орсон үед устгах нэг жишээг дараах жишээгээр үзүүлье.

```
import re re_col_ws_dig = re.compile(r'(:)\s+(\d)')

astring = ''.join(re_col_ws_dig.split(astring))
```

maxsplit нь 0-ээс их бол дээр өгүүлсэн *n* элементээс хуваалт бүр нь тогтох ихдээ *maxsplit* ширхэг хуваалт *L* –жагсаалтад байна. Дээрх жишээний сүүлийн командыг дараах мөрөөр соливол хамгийн эхэнд тохиолдсон зөвхөн 1 'hello'-г устгадаг болно.

```
astring = ''.join(rehello.split(astring, 1))
```

sub

r.sub(*repl,s,count*=0)

Функц шиг дуудагдах обьект буюу тэмдэгт мөрийн аль нэг нь байж болох - *repl* – ээр солигдох *r* обьекттой таарах үл давхцах тохиролыг агуулах *s* –тэмдэгт мөрийг буцаана. Зөвхөн өмнөх тохиролтой ойр биш тохиолдолд л хоосон тохиролыг

сольж болно. *count* нь 0-ээс их бол *s* доторх *r* обьектийн эхний *count* ширхэг тохирлыг солино. *count* нь 0- бол *r* обьектийн бүх тохирлыг солино. Хамгийн эхэнд тохиолдсон зөвхөн 1 'hello'-г том жижиг үсэг харгалзалгүй устгах өөр 1 жишээг үзүүлье.

```
import re rehello = re.compile(r'hello', re.IGNORECASE)

astring = rehello.sub('', astring, 1)
```

Сүүлийн 1 гэсэн параметр байхгүй бол бүх 'hello'-г устгана.

repl нь дуудагдах обьект байгаа үед харьцуулах обьект гэсэн 1 л параметртэй байх ба харьцуулах үеийн солих үйлдэлд ашиглагддаг тэмдэгт мөр буюу None-ийг буцаана. None нь '' хоосон тэмдэгт мөртэй адил. Энэ тохиолдолд sub нь зохих харьцуулах обьектийг параметрээр дамжуулан *repl*-ийг , өөрийн солих обьектийн тохирол бүрийн хувьд дууддаг. Том, жижиг үсгийг ялгалгүй 'h'–ээр эхэлж –'o'оор төгссөн үгийн бүх тохиолдлыг том үсгийн хэлбэрт шилжүүлэх жишээг үзье.

```
import re h_word = re.compile(r'\bh\w+o\b', re.IGNORECASE)

def up(mo): return mo.group(0).upper( )

astring = h_word.sub(up, astring

)
```

repl нь тэмдэгт мөр бөгөөд буцан заах заагч биш үед sub нь –ийг *repl*- ийг нь солих хэсэг болгон ашигладаг. Буцан заах заагч нь \g<*id*> буюу *dd* хэлбэртэй *repl*-ийн дэд тэмдэгт мөр юм. Үүнд *id* нь *r* обьектийн 1 бүлгийн нэр ба тухайн обьектийн хэв загварын (?P<*id*>) бичиглэлд бичиж өгнө. *dd* нь бүлгийн дугаар гэж ойлгогддог 1 буюу 2 цифр юм. Нэрлэгдсэн буюу дугаарлагдсан буцаан заах заагч нь өөрийн зааж байгаа *r* обьектийн бүлэгт таарч байгаа дэд мөрөөр солигдоно. Жишээ нь:Үүгээр үг бүрийг {} хаалтанд хашиж болно.

```
import re grouped_word = re.compile('(\w+)')

astring = grouped_word.sub(r'{\1}', astring)
```

subn

r.subn(*repl,s,count*=0)

subn нь sub тай адил гэхдээ (*new_string*, *n*)гэсэн хосыг буцаана. Үүнд: *n* нь –ийн гүйцэтгэсэн солих (орлуулах) үйлдлийн тоо. Том жижиг үсэг харгалзахгүйгээр 'hello' дэд тэмдэгтийн тохиолдсон тоог тоолох 1 жишээг үзүүлье.

```
import re rehello = re.compile(r'hello', re.IGNORECASE)
junk, count = rehello.subn('', astring)
print 'Found', count, 'occurrences of "hello"'
```

re модулийн функцүүд

‘Заавал биш флаг’ хэсэгт re модулийн шинж чанаруудыг тухай өгүүлсэн. Хэвийн илэрхийллийн обьектийн(findall, finditer, match, search, split, sub, ба subn зэрэг) метод бүр нь 1 функц, хэвийн илэрхийллийн обьект (RE)-рүү сохроор эмхтгэгддэг хэв загварын мөр буюу эхний нэмэлт параметртэй байна.

Ихэнхдээ хэв загварын мөрийг хэвийн илэрхийллийн обьект (RE)-рүү илэрхий байдлаар эмхтгэх нь зохимжтой боловч заримдаа хэв загварын мөрийг 1 л удаа хэрэглэхдээ re модулийн функцийг дуудахад нь үл ялиг эвтэйхэн байдаг. Том жижиг үсэг харгалзахгүйгээр 'hello' дэд тэмдэгтийн тохиолдсон тоог тоолох 1 функц ашигласан хувилбарыг үзүүлье.

```
import re junk, count = re.subn(r'(?i)hello', '', astring)
print 'Found', count, 'occurrences of "hello"'
```

RE конголт(энд (?i)) нь RE-ийн хэв загварт орсон учраас re модулийн функцүүд нь *flags* параметрийг хүлээн авахгүй. re модуль нь хэв загвараас дамжин функцэд ирсэн хэвийн илэрхийлэл (RE)ийн обьектийг дотроо нөөц хэлбэрээр үүсгэх ба дараа нь re.purge()–ийг дуудаж санах ойг дахин ашиглах үүднээс чөлөөлнө.

re модуль нь (ихэвчлэн хэв загварын мөрөнд) алдаа гарахад үүсэх exception-ны error класс ба өөр 2 функцтэй байдаг.

compile	compile(*pattern,flags*=0) *pattern*- хэв загварын мөрийг ялгаж хэвийн илэрхийлэл (RE)ийн обьектийг үүсгэж буцаана.
Escape	escape(*s*)

compile	compile(*pattern,flags*=0) *pattern*- хэв загварын мөрийг ялгаж хэвийн илэрхийлэл (RE)ийн обьектийг үүсгэж буцаана.
	s тэмдэгт мөрийн үсэг ба тоо биш тэмдэгтийг \ (урагшаа налуу) тэмдэгтээр илэрхийлсэн хэлбэрээр сольсон хуулбарыг буцаана. Энэ нь *s* тэмдэгт мөрийг хэвийн илэрхийлэл (RE)ийн хэсэг болгон ашиглахад хэрэглэгдэнэ.

Системийн скрипт бичих тойм

Энэ хэсэгт системийн программчлалын зарчмуудыг үзэхээс өмнө sys ба os стандарт санг сонирхьё. Эдгээр 2 том модулийн шинж чанарыг тоог хэлж болно. Энэ тоо нь Python-ны хувилбар ба орчноос хамаарч өөр өөр байна.

```
>>> import sys, os
>>> len(dir(sys))         # 56 attributes
56
>>> len(dir(os))          # 118 on Windows, more on Unix
118
>>> len(dir(os.path))     # a nested module within os
43
```

Модуль бүрийн элемент бүрийг тайлбарлах шаардлагагүй гэж үзээд хийхийг хүсэж байгаа зүйл нь та өөрөө хэрхэн нэгбүрчилсэн мэдээллийг олж авахыг үзүүлье. Энэ бодлого нь системийн скрипт бичих гол хэдэн зарчмыг танилцуулах нэгэн шалтаг болж өгнө. Энэ маягаар баримтыг форматлах эхний скриптийг бичье.

Python-ны систем модулиуд

Python-тэй хамт ирдэг хамгийн системийн түвшний интерфэйс нь sys ба os модулиуд юм. Зарим стандарт модуль нь энэ домэйнд харьяалагддаг.Эдний дунд дараах процедурууд байна.

glob

For filename expansion

Файлыг нэрийг шүүж авах

socket

Процессийн хоорондох холбоо (IPC) ба сүлжээний холболт

thread and queue

Зэрэгцээ трийд

time

Системийн цагийг унших, өөрчлөх

fcntl

Файлын доод түвшний удирдлага

sys модультай танилцах нь

Python-ны системтэй холбоотой багажийн үндсэн хэсэг нь sys ба os модуль юм.Том жишээ авч үзэхээс өмнө энэ 2 модулийн зарим багажтай танилцах богино аялал хийе. Эхлээд sys модулийн бүх шинж чанарыг жагсаан үзүүлэх, функцэд түүнийгээ дамжуулах гэсэн 2 жишээг үзье.

Суурь ба хувилбар

Бусад ихэнх модультай адил sys нь үйлдэл хийх мэдээллийн нэр ба функцүүдтэй. Жишээ нь: Суурь код ажиллаж байгаа үйлдлийн системийн нэр, энэ компьютер дээрх боломжит хамгийн их бүхэл тоо, Python хөрвүүлэгчийн хувилбарын дугаарыг буцаах шинж чанаруудыг sys модуль агуулдаг.

```
C:\...\PP3E\System>python
>>> import sys
>>> sys.platform, sys.maxint, sys.version
('win32', 2147483647, '2.4 (#60, Nov 30 2004, 11:49:19) [MSC v.1310 32 bit (Intel)]')
>>>
>>> if sys.platform[:3] == 'win': print 'hello windows'
...
hello windows
```

Python нь нэг сууриас нөгөө суурь руу шилжих боломжтой хэдий ч, янз бүрийн компьютерийн системээс хамааран өөрөөр ажиллах программыг бичье гэвэл sys.platform тэмдэгт мөрийг шалган үл зөөгдөх багажийг дээр бичсэн шиг if нөхцөлд бичиж болно.

Модулийг хайх зам

sys модуль нь модулийг хайх замыг интерпретатораас эсвэл Python-ны программ дотроос үзэх боломжийг олгодог. sys.path нь Python-ны интерпретатор ажиллаж байхдаа хайлтад хэрэглэх зөв замыг илэрхийлэх тэмдэгт мөрийн жагсаалт юм. Модулийг импортлох үед Python нь энэ жагсаалтыг зүүнээс баруун тийш уншиж , энд дурдагдсан директор бүрээс модулийн файлыг хайна. Үүний учир нь зориулалтын дагуу хайх зам нь тогтоогдсон эсэхийг шалгах байрлал нь энэ юм.

Python нь PYTHONPATH-ийг таны харж байгаагаас өөрөөр уншдаг огт боломжгүй зүйл биш юм. Системийн конфигурацийн файл танд хэвийн мэт үзэгдэж байсан ч бичиглэлийн алдаатай байвал PYTHONPATH-ийн тохиргоо нь будлиантаж болно. Python –ны интерпретатор анх ажилласан директорт байгаа дурын .pth файлын агуулга, PYTHONPATH- –ны тохиргооноос sys.path жагсаалтыг нь авдаг. sys.path –ийг интерпретатороос ажиллуулахад, цөөн хэдэн директор нь PYTHONPATHsys.path гаднах байх бөгөөд скриптийн home директорыг агуулна.Стандарт сангийн директорууд нь хэрхэн суулгаснаас хамааран өөр өөр байна.

```
>>> sys.path
['', 'C:\\PP3rdEd\\Examples',  ...plus standard paths deleted... ]
```

Гайхамшигтай нь sys.path нь программ дотроос өөрчлөгдөх боломжтой. Хандах хэрэгцээтэй байгаа эх код бүхий директорыг ажиллах үедээ жагсаалтын append, del, зэрэг үйлдлүүдийг ашиглан хайлтын зам руу нэмж болно. Хэзээ өөрчилснөөс үл хамааран sys.path-ийн одоогийн тохиргоог модулийг импортлохдоо хэрэглэдэг.

```
>>> sys.path.append(r'C:\mydir')
>>> sys.path
['', 'C:\\PP3rdEd\\Examples',  ...more deleted..., 'C:\\mydir']
```

sys.path ийг өөрчлөх нь PYTHONPATH хувьсагчийн утгыг шинэчлэх 1 арга боловч 1 муу талтай. Өөрчлөлт нь Python-ны программ дуустал хадгалагдах ба программыг шинээр ажиллуулах бүрт өөрчлөлтийн дахин хийх шаардлагатай. Зарим программ нь PYTHONPATH-хувьсагчаас хамааралгүй байдаг ба эхлэх бүртээ модуль импортлох директоруудыг sys.path-д олгодог.

Ачаалагдсан модулийн хүснэгт

sys.modules нь *name:module* элемент бүхий ажиллаж буй Python программ буюу түүний хэсгийн импортолсон бүх модулийн нэрийг агуулсан хүснэгт юм.

```
>>> sys.modules
{'os.path': <module 'ntpath' from 'C:\Program Files\Python\Lib\ntpath.pyc'>,...

>>> sys.modules.keys( )
['os.path', 'os', 'exceptions', '_ _main_ _', 'ntpath', 'strop', 'nt', 'sys',
'_ _builtin_ _', 'site', 'signal', 'UserDict', 'string', 'stat']

>>> sys
<module 'sys' (built-in)>
>>> sys.modules['sys']
<module 'sys' (built-in)>
```

Энэ хүснэгтийг ашиглан ачаалагдсан бүх модулийг харуулах буюу боловсруулах программыг бичиж болно.

sys also exports tools for getting an object's reference count used by Python's garbage collector (getrefcount), checking which modules are built into this Python (builtin_module_names), and more.

Exception-ний тухай

sys модулийн зарим шинж чанар нь Python-ний саяхан үүссэн exception-ний тухай мэдээллийг авах боломжийг олгоно. Эдгээр нь exception-ийг илүү ерөнхий байдлаар боловсруулах үед хэрэглэхэд эвтэйхэн байдаг. Жишээ нь: sys.exc_info функц нь сүүлийн exception –ний төрөл, утга, обьектийг буцаана.

```
>>> try:
...     raise IndexError
... except:
...     print sys.exc_info( )
...
(<class exceptions.IndexError at 7698d0>, <exceptions.IndexError instance at
797140>, <traceback object at 7971a0>)
```

График интерфэйс буюу веб хуудас дээр гарах алдааны мэдээллийг өөрийн хүссэн хэлбэрээр өөрчлөхөд энэ мэдээллийг хэрэглэнэ. Энэ функцээс буцаах утгын эхний 2 элемент нь шууд хэвлэгдэх тэмдэгт мөр байх ба 3-р элемент нь traceback стандарт модулиар боловсруулагдах traceback обьект байна.

```
>>> import traceback, sys
>>> def grail(x):
...     raise TypeError, 'already got one'
...
```

```
>>> try:
...     grail('arthur')
... except:
...     exc_info = sys.exc_info( )
...     print exc_info[0]
...     print exc_info[1]
...     traceback.print_tb(exc_info[2])
...
exceptions.TypeError
already got one
  File "<stdin>", line 2, in ?
  File "<stdin>", line 2, in grail
```

TRaceback модуль нь мессэжийг тэмдэгт мөр шиг хэлбэртэй болгож ба тодорхой файлын обьект руу чиглүүлж болно.

Other sys Module Exports sys модулийн бусад экспорт

sys –модуль нь том бүлгийн хам сэдэвт нийцэх нэмэлт багажийг экспортолдог. Жишээ нь:

- Командын мөрийн параметрүүдийг sys.argv гэж нэрлэгддэг тэмдэгт мөрийн жагсаалтаар илэрхийлнэ.
- sys.stdin, sys.stdout, ба sys.stderr зэрэг стандарт урсгалуудыг ашиглахад бэлэн.
- sys.exit –ийг дуудаж программаас гарч болно.

Эдгээр нь томоохон сэдвийг хөндөх болно.

Os модулийг танилцуулах нь

os нь 2 үндсэн систем модулийн том нь юм. Энэ нь С программ болон системийн скриптээс хэрэглэж болох бүх функц буюу дуудлагыг агуулна. Энэ функцүүд нь директорүүд, процесс, бүрхүүлийн хувьсагч зэргийг боловсруулна. Техникийн хувьд POSIX toolsa үйлдлийн системийн зөөгдөх стандарт, callsalong- os.path –д агуулагдсан сууриас үл хамаарах директорыг боловсруулах системээс тогтоно. Үйл ажиллагааны хувьд, os нь компьютерийн системийн функцүүдтэй холбогдох томоохон зөөгдөх холбогч болж өгнө. os ба os.path дээр бичигдсэн скриптүүд нь дурын суурийн хувьд өөрчлөгдөхгүй. Python эх кодын сангийн директор дах os.py файлыг үзэж, from* командыг ажиллуулж сууриас хамааралтай модулийн нэрсийг гаргана. Сууриас хамааралтай модулиас илүүтэй os –ийг үргэлж импортолсноор суурь ялгаа гарсан ч түүнээс ихэнхдээ чөлөөтэй байна. Зарим суурьт os нь (Unix-ийн доод түвшний функц)- тухайн суурьт

хэрэгцээтэй нэмэлт багажийг агуулдаг. Гэхдээ бүхэлдээ суурь дамжих нь техникийн хувьд боломжтой.

os-ийн дэлгэрэнгүй жагсаалт

os –ийн үндсэн интерфэйсүүдийн тухай үзье. Хүснэгт 3-1 –д үйл ажиллагааны хүрээллээр ангилсан os модулийн энгийн багажийг тоймлоё.

Let's take a quick look at the basic interfaces in os. As a preview, Table 3-1 summarizes some of the most commonly used tools in the os module organized by functional area.

Хүснэгт 3-1. os модулийн энгийн багаж

Бодлого	**Багаж**
Орчны бүрхүүлийн хувьсагч	os.environ
Программыг ажиллуулах	os.system, os.popen, os.popen2/3/4, os.startfile
Процессийг эхлүүлэх	os.fork, os.pipe, os.exec, os.waitpid, os.kill
Файлын хувьсагчтай ажиллах ба түгжих үйлдлүүд	os.open, os.read, os.write
Файлыг боловсруулах	os.remove, os.rename, os.mkfifo, os.mkdir, os.rmdir
Администраторын багаж	os.getcwd, os.chdir, os.chmod, os.getpid, os.listdir
Зөөгдөх шинжийн багаж	os.sep, os.pathsep, os.curdir, os.path.split, os.path.join

Хүснэгт 3-1. os модулийн энгийн багаж

Бодлого	Багаж
Файлын замтай холбоотой багаж	os.path.exists('path'), os.path.isdir('path'), os.path.getsize('path')

Python-ны рилийз болон суурь бүр дээр өөр өөр байх энэ модулийн шинж чанаруудын том хэмжээний жагсаалтыг интерпретатор ашиглан үзэж болно.

```
>>> import os
>>> dir(os)
['F_OK', 'O_APPEND', 'O_BINARY', 'O_CREAT', 'O_EXCL', 'O_NOINHERIT',
'O_RANDOM', 'O_RDONLY', 'O_RDWR', 'O_SEQUENTIAL', 'O_SHORT_LIVED',
'O_TEMPORARY', 'O_TEXT', 'O_TRUNC', 'O_WRONLY', 'P_DETACH', 'P_NOWAIT',
...
 ...10 lines removed here...
...
'popen4', 'putenv', 'read', 'remove', 'removedirs', 'rename', 'renames', 'rmdir',
'sep', 'spawnl', 'spawnle', 'spawnv', 'spawnve', 'startfile', 'stat',
'stat_float_times', 'stat_result', 'statvfs_result', 'strerror', 'sys', 'system',
'tempnam', 'times', 'tmpfile', 'tmpnam', 'umask', 'unlink', 'unsetenv', 'urandom',
'utime', 'waitpid', 'walk', 'write']
```

Үүнээс гадна os.path- модуль нь өөрт агуулагдаж байгаа файл ба директорыг боловсруулах илүү олон багажийг экспортолно.

```
>>> dir(os.path)
['_ _all_ _', '_ _builtins_ _', '_ _doc_ _', '_ _file_ _', '_ _name_ _', 'abspath',
'altsep', 'basename', 'commonprefix', 'curdir', 'defpath', 'devnull', 'dirname',
'exists', 'expanduser', 'expandvars', 'extsep', 'getatime', 'getctime', 'getmtime',
'getsize', 'isabs', 'isdir', 'isfile', 'islink', 'ismount', 'join', 'lexists',
'normcase', 'normpath', 'os', 'pardir', 'pathsep', 'realpath', 'sep', 'split',
'splitdrive', 'splitext', 'splitunc', 'stat', 'supports_unicode_filenames', 'sys',
'walk']
```

Администраторын багаж

Os-ийн зарим энгийн багажийг туршиж үзэцгээе. sys -тэй адил os модуль нь мэдээллийн болон администраторын багажийн цуглуулга юм.

```
>>> os.getpid( )
-510737
>>> os.getcwd( )
'C:\\PP3rdEd\\Examples\\PP3E\\System'

>>> os.chdir(r'c:\temp')
>>> os.getcwd( )
'c:\\temp'
```

Дээр үзүүлснээр os.getpid функц нь дуудсан процессийн процессийн дугаарыг (Ажиллаж буй программын системээс тодорхойлсон цор ганц ялгах нэр) буцаана. os.getcwd нь одоогийн ажлын директорыг буцаана. Одоогийн ажлын директор нь ажиллаж буй скриптийн нэр нь тодорхой замыг агуулаагүй бол энэ скриптийн байрлаж буй директорийн нэрийг хэлнэ. more.py байгаа директорт шилжээд дараах командын ажиллуулна уу!

```
C:\...\PP3E\System>python more.py more.py
```

Оролтын файлын нэр гэсэн параметрт тодорхой директор бүхий замгүй файлын нэрийг олгож дуудсан.Өөр ажлын директороос ажиллуулахыг хүсвэл, өөр директорт шилжихийн тулд os.chdir -ийг дуудах ба программын үлдсэн хэсэг нь энэ шинэ директортой уялдаатай ажиллана. Скриптийг ажиллуулахтай холбогдолтой сэдвийг тайлбарлах үед одоогийн ажлын директор ба түүний модулийг импортлохтой холбогдох холбооны тухай ойлголтыг илэрхийлнэ .

Зөөгдөх шинж чанарын тогтмолууд

Суурь дамжих программчлалын жишээг үүсгэхэд зориулагдсан нэрийн олонлогийг os модуль экспортолдог. Энэ олонлог нь тухайн компьютерийн системийн зам ба директорын тусгаарлагч тэмдэгт, эцэг болон одоогийн директорын тэмдэглэгээ, мөрийг төгсгөх тэмдэгт зэрэг сууриас хамаарах тохиргоог энэ олонлог агуулна. os.linesep нь Windows-ийн зөвшлийн дагуу \015\012-ийн үсгэн илэрхийлэл болох \r\n –г буцаадаг. Python-ны хуучин хувилбарууд дээр үүнийг 16-т, 8-тын хэлбэрээр үзүүлдэг.

```
>>> os.pathsep, os.sep, os.pardir, os.curdir, os.linesep
(';', '\\', '..', '.', '\r\n')
```

Python-ажиллаж байгаа суурь дээр директорын нэрсийг ялгах тэмдэгт болгон os.sep -ийг ашигладаг. Скриптийн системтэй холбоотой тэмдэгт мөрүүдийг үүсгэх, задлахад эдгээр шинэ чанаруудыг хэрэглэснээр энэ скрипт нь системийн хооронд зөөгдөх бүрэн боломжтой болно. dirpath нь Windows дээр dir\dir, Linux дээр dir/dir Macintosh дээр dir:dir байсан ч, os.sep.split(dirpath) –ийг дуудаж сууриас хамааралтай директорын нэрсийг салган хувааж чадна. Windows нь урагшаа налуу зураасны оронд хойшоо налуу зураасыг хэрэглэж чаддаг ч гэсэн зөөгдөх шинжийн тогтмол утгууд нь сууриас хамааралгүйгээр директорийг боловсруулах боломжийг олгоно.

os.path багажийн үндэс

os.path-модуль нь өөрийн директортой холбоотой олон ширхэг багажтай. Жишээ нь энэ модуль нь файлын төрөл(isdir, isfile, ба бусад), файл байгаа эсэх (exists), файлын нэрийг авч хэмжээг буцаах (getsize) зөөгдөх шинжтэй функцүүдтэй.

```
>>> os.path.isdir(r'C:\temp'),       os.path.isfile(r'C:\temp')
(True, False)
>>> os.path.isdir(r'C:\config.sys'),  os.path.isfile(r'C:\config.sys')
(False, Tuue)
>>> os.path.isdir('nonesuch'),       os.path.isfile('nonesuch')
(False, False)

>>> os.path.exists(r'c:\temp\data.txt')
0
>>> os.path.getsize(r'C:\autoexec.bat')
260
```

os.path.isdir and os.path.isfile ийг дуудаж энгийн файл эсвэл директорын аль нь болохыг мэдэх ба False утгыг буцаавал ийм файл директор байхгүй байна гэж үзнэ. Python-ны ажиллаж байгаа суурьт зохицсон зөвшлийн дагуу директорын нэрийг автоматаар авч ашиглах директорын замыг хуваах нэгтгэх функцүүдийг дуудаж болно.

```
>>> os.path.split(r'C:\temp\data.txt')
('C:\\temp', 'data.txt')
>>> os.path.join(r'C:\temp', 'output.txt')
'C:\\temp\\output.txt'

>>> name = r'C:\temp\data.txt'                    # Windows paths
>>> os.path.basename(name), os.path.dirname(name)
('data.txt', 'C:\\temp')

>>> name = '/home/lutz/temp/data.txt'             # Unix-style paths
>>> os.path.basename(name), os.path.dirname(name)
('data.txt', '/home/lutz/temp')

>>> os.path.splitext(r'C:\PP3rdEd\Examples\PP3E\PyDemos.pyw')
('C:\\PP3rdEd\\Examples\\PP3E\\PyDemos', '.pyw')
```

os.path.split нь директорын замаас файлын нэрийг салгадаг ба скрипт дуудагдаж байгаа компьютерт зохицсон директорын зөвшилд зохицсон зөөгдөх хэлбэрээр os.path.join нь эдгээрийг буцааж нэгтгэдэг. basename ба dirname нь дээрх хуваалтын 1, 2-р элементийг буцаадаг ба splitext нь файлын нэр,файлын өргөтгөл (сүүлийн цэгээс хойших) болгон хуваана. normpath -ийг Unix ба Windows тусгаарлагч холилдон орсон үед ашиглахад эвтэйхэн байдаг.

```
>>> mixed
'C:\\temp\\public/files/index.html'
>>> os.path.normpath(mixed)
'C:\\temp\\public\\files\\index.html'
>>> print os.path.normpath(r'C:\temp\\sub\.\file.ext')
C:\temp\sub\file.ext
```

Энэ модуль нь файлын замын дах директорын бүтэн нэрийг буцаадаг, .. нь эх директор гэх мэтээр одоогийн директороос тооцогдсон abspath гэсэн зөөгдөх функцтэй.

```
>>> os.getcwd( )
'C:\\PP3rdEd\\cdrom\\WindowsExt'
>>> os.path.abspath('temp')                # expand to full pathname
'C:\\PP3rdEd\\cdrom\\WindowsExt\\temp'
>>> os.path.abspath(r'..\examples')        # relative paths expanded
'C:\\PP3rdEd\\examples'
>>> os.path.abspath(r'C:\PP3rdEd\chapters')   # absolute paths unchanged
'C:\\PP3rdEd\\chapters'
>>> os.path.abspath(r'C:\temp\spam.txt')      # ditto for filenames
'C:\\temp\\spam.txt'
>>> os.path.abspath('')                    # empty string means the cwd
'C:\\PP3rdEd\\cdrom\\WindowsExt'
```

Файлын нэр нь бүрэн тодорхойлогдсон биш, одоогийн ажлын директороос харьцангуйгаар тодорхойлогдсон байдаг тул файлыг хадгалахад үнэндээ хэрэглэгддэг директорийг хэрэглэгч харахыг хүсэх үед os.path.abspath функцийг хэрэглэнэ.

On Windows, for example, when GUI-based programs are launched by clicking on file explorer icons and desktop shortcuts, the execution directory of the program is the clicked file's home directory, but that is not always obvious to the person doing the clicking; printing a file's abspath can help.

Скриптээс системийн бүрхүүлийн командыг ажиллуулах

os модуль нь Python-ний скриптийн дотроос системийн бүрхүүлийн программыг ажиллуулах функцүүдтэй. os –модулийн дараах 2 функц нь системийн бүрхүүлийн командын мөрийг ажиллуулах боломжийг олгоно.

os.system

Python-ний скриптээс системийн бүрхүүлийн командыг ажиллуулах

os.popen

Бүрхүүлийн командыг ажиллуулж оролт ба гаралт урсгалд холбох

Системийн бүрхүүлийн команд гэж юу вэ?

Эдгээр функцүүдийг ойлгох хүрээнд эхлээд хэдэн нэр томъёог тодорхойльё. Энэ текст дээрх бүрхүүл гэсэн нэр нь компьютер дээрх командын тэмдэгт мөрийг уншиж ажиллуулдаг систем, бүрхүүлийн команд гэдэг нь компьютерийн системийн бүрхүүлд хэрэглэгч шивж оруулдаг командын тэмдэгт мөр гэсэн утгатай. Windows-дээр MS-DOS консолыг эхлүүлж директорын жагсаалтыг харуулах dir , файлын агуулгыг харуулах type гэх мэт DOS-ийн командыг шивж ажиллуулж болно. Linux дээр бол xterm-цонхыг ажиллуулж директорын жагсаалтыг харуулах ls файлын агуулгыг харуулах cat командыг шивж ажиллуулж болно. Unix дээр маш олон янзын системийн бүрхүүл байдаг ч бүгд командыг уншиж ажиллуулж болно. Dir, type, ls, cat команд нь бүрхүүлийн командууд юм. Дараах жишээнд Windows –ийн MS-DOS-ийн бүрхүүл дээр 2 командыг шивж ажиллуулсан байна.

```
C:\temp>dir /B                ...type a shell command line
about-pp.html                 ...its output shows up here
python1.5.tar.gz              ...DOS is the shell on Windows
about-pp2e.html
about-ppr2e.html
newdir

C:\temp>type helloshell.py
# a Python program
print 'The Meaning of Life'
```

Бүрхүүлийн командыг ажиллуулах

Python скрипт нь өөрийн ажиллаж байгаа системийн бүрхүүлийн янз бүрийн командыг ажиллуулж болох учир командын мөрнөөс ажиллуулж болох Python-дээр бичигдсэн ба бичигдээгүй бүх багажийг дуудан ажиллуулж болно. Дараах Python код нь өмнөх жишээн дээр системийн бүрхүүлд шивж оруулсан 2 командыг дуудан ажиллуулна.

```
C:\temp>python
>>> import os
>>> os.system('dir /B')
about-pp.html
python1.5.tar.gz
about-pp2e.html
about-ppr2e.html
newdir
0

>>> os.system('type helloshell.py')
# a Python program
print 'The Meaning of Life'
0
```

The 0s at the end here are just the return values of the system call itself. The system call can be used to run any command line that we could type at the shell's prompt (here, C:\temp>). The command's output normally shows up in the Python session's or program's standard output stream.

Төгсгөлд нь байгаа 0–үүд нь системийн дуудлагаас буцааж байгаа утга юм. Системийн дуудлагыг системийн бүрхүүлд бичихэд ойлгох дурыг командыг ажиллуулахдаа хэрэглэж болно. Программын гаралт нь Python-ны хэсэг буюу программын гаралтын стандарт урсгал руу гарна.

Бүрхүүлийн командтай мэдээлэл солилцох нь

Скриптээс дуудагдсан командын мөрийн гаралтыг авахыг хүсвэл юу болох вэ? os.system нь бүрхүүлийн командыг мөрийг зүгээр л дууддаг бол os.popen нь дуудахын хамт стандарт оролт гаралттай холбодог. Анхны тохируулгаар командын мөрийн гаралттай холбоотой файл хэлбэрийн обьектыг буцаана. (popen –д w флагийг дамжуулахад гаралтын оронд нь командын мөрийн оролтын урсгал руу холбогдоно.) popen –г ашиглан эхлүүлсэн командын мөрийн гаралтыг унших обьектийг ашигласнаар терминалын цонхон дээр тухайн командын мөрийг эхлүүлсний дараа гарч ирэх мэдээллийг дундаас нь барьж авна.

```
>>> open('helloshell.py').read( )
"# a Python program\nprint 'The Meaning of Life'\n"

>>> text = os.popen('type helloshell.py').read( )
>>> text
"# a Python program\nprint 'The Meaning of Life'\n"

>>> listing = os.popen('dir /B').readlines( )
>>> listing
['about-pp.html\n', 'python1.5.tar.gz\n', 'helloshell.py\n',
'about-pp2e.html\n', 'about-ppr2e.html\n', 'newdir\n']
```

[*] In the next chapter, after we've learned about file iterators, we'll also learn that the popen objects have an iterator that reads one line at a time, often making the readlines method call superfluous.

```
>>> os.system('python helloshell.py')      # run a Python program
The Meaning of Life
0
>>> output = os.popen('python helloshell.py').read( )
>>> output
'The Meaning of Life\n'
```

Энэ бүх жишээн дээр system ба popen-д дамжуулж байгаа командын мөрүүд нь уян хатан биш байгаа ч (+, %,)-зэрэг тэмдэгтүүдийг ашиглан ажиллах үед үүсдэг тэмдэгт мөрийг Python –ны программд ашиглахгүй байх шалтгаан байхгүй. Динамикаар үүсгэх командыг дамжуулснаар system ба popen функцүүд нь Python скриптийг бусад программыг эхлүүлэх, зохион байгуулах боломжтой уян хатан бөгөөд зөөгдөх багаж болгон хувиргадаг.

Бүрхүүлийн командын хязгаарлал

system ба popen –д 2 хязгаарлал байгааг тогтоож авах хэрэгтэй.

Эхнийх нь: Энэ функцүүд нь зөөгдөх шинжтэй ч гэсэн тэдгээрийг ашиглаж байгаа командын мөрүүд нь зөөгдөх шинжтэй байвал л бүрэн зөөгдөх шинжтэй байна.Жишээ нь өмнөх жишээний DOS-ийн dir ба type, Unix-төст системийн ls ба cat юм.

2-рх нь: Python файлыг программ шиг дуудаж ажиллуулах нь программын файлыг импортлох, өөрийн тодорхойлсон функцүүдийг дуудаж ажиллуулахаас их удаан, илт ялгаатай арга юм. os.system ба os.popen- ыг дуудахад тэдгээр нь таны үйлдлийн системд цоо шинэ биеэ даасан программыг ажиллуулдаг. Программыг модуль хэлбэрээр импортолоход Python-ны интерпретатор нь модулийг үүсгэхийн тулд нэг процесст программын эхийг зүгээр л ачаалж ажиллуулна. Өөр ямар ч программ үүний үр дүнг эхлэхгүй.

Python-ны execfile функц нь мөн программын эхийг ажиллуулах боловч түүнийг дуудсан процесстэй 1 процесс болон ажиллана. Энэ талаараа импортлохтой ижил боловч файлын текстийг дуудсан программын execfile функц тохиолдсон газарт хуулбарлан авч ажиллуулдгаараа ялгаатай.(Хэрэв нийтийн ба орчны нэрийн олонлогийг дамжуулаагүй бол) . Импортолохоос ялгаатай нь файлын кодыг энэ процесст олон удаа хэрэглэгдэж байсан ч үг дуугүй уншиж ажиллуулах бөгөөд файлыг ажиллуулсны дараа модулийн обьект үүсэхгүй. Хэрэв заавал биш нэрийн олонлог бүхий толь бичиг нь дамжигдаагүй бол файлын код дотор утга олгосон үйлдэл нь execfile функц байгаа программын хүрээлэл дэх хувьсагчийг дарж болно.

Тусдаа программ үүсгэх дорвитой шалтгаанууд байдаг. Гэвч ихэнх тохиолдолд модулийг импортлох нь программ хөгжүүлэх хурдан бөгөөд илүү шулуун арга зам юм.

Эдгээр функцийг та хэрэг болгон ашиглахаар төлөвлөсөн бол os.system нь дуудагдсан команд хийгдэж дуустал хүлээн бусад үйлдлийг хаадгийг мэдэх хэрэгтэй. Linux, Windows дээр дуудагдсан команд паралелаар ажиллаж эхэлнэ.

os.system("python program.py arg arg &")

os.system("start program.py arg arg")

Python –ны сүүлийн хувилбаруудад нэмэгдсэн os.startfile ийг үнэндээ их ашигладаг. Энэ функц нь хулганын курсороор файлын дүрсийг дарсантай адилаар Windows –регисрт бүртгэсэн программаар тухайн файлыг нээх боломжтой.

```
os.startfile("webpage.html")    # open file in your web browser
os.startfile("document.doc")    # open file in Microsoft Word
os.startfile("myscript.py")     # run file with Python
```

(Тодорхойлолтын дагуу дуудсан программ нь командаас буцаасан файл хэлбэрийн обьектийг уншиж бичих боломжтой) os.popen нь ихэнхдээ дуудсан программыг хаахгүй боловч дамжуулах обьект хаагдсан, дуудагдсан программаас гарах буюу дамжуулах сувгийг дуустал нь уншихаас өмнө хог цугларсан зэргээс болж Windows ба Linux орчинд санамсаргүй хаагдаж болно. Unix-ийн os.fork/exec ба Windows-ийн os.spawnv –ийг дуудсан программ хаахгүй параллелаар ажиллуулахад ашиглана.

os модулийн system ба popen функц нь программ эхлүүлэгч, урсгалын чиглэлийг өөрчлөгч, процесс хоорондын холболтын төхөөрөмж болон ашиглагддаг.

os тодулийн бусад функцүүд

Энэ модулийн бусад функцүүдийг тайлбарлая.

os.environ

Системийн бүрхүүлийн хувьсагчийн утгыг авах ба олгох

os.fork

Unix –ийн шинэ процессыг үүсгэж эхлүүлэх

os.pipe

Программуудыг холбох

os.execlp

Шинэ программыг эхлүүлэх

os.spawnv

Шинэ программыг доод түвшний хяналттай эхлүүлэх

os.open

Opens a low-level descriptor-based file

Доод түвшний хувьсагч дээр тулгуурласан файлгы нээх

os.mkdir

Шинэ директорыг үүсгэх

os.mkfifo

Нэрлэгддсэн шинэ сувгийг үүсгэх

os.stat

Доод түвшний мэдээллийг авах

os.remove

Директорын замыг ашиглан файлыг устгах

os.path.walk, os.walk

Директорийн бүтэн модны бүх хэсгүүдийн дагуу давтах буюу ийм давталттайгаар үйлдлийг гүйцэтгэх

os модуль нь open, read, ба write зэрэг файлыг нээх, унших, бичих функцүүдтэй боловч доод түвшинд файлтай ажиллах энэ функцүүд нь open функцийг ашиглан үүсгэсэн Python-ны файлын обьект болох stdio-ээс бүхэлдээ өөр байна. Ер нь Python-ны энэ open функцийг(os модулийнх биш) файлын илүү онцгой боловсруулалтад ашиглана.

sys ба os модулийн багажийг системийн энгийн түвшинд хэрхэн хэрэглэхийг тайлбарласан ба бусад хэсгийг Python-сангийн гарын авлагаас үзнэ үү.

Параллел системийн багажууд

"Сармагчинд юу хийхийг нь хэлж өгөх "

Ихэнх компьютерүүд олон цагаар сул зогсдог. Процессорын ашиглалтыг үзэх системийн удирдлагын программыг ажиллуулахад энэ үзүүлэлт 100%-д хүрэх нь их ховор болохыг харж болно. Дисктэй харьцах, сүлжээний урсгал,өгөгдлийн сангийн лавлагаа, хэрэглэгч товч дарахад гадны үзэгдэл үүсэхийг хүлээх зэрэг олон хүлээлт программд байдаг. Үнэндээ орчин үеийн процессорын багтаамжийн ихэнх нь сул зогсолтын горимд зарцуулагддаг.Оргил хэрэглээний үед хурдтай процессор үйлдлийг хурдасгах боловч түүний хүчин чадлын ихээхэн нь ашиглагдахгүй байж болно.

Дээр үеийн компьютерийн системд олон программыг зэрэг ажиллуулах замаар процессорын ашиглагдаагүй хүчин чадлыг нөхөн ашигладаг. Процессорыг олон бодлогод хуваан ашигласнаар гадны үзэгдэл үүсэхэд хүлээх үйлдэлд цаг үрэх хэрэгцээгүй болно. Олон бодлогыг нэг зэрэг цагийн хувьд давхцуулан параллелаар гүйцэтгэх учир ер нь энэ техникийг параллел боловсруулалт гэнэ. Энэ нь орчин үеийн үйлдлийн системийн үндэс ба олон идэвхтэй цонх хэрэглэх санааг дэвшүүлэхэд хүргэсэн. 1 программыг ч олон бодлого болгон хувааж параллелаар ажиллуулах нь нийт системийг хурдан болгоно.

Ямар хэмжээний ажил гүйцэтгэхээс үл шалтгаалан хэрэглэгчид орчин үеийн программын систем уриалгахан ханддаг болно гэж найдаж байна. Удаан ажиллах бодлогыг программын бусад хэсэгтэй хамт ажиллуулан хэдийгээр зарим хэсэг нь чөлөөгүй байсан ч тухайн систем уриалгахан (хариу өгөмтгий) байж болно. Үүнээс гадна параллел боловсруулалт нь ийм болон бусад программд угаасаа зохицсон байдаг: Зарим бодлогыг бие даан параллелаар ажиллах бүрэлдэхүүнтэйгээр зохиомжилж программчлахад хялбархан байдаг.

Python-ны процессыг 1 зэрэг ажиллуулах 2 үндсэн арга байдаг: Салаалах ба трийдийг эхлүүлэх. Ажил хэрэг болгоход хоёулаа Python-ны жижиг кодыг параллелаар ажиллуулах үйлдлийн системийн сервис дээр түшиглэдэг. Процепурын хувьд, холболт, интерфэйс, авсаархаг байдлын үүднээс энэ 2 арга нь өөр өөр байдаг. Салаалалтын процесс нь Windows дээрх стандарт Python-д байхгүй боловч Python –ний трийд нь бүх гол суурьт ажиллана. Үүнээс гадна os.popen ба os.system функц, салаалах үйлдэлтэй адилаар завсрын сууриуд дээр программыг эхлүүлэх нэмэлт аргыг os.spawn –бүлэг функц нь хангана.

Процессийг салаалах

Салаалсан процесс нь паралель бодлогын зохион байгуулах уламжлалт арга ба Unix-ийн багажийн олонлогийн гол хэсэг юм. Өөр программаас дуудагдсан эсэхээс үл хамааран биеэ даасан программыг эхлүүлэх шууд арга болно. Салаалах үйлдэл нь программыг хувилах ойлголт дээр тулгуурладаг. Салаалсан программыг дуудахад үйлдлийн систем нь энэ программын шинэ хуулбарыг ойд үүсгэн эхтэй нв параллелаар ажиллуулдаг. Зарим систем нь эх программыг үнэндээ хуулдаггүй боловч хэрэв энэ яг ижил хуулбар байвал шинэ хуулбар ажилладаг.

Салаалах үйлдлийн дараа программын эх хуулбар нь үндсэн процесс гэж нэрлэгддэг ба os.fork –ийн үүсгэсэн хуулбарыг дэд процесс гэж нэрлэдэг. Ер нь үндсэн процесснь хэдэн ч дэд процесстой байж болно. Бүх салаалсан процесс нь үйлдлийн системийн хяналт доор бие даан параллелаар ажиллана. Энэ нь онолоос практикт илүү энгийн бөгөөд Жишээ 5-1-д q үсэг оролтонд шивтэл шинэ дэд процессыг эхлүүлнэ.

Example 5-1. PP3E\System\Processes\fork1.py

```
# forks child processes until you type 'q'

import os

def child( ):
    print 'Hello from child',  os.getpid( )
    os._exit(0)  # else goes back to parent loop

def parent( ):
    while 1:
        newpid = os.fork( )
        if newpid == 0:
            child( )
        else:
            print 'Hello from parent', os.getpid( ), newpid
        if raw_input( ) == 'q': break

parent( )
```

os модульд процессыг салаалах Python-ны багаж байх бөгөөд энэ нь C-гийн сангийн стандарт салаалах функцийн өнгөц нөмрөг юм.Шинэ параллел процессыг эхлүүлэхийн тулд Python-ны функцийг дуудна. Дуудагдаж байгаа программын хуулбарыг үүсгэх учраас хуулбар бүрээс өөр өөр утгуудыг буцаана. Дэд процессоос 0-ийг, үндсэн процесс дотроос шинэ дэд процессийн процессийн дугаарыг буцаана. Дэд процессийн доторх ялгаатай боловсруулалтыг эхлүүлэхдээ ихэнхдээ энэ үр дүнг программ шалгадаг. Жишээ нь : Энэ скриптэд зөвхөн дэд процесс бүрт child функцийг ажиллуулна.

[*]Python –ны одоогийн хувилбарт Python скриптэд os.fork –ийг дуудсанаар Python-ны интерпретаторын процессыг үнэндээ хувилна. Python-ны интерпретатор таны

Python: алдааг боловсруулах

try команд

try команд нь Python-ны exception боловсруулах механизмыг хангана. Энэ нийлмэл команд нь дараах 2 хэлбэрээр бичигдэж болно.

- Try хэсгийг нэг буюу түүнээс дээш except хэсэг дагалдан орж болно.
- try хэсгийг зөвхөн 1 finally хэсэг дагалдана.

Python 2.5-дээр Try команд нь except хэсгүүдтэй (заавал биш else хэсэг) , төгсгөлд нь finally хэсэгтэй байж болно.

try/except

try командын try/except хэлбэрийн өгүүлбэр зүй дараах байдалтай байна.

```
try:
   statement(s)
except [expression [, target]]:
   statement(s)
[else:
   statement(s)]
```

This form of the TRy statement has one or more except clauses, as well as an optional else clause.

except хэсэг бүрийн доторх их бие нь exception боловсруулах хэсэг болно. try хэсгээс илгээсэн exception –ны төрөлтэй except хэсгийн *expression* –дэх төрөл таарч байвал тухайн хэсэг дэх код ажиллана. The optional *target* is an identifier that names a variable that Python binds to the exception object just before the exception handler executes. A handler can also obtain the current exception object by calling the exc_info function of module sys (covered in exc_info on page 168).

Here is an example of the try/except form of the try statement:

```
try: 1/0
except ZeroDivisionError: print "caught divide-by-0 attempt"
```

try/finally

try командын TRy/finally хэлбэрийн өгүүлбэр зүй дараах байдалтай байна.

try:
   statement(s)
finally:
   statement(s)

Энэ хэлбэр нь зөвхөн 1 хэсэгтэй байдаг ба else хэсэгтэй байх ёсгүй.

finally хэсэг нь цэвэрлэх боловсруулалтыг гүйцэтгэдэг. try хэсэг ямар ч байдлаар дууссан байсан энэ код нь ажилладаг . try хэсэгт exception үүсэж илгээгдэж, энэ хэсэг дуусмагц цэвэрлэх боловсруулалт хийгдэх ба exception илгээгдсээр байна. exception үүсэхгүй байсан ч, try хэсэг дууталаа хийгдсэн буюу break, continue, буюу return командаар гарсан эсэхээс үл хамааран энэ боловсруулалт хийгдэнэ.

Файлын Обьект

file нь Python-ны үндсэн төрөл нь ба Python программын өгөгдөл унших бичих энгийн арга болж өгдөг. Файлын обьектийг хэрэглэн ашиглаж буй үйлдлийн системээс нээж болох файлыг уншиж бичих болно. IOError гэсэн exception-ны үндсэн классын жишээг үүсгэх замаар файлын обьекттой холбоотой оролт гаралтын алдаанд хариу үйлдэл үзүүлдэг. Энэ exception-ны шалтгаан болох алдаанууд нь файл үүсгэх , нээх үед гарах нээх үйлдэл амжилтгүй болсон, нэг файлын обьект нь тухайн методыг агуулаагүй үед энэ методыг дуудах, файлын обьектийн методуудыг ажиллуулан олсон оролт гаралтын алдааг багтаана. (Жишээ нь зөвхөн унших файлын обьектийн write методыг, эсвэл файлын байрлал заагчийг нь өөрчлөх боломжгүй файлын seek методыг дуудах).

Open-ийг ашиглан файлын обьектийг үүсгэх

Python-ны файлын обьектийг үүсгэхдээ дараах бичиглэлийн дагуу open гэдэг үндсэн функцийг дуудна.

open(*filename, mode*='r', *bufsize*=-1)

Файлын замыг илэрхийлсэн *filename* тэмдэгт мөрөнд нэр нь байх файлыг open функц нээж file үндсэн төрлийн жишээ болсон Python-ны файлын обьектыг буцаана. file –ийг шууд дуудах нь open функцийг дуудахтай ижил боловч Python-ны ирээдүйн рилийзүүдэд үйлдвэрлэх функц болох open функцийг дуудаарай. *mode* тэмдэгт мөрийг ил дамжуулсан ба *filename*-дэх файл хэдийнээс байхгүй бол open функц энэ файлыг үүсгэнэ. Өөрөөр

хэлбэл, нэрээс үл хамааран функц нь хэдийн байгаа файлыг нээх төдийгүй, шинэ файл үүсгэж чадна.

File mode

mode нь файлыг хэрхэн нээх буюу үүсгэхийг илэрхийлсэн тэмдэгт мөр.

'r'

Файл хэдийн байж байх ёстой. Энэ нь файлыг зөвхөн унших горимд нээнэ.

'w'

Файлыг зөвхөн бичих горимд нээнэ. Файл хэдийн байвал дарж буюу хасагдан хадгалагдана. Файл хэдийн байхгүй бол түүнийг үүсгэнэ.

'a'

Файлыг зөвхөн бичих горимд нээнэ.Хэрэв файл хэдийн байвал, тэр чигээр хадгалагдах ба бичих өгөгдөл энэ файлд нэмэгдэнэ. Файл хэдийн байхгүй бол түүнийг үүсгэнэ. *f*.seek –ийг дуудахад алдаа гарахгүй ч ямар ч нөлөөгүй байна.

'r+'

Файл хэдийн байх ёстой ба бичих унших зорилгоор нээнэ. Бүх методыг дуудах боломжтой.

'w+'

Файлыг бичих унших зорилгоор нээх ба бүх методыг дуудах боломжтой. Файл хэдийн байвал дарж буюу хасагдан хадгалагдана. Файл хэдийн байхгүй бол түүнийг үүсгэнэ.

'a+'

Файлыг бичих унших зорилгоор нээх ба бүх методыг дуудах боломжтой. Хэрэв файл хэдийн байвал, тэр чигээр хадгалагдах ба бичих өгөгдөл энэ файлд нэмэгдэнэ. Файл хэдийн байхгүй бол түүнийг үүсгэнэ. *f*.seek –ийг дуудахад алдаа гарахгүй ч хэрэв *f* обьект дох оролт гаралтын дараагийн үйлдэл нь өгөгдлийг бичин хэвийн ажиллаж, оролт гаралтын дараагийн үйлдэл нь өгөгдлийг уншиж байвал энэ файлын *f*.seek нь ямар ч нөлөөгүй байна.

2-тын ба текст горим

mode тэмдэгт мөр нь b буюу t ийн аль нэг нь байж болно. b нь 2-тын горимыг t текст горимыг илэрхийлнэ. Угийн горим нь t текст горим юм.

Unix-дээр 2-тын ба текст горимын хувьд ялгаа байхгүй. Windows-дээр файлыг текст горимд нээсэн үед os.linesep -ийг уншин тэмдэгт мөр утгыг авахад '\n'-ийг буцаах ба файлыг уншихад '\n' тэмдэгт учирна.Урвуугаар, '\n' тэмдэгтийг бичих бүрт os.linesep-ийн хуулбарыг файл руу бичнэ.

Энэ өргөн дэлгэрсэн зөвшил нь С хэлнээс гарч ирсэн бөгөөд тухайн суурийн мөрийн төгсгөлийн зөвшилд санаа зоволгүй дурын суурьт байгаа файлаас унших, файлд бичих боломжийг олгодог.Гэвч Unix төст сууриас бусад суурийн хувьд файл нь 2-тын эсвэл текст горимын алинд байгааг мэдэж байх ёстой.Мөрийн төгсгөлийг илэрхийлэх \n – тэмдэгтийг ашигласан ч текст горимд нээсэн файлын санах ойд байгаа агуулга дах \n тэмдэгтийг хөрвүүлэх ба файлын систем дэх файлд үнэндээ os.linesep тэмдэгт мөр бичигдэнэ гэдгийг тогтоогоорой.

Python нь бас 'U' унших горимд текст файлыг нээх боломж олгодог олон талт шинэ мөрийг дэмждэг. Үүнийг мөрийг төгсгөл ямар кодтой байгаа мэдэхгүй байгаа үед хэрэглэнэ. Өөр өөр үйлдлийн системтэй компьютерийн дунд текст файлыг ашиглахад энэ горимыг ихэнхдээ хэрэглэдэг. 'U' горим нь '\n', '\r', ба '\r\n' тэмдэгт мөрийн аль ч таарсан мөрийн төгсгөл гэж ойлгож '\n' тэмдэгт болгон хөрвүүлдэг.

Буферлэх

bufsize бүхэл тоо нь файлд шаардагдаж байгаа буферийн хэмжээг илэрхийлнэ. *bufsize* –нь 0-ээс бага байвал үйлдлийн системийн угийн утга хэрэглэгдэнэ.Ер нь энэ угийн утга нь харилцах терминалд зориулсан файлын мөрүүдийг хадгалах буферийн хэмжээ ба бусад файлын хувьд 8,192 байт байна. *bufsize* нь 0 бол файл нь буфергүй байх бөгөөд файлд бичих бүрт файлын буферээс бичигдэн суларна. *bufsize* нь 1 бол файл нь мөрөн буфертэй байх ба файлд \n тэмдэгтийг бичих бүрт файлын буферээс бичигдэн суларна. *bufsize* нь 1-ээс их бол файл нь *bufsize*-ийг тоймолсон утга бүхий хэмжээтэй буфертэй байна.Зарим суурийн хувьд хэдийн нээлттэй байгаа файлын буферийг өөрчилж болох боловч бүх суурьт тохирохоор хийх боломжгүй.

Дараалсан ба дараалсан биш хандалт

f файлын обьект нь угаасаа дэс дараалсан хандалттай. Файлаас унших үед файлд хадгалагдсан тэр дарааллаар байтуудыг уншиж авдаг. Файлд бичих үед таны бичсэн дарааллаар байтуудыг файлд хадгална.

Дэс дараалсан биш хандалтыг сонгосон үед файлын обьект бүр нь одоогийн байрлалаа хадгалдаг.(дараагийн бичих унших үйлдэл өгөгдлийн дамжуулалтаа эхлэх, ашиглаж байгаа файл дах байрлал)

Файлыг нээх үед, эхний байрлал нь файлын эхэнд байна. 'a' буюу 'a+' горимд нээсэн*f* файлын обьектийн *f*.write функцийг дуудахад *f* обьект руу өгөгдөл бичихээс өмнө *f* –ийн файлын байрлалыг дандаа файлын төгсгөлд болгодог. *f* файлын обьект рүү *n* байтыг бичих буюу унших үед *f* –ийн файлын байрлал *n* –ээр нэмэгдэнэ. *f*.tell –ийг дуудан файлын одоогийн байрлалыг лавлан авч болно.

Файлын обьектийн давталт

Текст горимд нээгдсэн *f* файлын обьект нь элементүүд нь файлын мөрүүд байх давтагч болдог. Ийм маягаар, энэ давталт нь:

for *line* in *f*:
файлын мөр бүрийг давтдаг. Буферлэлтийн учир шалтгаанаар давталтыг break шиг командаар дуусгалгүй таслах, *f*.readline-ийн оронд *f*.next( )-ийг хэрэглэсэн үед файлын байрлалд дурын утга хадгалагддаг. *f* обьектийг давтагч болгон ашиглаж байгаад түүний өөр 1 унших модулийг дуудсан хэрэглэхийг хүсвэл *f*.seek –харгалзуулан дуудах замаар файлын байрлалыг зөв тогтоосондоо итгэлтэй байвал зохино. *f* обьектод давталтыг шууд хэрэглэхийн 1 эерэг тал нь өндөр бүтээмж ба учир нь хэдийгээр том файл байсан ч хэт их санах ой хэрэглэхгүйгээр оролт гаралтыг бага хэмжээнд байлгах дотоод буферлэлтийг энэ давталт ашиглана.

Polymorphism

x обьект нь файлтай адил байдаг бол функц буюу программын өөр хэсэн нь *x*-ийг файл шиг хэрэглэж болно. Ийм обьектийг ашигладаг код нь хэвийн нөхцөлд энэ обьектийг параметртээ хүлээж авах буюу энэ обьектийг үр дүн болгон буцаадаг үйлдвэрлэх функцийг дуудаж түүнийг үүсгэж авдаг. Жишээ нь: Хэрэв программын кодноос *x* обьектийн *x*.read( ) методыг параметргүйгээр дууддаг бол бүх *x* обьект нь файл шиг байхын тулд энэ кодод зориулан read метод нь параметргүйгээр дуудагддаг ба тэмдэгт мөрийг буцаадаг байх хэрэгтэй. Файлын олон тоотой методыг хэрэгжүүлэхэд өөр программын кодод *x* обьект хэрэгтэй байж болно. Программын тодорхой кодыг ашиглан обьектод байрлуулах хэрэгцээтэй эдгээр нь уялдаатай байдаг.

Tempfile модуль

tempfile модуль нь таны ашиглаж байгаа суурийн дэмждэг хамгийн аюулгүй арга болох түр зуурын файл, директорын үүсгэнэ. Санах ойд багтахааргүй их хэмжээний өгөгдлийг боловсруулж байгаа эсвэл өөр процессийн дараа ашиглах өгөгдлийг таны программ хадгалах үед түр зуурын файлын хэрэглэх нь гойд шийдэл болно.

Энэ модулийн функцийн параметрийн дараалал нь бага зэрэг төөрөлдөхөөр байдаг. Өөрийн кодыг илүү ойлгомжтой болгохын тулд эдгээр функцийг үргэлж нэрлэсэн параметртэйгээр дуудаж байна уу!

XML-ийг задлах тухай

Программ нь XML баримтыг задлан ашиглахад ямар задлалтыг ашиглах вэ гэдэг нь гол сонголт болдог. Баримтыг дэс дараалуулан уншаад ач холбогдолтой хэсэг бүрийг задалсны дараа программ руу буцаадаг үзэгдлээр жолоодогддог арга, баримтыг нэгмөсөн уншаад санах ойд энэ бартмтыг илэрхийлсэн өгөгдлийн бүтцийг байгуулдаг обьектод тулгуурласан арга бий. SAX нь XML баримтыг задлалтын үзэгдлээр жолоодогдох үндсэн арга нь бөгөөд DOM нь обьектод тулгуурласан гол арга юм. Дээр 2 хэлбэрийн аль алинд нь хувилбарууд бий. Үзэгдлээр жолоодогдох аргын хувьд expat, обьектод тулгуурласан аргын хувьд ElementTree гэх мэт арга бий.

Өөр нэг сонирхолтой боломж нь pulldom –д байгаа сугалалт дээр тулгуурласан задлалт ба C-дээр бичигдсэн бүхий ElementTree-ийн iterparse функц юм.

Another interesting possibility is pull-based parsing, supported by pulldom, covered later in this chapter (and also, to some extent, by ElementTree, via the iterparse function of C-coded module cElementTree).

Үзэгдлээр жолоодуулах задлалт нь ялангуяа маш том баримтыг задлахад тохируулан хийгдсэн тул бага хэмжээний нөөцийг ашигладаг. Хэдий тийм боловч үзэгдлээр жолоодуулах задлалт нь задлах программын дуудагддаг метод доторх боловсруулалтын дагуу өөрийн хэрэглээг төлөвлөх шаардлагатай. Обьектод тулгуурласан аргад шаардагдах илүүдэл нөөцийг олгох шиг маш ярвигтай боловсруулалтыг гүйцэтгэх үед хэрэглэхэд тохиромжтой тийм хэрэглээг илүү уян хатан төлөвлөх боломжийг олгоно.

Эхлээд үзэгдлээр жолоодогддог задлалтыг ашиглаж үзэхийг санал болгож байгаа бөгөөд энэ задлалтыг ашиглах нь өөрийн программдаа ашиглах нь ухаалаг шууд арга юм. Үзэгдлээр жолоодогддог задлалтыг нь илүү өргөсгөх боломжтой: Таны программ үзэгдлээр жолоодогддог задлалтыг гүйцэтгэдэг бол жижиг хэмжээтэй баримтаас илүү том хэмжээний баримт боловсруулахад хэрэглэхэд тохиромжтой. Үзэгдлээр жолоодогддог задлалт нь дэндүү хязгаарлагдмал байвал сугалалт дээр тулгуурсан задлалт pulldom (буюу cElementTree.iterparse) –ыг хэрэглэж үзнэ үү. Хэтэрсэн зөрүү байхгүй программыг гүйцэтгэх зам нь зөвхөн DOM гэж үзэж байвал та DOM-ийг хэрэглэнэ үү! Хамгийн бага санах ой хэрэглэдэг, хурдан Python хэлбэрийн программ болох ElementTree-г хэрэглэх боломжгүй бол DOM нь хамгийн сайн арга бөгөөд энэ аргыг хэрэглэснээр том хэмжээтэй баримтынболовсруулалтад хэрэглэх санах ой, цаг зарцуулалтыг хязгаарлаж үр дүнд хүрнэ.(DOM нь Document Object Model буюу Баримтын Обьектийн Загвар буюу XML, HTML зэрэг форматыг илэрхийлэх хэлнээс үл хамаарах стандарт загвар юм.)

XML –ийг SAX-аар задлах

Ихэнх тохиолдолд XML баримтаас мэдээллийг авах хамгийн сайн арга нь XML-ийн энгийн API болох үзэгдлээр жолоодогдох задлагч SAX-ыг ашиглан задлах явдал юм. SAX

нь олон өөр өөр үндсэн задлах хэрэгслэл дээр хийгдсэн стандарт программын холбоос (API) юм. SAX –ийн арга нь олонхи HTML задлагчтай адилхан шинж ихтэй. Задлагч нь XML элемент, текст агуулга болон бусад оролтын урсгалын чухал үзэгдлүүдтэй учирвал задлагч таны классын методыг буцааж дуудна. Харгалзах үзэгдэл үзэгдэхэд таны үүсгэсэн классын метод дуудагдах арга дээр тулгуурласан үзэгдлээр жолоодогдох задлалт нь олон талт график интерфэйс ба Twisted шиг зарим хамгийн сайн сүлжээний тулгуур программтай ижил шинжтэй байдаг. Янз бүрийн программчлалын талбар дах үзэгдлээр жолоодогдох арга нь эхлэн суралцагчдад жинхэнэ ёсоор үзэгдэхгүй байж болох боловч их ачааллын үед зохицуулдаг өндөр бүтээмж ба дээд зэргийн өргөсгөх боломжийг олгодог.

SAX-ийг ашиглахын тулд боловсруулах классын агуулгыг тодорхойлж, сангийн классын дэд классыг үүсгэж зарим методыг дарж өөрчилдөг. Тэгээд *p* задлагч обьектыг байгуулж өөрийн классын жишээг *p* –гийн боловсруулагч шиг суулгах ба *p* –гийн оролтын урсгалыг *p* –гийн боловсруулах метод руу задлахын тулд өгнө. Энэ боловсруулах метод нь баримтын бүтэц ба агуулгад нөлөөлөх ба хэрэглэгээнд зохицсон боловсруулалтыг гүйцэтгэнэ. xml.sax багц нь *p*-ийг байгуулах үйлдвэрлэх функц ба ерийн тохиолдлуудын энгийн үйлдэлд зохимжтой функцүүдтэй. Энэ багц нь бас оролтын болон бусад алдааны үед үүсэх exception-ны класстай байна.

p задлагчид агуулгыг боловсруулагчаас өөр бусад төрлийн боловсруулагчийг сайн дураар бүртгэж болно. Жишээлбэл, задлах үйлдлийг хийх үед төрөл бүрийн алдааг оношлох үүднээс exception-ийг боловсруулах энгийн аргаас өөр алдаа оношлох стратегийг ашиглахад зориулагдсан өөрийн гэсэн алдаа боловсруулагчтай байж болно. XML баримт дахь Баримтын Төрлийн Тодорхойлолт (DTD)-оос задраагүй элемент ба тэмдэглэлийн тухай мэдээллийг хүлээн авдаг өөрийн гэсэн DTD боловсруулагчтай байж болно. Дэвшилтэт, хүссэнээр өөрчилсөн аргаар гаднах элемэнтийг заасан заагчийг боловсруулдаг өөрийн гэсэн элемэнт боловсруулагчтай байж болно. Энэ дэвшилтэт боломж нь ховорхон ашиглагддаг.

SAX (Simple API for XML)

xml.sax багц

xml.sax багц нь exception-ны нарийн боловсруулалтыг хангадаг SAXException exception –ны класс ба түүний дэд классыг агуулна.

Шинж чанарууд

Методын сүүлийн параметр болох startElement ба startElementNS нь *attr* гэдэг шинж чанарын обьект буюу шинж чанарын нэр ба шинж чанарын утгын зөвхөн уншигдах харгалзаа юм. startElement методын хувьд нэрс нь (*uri*,*localname*) гэсэн хос байх ба үүнд *uri* нь нэрийн олонлогийн URI буюу None , *localname* нь хаяглах тэмдгийн нэр байна. Харгалзааны зарим метод гадна, *attr* нь шинж чанар бүрийн *qname*(боловсорсон нэр) –тэй харьцах методуудтай байна.

Өсгөх задлалт

Унших гэж нээсэн файл төст обьект эсвэл тэмдэгт мөр хэлбэрээр XML баримтын параметр болгон хүлээн авдаг parse методыг бүх задлагч агуулдаг. XML баримтын төгсгөл хүртэл боловсруулж дуусаагүй бол parse нь буцахгүй. Хэдийгээр бүгд биш ч ихэнх SAX задлагч нь өсгөх задлалтыг ашигладаг бөгөөд энэ задлалтын үед баримт нь сүлжээгээр буюу өөр эх үүсвэрээс ирж байгаа юм шиг XML баримтыг бага багаар задлагч руу дамжуулдаг. Сайн өсгөх задлагч нь боловсруулах классын бүх боломжит дуудлагыг боломжийн хэрээр хурдан хийх ба ингэснээр боловсруулалтыг эхлэхээс өмнө баримтыг бүтнээр нь унштал хүлээх шаардлагагүй болдог. Өсгөх аргаар задлах *p* задлагч нь 3 методтой байна. close, feed, reset

xml.sax.saxutils модуль

xml.sax багцын saxutils модуль нь оролтын XML баримт дээр тулгуурлан XML гаралтыг үүсгэх гарын доорх арга болдог 1 класс, 2 методтой байна.

xml.sax –аар XHTML-ийг задлах

Дараах жишээ нь xml.sax-ийг ерийн XHTML-тэй холбоотой бодлогыг биелүүлэхэд ашигласныг харуулсан. Энэ жишээ нь urllib-ээр XHTML хуудсыг авч задлаад энэ хуудаснаас өөр хуудас руу заасан бүх цор ганц холбоосуудыг гаргана. Энэ жишээ нь өгөгдсөн сайтын холбоосыг шалгахын тулд urlparse-ийг ашиглах ба 'http'-ийн илэрхий бүдүүвч бүхий URL-үүдтэй холбоосыг л зөвхөн гаргана.

```
import xml.sax, urllib, urlparse

class LinksHandler(xml.sax.ContentHandler):
    def startDocument(self):
        self.seen = set( )
    def startElement(self, tag, attributes):
        if tag != 'a': return
        value = attributes.get('href')
        if value is not None and value not in self.seen:
            self.seen.add(value)
            pieces = urlparse.urlparse(value)
            if pieces[0] != 'http': return
            print urlparse.urlunparse(pieces)

p = xml.sax.make_parser( )
p.setContentHandler(LinksHandler( ))
f = urllib.urlopen('http://www.w3.org/MarkUp/')
BUFSIZE = 8192

while True:
    data = f.read(BUFSIZE)
    if not data: break
    p.feed(data)
```

p.close( )

XML –ийг DOM-оор задлах

SAX задлалт нь XML баримтыг илэрхийлэх бүтцийг санах ойд байгуулдаггүй. SAX-ийг хурдан бөгөөд дээд зэргээр өргөсгөх боломжтой болгоно.Энэ нь шаардагдаж буй санах ой дох бүтцийн хэмжээнд тохирохоор таны программыг бага хэмжээтэй байх боломжийг олгоно. Хэдий тийм боловч ялангуяа боломжийн жижиг XML баримтыг уншин авч түвэгтэй боловсруулалтыг хийхэд хувьд бүтэн XML баримтыг илэрхийлэх санах ой дох бүтцийг байгуулдаг санг ашиглахыг эрхэмлэдэг. XML-ийн стандарт нь XML-ийн DOM-ыг(Баримтын Обьектийн Загвар) тайлбарладаг. DOM обьект нь XML баримтыг үндэс нь баримтын обьект, зангилаанууд нь модны элемэнтүүд нь байх мод болгон дүрсэлнэ. ElementTree модуль нь XML баримтыг илэрхийлэх, илүү Python хэлбэрийн хурдан арга юм.

Python-ны стандарт сан нь XML DOM стандартын хамгийн бага хэрэгжүүлэлт болох xml.dom.minidom-той байна. minidom нь XML задлалтын DOM аргын ердийн давуу ба сул талууд бүхий бүхий л зүйлсийг санах ойд байгуулдаг. Python-ны стандарт сан нь бас өөр 1 DOM-төст арга болох xml.dom.pulldom модулийг агуулдаг. pulldom нь SAX ба DOM-ийн засварын сонирхолтой арга бөгөөд буцаан дуудах кодыг бичих хэрэгцээгүй Python-ны давтагч шиг задалж буй үзэгдлүүдийг илэрхийлнэ. Гэвч энэ элемэнт бүр нь сонирхолтой эсэхийг үзэх замаар үзэгдлүүдийг давталт хэлбэрээр авч шалгана. Таны программд хэрэгцээтэй үзэгдлүүдийг олохын тулд тухайн үзэгдэл дээр үндэс нь байрлах DOM дэд модыг байгуулахыг expandNode методыг дуудах замаар pulldom-оос хүснэ. Ингээд –д minidom-д байгаа дэд модыг ашиглана. –ийн зохиогч, XML ба Python-ны мэргэжилтэн Paul Prescod-нь “SAX –ийн бүтээмжийн 80%, DOM-ийн ая тухын 80%” гэж тодорхойлсон.Өөр DOM задлагч нь PyXML ба 4Suite өргөтгөх багц юм.

Хэрэглэгчийн график интерфэйс

Tkinter энгийн график интерфэйстэй программыг хялбархан зохиох боломжтой болгодог. Tkinter –ийг импортолж цонхны элементүүдийг байрлуулан тэдгээрийг Tkinter үндсэн давталт руу оруулна. Ингэхэд программ нь үзэгдлээр жолоодогддог болох ба хэрэглэгч цонхны элементүүдтэй харилцахад үзэгдэл үүсэж энэ программ үзэгдлийг боловсруулахад зориулан бичсэн функцүүдээрээ дамжуулан хариулах болно.

Дараах жишээ нь энэ ерөнхий бүтцийг үзүүлэх энгийн программыг толилуулж байна.

import sys, Tkinter Tkinter.Label(text="Welcome!").pack( )

```
Tkinter.Button(text="Exit", command=sys.exit).pack( )
Tkinter.mainloop( )
```

Label ба Button –ийг дуудсанаар харгалзах цонхны элементийг үүсгэх ба энэ элементүүдийг үр дүн болгон буцаана. Эцэг цонхыг тодорхойлохгүй бол нь цонхны элементийг үндсэн цонхон дээр тавина. Нэр бүхий параметрүүд нь цонхны элемент бүрийг тохируулахад хэрэглэгдэнэ. Энэ энгийн жишээн дээр цонхны элементийг ямар 1 хувьсагчид холбох шаардлагагүй байсан. Цонхны элемент бүрийн pack –методыг дуудсанаар цонхны элементийн байршлыг гар удирдлагаас баглагч гэдэг байршлын менежерийн удирдлагад шилжүүлнэ. Байршлын менежер нь геометрийн схемийг боловсруулан цонхны элементийг өөр элемент дотор байрлуулахад хэрэглэгддэг үл үзэгдэгч бүрэлдэхүүн юм. Өмнөх жишээ дээр баглагчийн үйлдлийг удирдах параметрийг дамжуулаагүй тул багцлагч угийн тохируулгаар ажиллана.

Хэрэглэгч дэлгэцийн товчийг дарахад дуудагдах command функц ямар 1 параметргүй ажиллана.Энэ жишээн дээр sys.exit функцийг command параметрт олгон илгээсэн тул хэрэглэгч энэ товчийг дарахад программаас гарна.

Цонхны элементийг үүсгэх байрлуулсны дараа Tkinter's mainloop –ийг дуудаж ажиллуулснаар энэ нь үзэгдлээр жолоодогддог болно.

Харилцах цонх

Tkinter нь харилцах цонхыг тодорхойлох хэд хэдэн нэмэлт модультай. Эдгээр нь идэвхжүүлсэн үед нэмэгдэн гарч ирдэг, хүссэн эрэг хариу болон Cancel зэрэг товчийг дарж өгөх татгалзсан хариултын аль алины дараа программд удирдлагыг буцаан шилжүүлдэг цонх юм.

tkMessageBox модуль

tkMessageBox нь Tk-ийн мессежийн цонхыг ашигладаг класс, функц, тогмолуудтай. Дэлгэц дээр үзүүлэх товчууд болон ямар гарчигтай, ямар мессежтэй, ямар дүрстэй байхыг тодорхойлж болно.

17.1.1.2. tkSimpleDialog модуль

tkSimpleDialog нь өөрийн хүссэн харилцах цонхыг үүсгэхэд шаардлагатай суурь класс юм. *title* ба *prompt* гэсэн сайн дурын параметртэй 3 функцтэй байдаг ба хэрэглэгчид зөвшөөрсөн ба татгалзсан 2 хариу өгнө:

askfloat

Asks the user to enter a floating-point number; returns a float

askinteger

Asks the user to enter an integer number; returns an int

askstring

Asks the user to enter a string; returns a plain str (when the user has entered only ASCII characters) or a unicode string (in all other cases)

17.1.1.3. The tkFileDialog module

tkFileDialog нь хэрэглэгч унших буюу хадгалахад зориулан файл буюу директорыг сонгоход хэрэгтэй класс ба функцүүдтэй байдаг. Энэ класс ба функцүүд нь анхны файлын өргөтгөл тодорхойлох *defaultextension* , ямар өргөтгөлийг хүлээн авахыг тодорхойлох *filetypes* ,файлууд буй анх үзүүлэх директорыг тодорхойлох *initialdir* зэрэг маш олон боломжуудтай.

askdirectory

Директорын замыг буцаана.

askopenfilename

Нээж ашиглах хэдийн үүссэн файлын замыг буцаана

askopenfilenames

Нээж ашиглах хэдийн үүссэн нэг буюу хэдэн файлын замыг буцаана

asksaveasfilename

Хадгалах файлын нэрийг буцаана. (Хэдийн байгаа бол лавлан асууна)

The tkColorChooser модуль

tkColorChooser нь параметергүй дуудахад одоо сонгосон байгаа цонхны өнгийг буцаана.Үүнд улаан, ногоон, хөх бүрэлдэхүүн ба Tkinter-ийн өнгөний тэмдэгт мөр багтана.

Цонхны агуулах элементүүд

Tkinter модуль нь өөр цонхны элементийг агуулахад зориулагдсан цонхны элементүүдтэй. Frame нь зөвхөн ийм л үүрэгтэй. Программын үндсэн цонх болох Tkinter –ийн үндсэн цонхыг багтаасан Toplevel нь дээд түвшний цонх бөгөөд цонхны менежер нь түүнтэй харилцдаг.

Хүрээ

Frame класс нь бусад хүрээ болон дээд түвшний цонхыг агуулсан дэлгэцийн тэгш өнцөгт хэлбэртэй хэсэг юм. Frame нь бусад цонхны элементийг агуулах үүрэгтэй. borderwidth – сонголтын эхний утга нь 0 байдаг нь хүрээ ирмэггүй байна гэсэн үг. 1 гэсэн утга олговол сая ирмэг нь харагдаж эхэлнэ.

17.4.2. Toplevel

Toplevel класс нь дээд түвшний цонх гэж нэрлэгддэг дэлгэцийн тэгш өнцөгт хэлбэртэй хэсэг бөгөөд чимэглэх өөрчлөлтүүдийг цонхны менежер боловсруулж дэлгэцнээ харуулна. Toplevel-ийн жишээ бүр нь цонхны менежертэй харилцдаг ба өөр цонхны элементийг агуулна. Tkinter программ бүр нь үндсэн цонх гэж нэрлэгдэх ядаж 1 дээд түвшний цонхтой байдаг. Tkinter-ийн үндсэн цонхыг *root*=Tkinter.Tk( )- ээр эхлүүлдэг. Программ эхлүүлэхгүй бол эхний хэрэгцээ гармагц үндсэн цонхыг ажиллуулна. Дахин 1өөр дээд түвшний цонхыг ажиллуулахыг хүсвэл *another_toplevel*=Tkinter.Toplevel( )- ийг дуудан ажиллуулж болно.

Байршлыг удирдах систем

Өмнөх жишээнээс харахад pack –ийг ашиглан бүх цонхны элемэнтүүдийг дэлгэцнээ харуулсан. Tkinter-г бодит амьдралд ашиглах жирийн арга юм.Хэдий тийм боловч байршлыг удирдах нөгөө 2 системийг заримдаа ашигладаг. Энэ хэсэгт Tkinter –ийн байршлыг удирдах бүх 3 системийг тайлбарласан. Багцлагч, тор хэлбэрт байрлуулагч, байрлуулагч гэсэн 3 системтэй.Цонхны элемэнтийн 1 агуулагчид байршлыг удирдах системийг холин хэрэглэж болохгүй: Тухайн агуулах элемэнтийн бүх дэд элемэнтүүдийг байршлыг удирдах 1 системийг хэрэглэн байрлуулсан байх ёстой. Эсрэг тохиолдолд маш сонин үр дүнд хүрнэ.(Tkinter нь төгсгөлгүй давталтад орох гэх мэт)

Багцлагч

Цонхны элемэнтийн *pack* методыг дуудаж элемэнтүүдийг байршуулах удирдлагыг байршлын энгийн, уян хатан Packer системд шилжүүлнэ. Packer нь цонхны элемэнтүүдэд

хэрэгцээтэй талбайн дагуу (padx ба pady –ийг оролцуулан) үндсэн агуулагчид тэдгээрийг байрлуулж, хэмжээг нь тохируулна.

Тор хэлбэрт байрлуулагч

Цонхны элемэнтийн grid методыг дуудаж элемэнтүүдийг байршуулах удирдлагыг байршлын тусгай зориулалттай системд шилжүүлнэ. Gridder нь үндсэн агуулагч элемэнтийн доторх хүснэгтийн (тор) нүд бүрт цонхны элементүүдийг тэдгээрийг байрлуулж, хэмжээг нь тохируулна.

Байрлуулагч

Цонхны элемэнтийн place методыг дуудаж элемэнтүүдийг байршуулах удирдлагыг байршлын Placer системд шилжүүлнэ. Placer нь *w* цонхны элемент бүрийг түүний шууд хэрэгцээт хэмжээгээр үндсэн агуулагчид байрлуулж хэмжээг тохируулна. Нөгөө 2 байршлыг удирдах системийн илүү их ашигладаг ба өөрийн гэсэн байршлыг удирдах системийг хэрэгжүүлэхэд Placer тусална.

Tkinter-ийн үзэгдлүүд

Өмнөх жишээнээс зөвхөн 1 төрлийн үзэгдлийн боловсруулалтыг үзсэн. Энэ нь цэсний элемэнт, товчны command= сонголтод дуудагдах функцийг тохируулснаар боловсруулах функцийг дуудах боломжийг бүрдүүлнэ. Tkinter –ийг ашиглан янз бүрийн үзэгдлийг боловсруулах дуудагдах функцийг тохируулах боломжтой. Tkinter -аар хэрэглэгчийн үзэгдлийг үүсгэх боломжгүй. Зөвхөн Tkinter –ийн өөрийнх нь үзэгдлүүдийг ашиглах хүрээнд хязгаарлагдана.

Event Object

Үзэгдлийн функц нь Tkinter–ийн үзэгдлийн обьект *event* гэсэн 1 параметрийг авдаг. Энэ обьект нь дараах шинж чанаруудтай.

char

Гарын товчны кодыг илэрхийлэх 1 тэмдэгт (зөвхөн гарын үзэгдлийн хувьд)

keysym

Гарын товчны бэлэгдлийн нэрийг илэрхийлэх тэмдэгт мөр(зөвхөн гарын үзэгдлийн хувьд)

num

Хулганын товчны дугаар(зөвхөн хулганын үзэгдлийн хувьд): 1 ба up

x, y

Цонхны элемэнтийн зүүн дээд булантай харьцангуй хулганын заагчийн байрлал(цэгээр)

x_root y_root

Дэлгэцийн зүүн дээд булантай харьцангуй хулганын заагчийн байрлал(цэгээр)

widget

Энэ үзэгдэл үзэгдсэн цонхны элемэнт

Боловсруулах функцийг үзэгдэлтэй холбох

Цонхны *w* элемэнтийн үзэгдэлтэй боловсруулах функцийг холбохын тулд ихэнхдээ гурвалжин хаалтанд бичсэн тэмдэгт мөрөөр үзэгдлийг тайлбарлан *w*.bind –функцийг дуудна. Дараах жишээ нь хэрэглэгч Enter товчийг дарах бүрт 'Hello World'-ийг хэвлэн үзүүлнэ.

```
from Tkinter import *

root = Tk( )
def greet(*ignore): print 'Hello World'
root.bind('<Return>', greet)
root.mainloop( )
```

Canvas ба Text классын tag_bind метод нь a Canvas –ийн жишээний олон элемэнт буюу Text-жишээн дэх хүрээний үзэгдэл боловсруулах функцтэй холбогддог.

Event Names

Гурвалжин хаалтанд бичигдэх бараг бүх нэр нь үзэгдлийн энгийн нэр юм.

Гарын үзэгдлүүд

Key

Хэрэглэгч дурын товч дарсан үед үүснэ. Үзэгдлийн обьектийн char шинж чанар нь энгийн товчны алиныг нь дарсныг хэлж өгнө. Үсэг, тоо, цэг тэмдэглэл, тусгай товчны нэр зэрэг тэмдэгтийн хувьд keysym шинж чанар нь char-тай ижил.

Special keys

Тусгай товчууд нь үзэгдлийн нэртэй: F1, F2, -аас F12 нь үүргийн товч, Left, Right, Up, ба Down нь сум төрлийн, page-up ба page-down; BackSpace, Delete, End, Home, Insert, Print, ба Tab нь нэр шигээ тэмдэглэгдсэн товч, Escape нь Esc ; Return нь Enter гэж тэмдэглэгдсэн товч, Caps_Lock, Num_Lock, ба Scroll_Lock нь хорихыг (lock)

хүсэх товч, Alt_L, Control_L, Shift_L нь Alt, Ctrl, Shift товчны бусад товчтой дарсан хувилбарын нэр юм. Бараг бусад бүх үзэгдлийн нэртэй адил эдгээр бүх үзэгдлийн нэрийг гурвалжин хаалтанд бичнэ.

Normal keys

Энгийн товч нь хаалт хэрэглээгүй гурвалжин хаалтаар хаших шаардлагагүй үзэгдлийн нэртэй байна.Энгийн товчны үзэгдлийн нэр нь тэр товчинд харгалзах 'w', '1', '+' гэх мэт тэмдэгт байна.

'Alt-', 'Shift-', ба 'Control-' гэсэн угтварыг авч товчны нэрүүд өөрчлөгдөж болно. Энэ тохиолдолд, үзэгдлийн нэр нь '<...>' хаалтаар хашигдана.Жишээ нь: '<Control-Q>' ба '<Alt-Up>'

Хулганы үзэгдлүүд

Button-1 Button-2 Button-3

Хэрэглэгч хулганын зүүн, дунд, баруун товчийг дарсан үед үүснэ. 2 товчтой хулгана нь дунд талын товч байхгүй учраас Button-1 ба Button-3-ны үзэгдлийг үүсгэнэ.

B1-Motion B2-Motion B3-Motion

Хулганын зүүн, дунд, баруун товчийг дарж байгаад хулганыг хөдөлгөсөн үед үүснэ. (товч даралгүйгээр хулганыг хөдөлгөх үед Enter ба Leave үзэгдэл л үүсдэг.)

ButtonRelease-1 ButtonRelease-2 ButtonRelease-3

Хэрэглэгч зүүн, дунд, баруун товчноос хуруугаа салгасан үед үүснэ.

Double-Button-1 Double-Button-2 Double-Button-3

Хэрэглэгч хулганын зүүн, дунд, баруун товчийг давхар товшсон үед үүснэ. (Энэ үйлдэл нь бас давхар товших үзэгдлийн өмнө Button-1, Button-2, or Button-3-ийг үүсгэнэ.)

Enter

Цонхны элемэнт дээр хулганын заагч ортол хулганыг хөдөлгөх үед үүснэ.

Leave

Цонхны элемэнтээс хулганын заагч гартал хулганыг хөдөлгөх үед үүснэ.

Үзэгдэлтэй холбоотой методууд

Цонхны элемент бүр нь дараах үзэгдэлтэй холбоотой методуудыг агуулна.

Bind	*w*.bind(*event_name,callable*[,'+']) *w*.bind(*event_name,callable*) нь *w*-элементийн *event_name* –үзэгдлийн боловсруулах функц болгон *callable* –ийг тогтооно. *w*.bind(*event_name,callable*,'+') нь *w*-элементийн *event_name* –үзэгдлийн боловсруулах функцийн нэрэнд *callable* –ийг нэмнэ.
bind_all	*w*.bind_all(*event_name,callable*[,'+']) *w*.bind_all(*event_name,callable*) нь цонхны бүх элементийн *event_name* –үзэгдлийн боловсруулах функц болгон *callable* –ийг тогтооно. *w*.bind_all(*event_name,callable*,'+') нь цонхны бүх элементийн *event_name* –үзэгдлийн боловсруулах функцийн нэрэнд *callable* –ийг нэмнэ.
unbind	*w*.unbind(*event_name*) *w*-элементийн *event_name* –үзэгдлийн боловсруулах бүх функцийг устгана.
unbind_all	*w*.unbind_all(*event_name*) bind_all -оор тогтоосон цонхны элементүүдийн *event_name* –үзэгдлийн боловсруулах бүх функцийг устгана.

Үзэгдлийн жишээ

Дараах жишээ нь bind_all –оор гар ба хулганын үзэгдлүүдийг илрүүлнэ.

```
import Tkinter from Tkinter import *

root = Tk( )
prompt='Click any button, or press a key'
L = Label(root, text=prompt, width=len(prompt))
L.pack( )
```

```
def key(event):
    if event.char==event.keysym:
        msg ='Normal Key %r' % event.char
    elif len(event.char)==1:
        msg ='Punctuation Key %r (%r)' % (event.keysym, event.char)
    else:
        msg ='Special Key %r' % event.keysym
    L.config(text=msg)
L.bind_all('<Key>', key)

def do_mouse(eventname):
    def mouse_binding(event):
        L.config(text='Mouse event %s' % eventname)
    L.bind_all('<%s>'%eventname, mouse_binding)
for i in range(1,4):
    do_mouse('Button-%s'%i)
    do_mouse('ButtonRelease-%s'%i)
    do_mouse('Double-Button-%s'%i)

root.mainloop( )
```

Боловсруулах функцтэй холбоотой бусад методууд

w цонхны элемент бүр нь боловсруулах функцтэй холбоотой дараах методыг агуулна.

After	*w*.after(*ms,callable,*args*) *callable*(**args*)- ийг дуудаж одооноос миллсекундээр тоологдох тоолуурыг эхлүүлнэ. Тоолуурыг зогсоохын тулд after_cancel –д дамжуулж болох ID –ийг буцаана. The timer is one-shot: to call a function periodically, the function itself must call after to install itself again.
after_cancel	*w*.after_cancel(*id*)

After	*w*.after(*ms,callable,*args*) *callable*(**args*)- ийг дуудаж одооноос миллсекундээр тоологдох тоолуурыг эхлүүлнэ. Тоолуурыг зогсоохын тулд after_cancel –д дамжуулж болох ID –ийг буцаана. The timer is one-shot: to call a function periodically, the function itself must call after to install itself again.
	id -ийг ашиглан тоолуурыг зогсооно
after_idle	*w*.after_idle(*callable,*args*) Үзэгдлийн давталт хоосон болсон үед боловсруулах функцийг *callable*(**args*)-д гүйцэтгэхийн тулд бүртгэнэ.

Дараах жишээн дээр энгийн электрон цагийг хэрэгжүүлэхэд after методыг ашиглахыг үзүүлнэ.

```
import Tkinter import time

curtime = ''
clock = Tkinter.Label( )
clock.pack( )

def tick( ):
   global curtime
   newtime = time.strftime('%H:%M:%S')
   if newtime != curtime:
      curtime = newtime
      clock.config(text=curtime)
   clock.after(200, tick)
tick( )
clock.mainloop( )
```

after метод нь шийдвэрлэх ач холбогдолтой. Цонхны олон элемент нь өөр дээр нь хийгдэх хэрэглэгчийн үйлдлүүдийг мэдэх, ийм үйлдлийг мөрдөх боловсруулах функцгүй, үүнийг мэдрэх нь сайн дурын байдаг. Дараах жишээн дээр бодит цаг хугацаан дах Listbox-ийн элементийн сонголтыг мөрдөхийн тулд after методоор хэрхэн асуухыг үзүүлсэн.

```
import Tkinter

F1 = Tkinter.Frame( )
```

```
s = Tkinter.Scrollbar(F1)
L = Tkinter.Listbox(F1)
s.pack(side=Tkinter.RIGHT, fill=Tkinter.Y)
L.pack(side=Tkinter.LEFT, fill=Tkinter.Y, yscrollcommand=s.set)
s['command'] = L.yview for i in range(30): L.insert(Tkinter.END, str(i))
F1.pack(side=Tkinter.TOP)

F2 = Tkinter.Frame( )
lab = Tkinter.Label(F2)
def poll( ):
   lab.after(200, poll)
   sel = L.curselection( )
   lab.config(text=str(sel))
lab.pack( )
F2.pack(side=Tkinter.TOP)

poll( )
Tkinter.mainloop( )
```

Клиент талын сүлжээний протоколын модулиуд

Программ нь интернэтэд клиент хэлбэрээр ажиллах буюу сервер хэлбэрээр ажиллах боломжтой. Энэ 2 төрлийн программ нь хэрхэн хандах, өгөгдөл солилцох, өгөгдлийн формат зэрэг протоколын холбогдолтой асуудлуудтай тулгардаг. Дараалал ба илэрхий байдлын хувьд Python –ны сан нь хэд хэдэн өөр модулийг хэрэглэдэг.

URL хандалт

A URL нь интернэт дэх нөөцийг ялган танихад хэрэглэгддэг.

urlparse модуль

urlparse модуль нь URL мөрийг задлан шинжлэхэд хэрэглэгдэнэ. urlparse модулийн өргөн хэрэглэгддэг функц нь urljoin, urlsplit ба urlunsplit юм.

.

urllib модуль

urllib модуль нь URL –ээс өгөгдлийн унших энгийн функцүүдтэй байдаг. urllib нь http, https, ftp, gopher ба файл гэсэн протоколуудыг дэмждэг. Файл гэдэг нь тухайн системийн файл юм.

urllib2 мадуль

urllib2 нь- urllib модулийн баян, өөрийн хүслээр их өөрчилсөн хувилбар юм.

Функцүүд

urlopen нь urllib-ийн urlopen –тэй адил шинж бүхий urlopen функцтэй. urllib2-ийг өөрийн хүслээр өөрчилье гэвэл суурилуулаад urlopen–ийг дуудахын өмнө build_opener ба install_opener функцүүдийг ашиглан хэд хэдэн боловсруулагчийг бүлэглэх болдог.

Хүсэх(Request) класс

URL-тэмдэгт мөрийн оронд Request классын жишээг urlopen функц рүү сайн дурааараа дамжуулна.

OpenerDirector класс

OpenerDirector классын *d* жишээ нь боловсруулах классуудыг цуглуулдаг ба URL-ийн янз бүрийн схемийг нээж алдаагаа боловсруулахад тэдгээрийг нарийн зохицуулдаг.

Боловсруулах классууд

urllib2 модуль нь хүслээрээ өөрчилдөг боловсруулах классын дээд класс болгон ашигладаг BaseHandler классыг агуулна.

И-мэйлийн протокол

Өнөөдөр ихэнхдээ и-мэйлийг Захиа Дамжуулах Энгийн Протокол (SMTP)-оор илгээж Шуудангийн Төвийн Протоколын (POP3) 3-р хувилбарыг ашиглан хүлээн авдаг. Python нь эдгээр протоколыг дэмжсэн smtplib ба poplib гэсэн стандарт модультай.

poplib модуль

poplib модуль нь POP шуудангийн хайрцагт хандах үүрэгтэй POP3 классыг агуулна.

e smtplib модуль

Smtplib модуль нь SMTP серверт и-мэйл илгээдэг SMTP классыг агуулна.

HTTP ба FTP протокол

urllib , urllib2 модуль нь http, https, ба ftp-протокол ашиглан серверт хандах гарын доорх арга болж өгдөг. Python –ны стандарт сан нь эдгээр протоколд зориулсан тусгай модулиудтай.

httplib модуль

httplib модуль нь HTTP серверт холбогдох үүрэгтэй HTTPConnection классыг агуулна.

ftplib модуль

ftplib модуль нь FTP серверт холбогдох үүрэгтэй FTP классыг агуулна.

Сокетын ба сервер талын сүлжээний протоколын модулиуд

Интернэттэй холбогдохдоо программ нь сокет гэсэн обьектүүдийг ашигладаг. Python-ны сан нь сокеттой харьцдаг socket модультай байдаг.

socket модуль

socket модуль сокет обьект үүсгэхийн тулд дуудаж ашигладаг socket нэртэй функцтэй байдаг. Үүнийг ашиглан *s* гэсэн обьектийг үүсгэсэн гэж үзье. Клиент программаас сервер рүү холбогдохын тулд *s*.connect –ийг дуудна. Сервер программ клиентийн холболтыг хүлээхийн тулд *s*.bind and *s*.listen-ийг дуудаж ашиглана. Клиент нь холбогдохыг хүсвэл *s*.accept –ийг дуудан хүлээн авах ба *s*.accept нь клиент рүү холбоотой *s1* гэсэн өөр 1 обьектыг буцаана.Нэгэнт холболт тогтсоны дараа send методыг дуудан өгөгдөл дамжуулах ба recv методыг дуудан өгөгдлийг илгээнэ.

socket модуль нь error гэсэн exception-ний класстай байдаг. socket -ийн функц ба метод нь сокеттэй холбоотой алдааг боловсруулахдаа –ийг error –ийг үүсгэдэг.Эдгээр функц нь бүхэл тоо гэх мэт өгөгдлийг серверийн төрөлх формат ба сүлжээний стандарт форматбн хооронд хувиргах функц байж болно. Харилцаж байгаа программуудын ашиглаж байгаа сокетийн дээд түвшний протокол нь ямар хувиргалтыг хийхийг тодорхойлно.

socket функцүүд

socket модуль нь байнга ашиглагддаг функцүүдтэй.

SocketServer модуль

Python-ны сан нь энгийн интернэт серверийг хэрэгжүүлэх боломжтой SocketServer модультай. Модуль нь TCP-ийг хэрэглэдэг серверт чиглэсэн TCPServer класс, UDP-ийг хэрэглэдэг серверт чиглэсэн UDPServer класстай бөгөөд энэ 2 ижил интерфэйстэй.

TCPServer ба UDPServer-ны жишээ нь маш олон шинж чанар, методуудтай бөгөөд зарим методыг дарж өөрчлөх өөр классын дэд класс болгох зэргээр өөрийн серверт тусгайлан зохицуулж болно.

BaseRequestHandler класс

SocketServer –ийг хэвийн байдлаар ашиглахдаа SocketServer –ийн BaseRequestHandler-классыг дэд класс болгон handle методыг дарж хэрэглэдэг.

HTTP серверүүд

BaseHTTPServer, SimpleHTTPServer, CGIHTTPServer, ба SimpleXMLRPCServer модуль нь SocketServer модуль дээр тулгуурлан янз бүрийн төгс, боловсронгуй HTTP серверүүдийг хэрэгжүүлсэн хувилбарууд юм.

BaseHTTPServer модуль

BaseHTTPServer модуль нь SocketServer.TCPServer –ийн дэд класс болсон сервер HTTPServer классыг агуулах ба яг ижилхэн ашиглагддаг.Энэ нь SocketServer.BaseRequestHandler –ийн дэд класс болох BaseHTTPRequestHandler гэсэн хүсэлт боловсруулах класстай байх ба HTTP серверт хэрэглэгддэг метод, шинж чанаруудыг нэмж өгнө .