



Python Programming

Лекц 7

Багш Ж.Золжаргал
Х.Хулан

OBJECTS



- Паятон нь олон төрлийн өгөгдөл дэмждэг

1234 3.14159 "Hello" [1, 5, 7, 11, 13]

{"CA": "California", "MA": "Massachusetts"}

- Тус бүр нь объект, объект бүр нь:
 - Төрөлтэй
 - Дотоод өгөгдлийн дүрслэлтэй (анхдагч, нийлмэл)
 - Объекттэй харьцах функцын олонлогтой
- Объект нь тухайн төрлийн нэг тохиолдол
 - 1234 нь int төрлийн тохиолдол
 - "Hello" нь string төрлийн тохиолдол

OBJECT ORIENTED PROGRAMMING (OOP)



- Паятон дээрх бүх зүйл объект юм. (бүгд төрөлтэй)
 - Ямар нэгэн төрлийн **шинэ объект үүсгэж** болно
 - **Объектуудыг удигдаж** болно
 - **Объектуудыг устгаж** болно
 - del-ийг хэрэглэж болно эсвэл зүгээр л мартаж болно
 - Паятон систем нь устгагдсан эсвэл хандах боломжгүй объектуудыг санах ойгоос цэвэрлэгдэг. Үүнийг **"Garbage collection"** гэж нэрлэгдэг.

WHAT ARE OBJECTS?



- Объект нь өгөгдлийн хийсвэрлэл ...
 - **Дотоод дүрслэл**
 - Өгөгдлийн шинж чанаруудаар дамжуулан
 - Объекттэй харьцас **интерфэйс**
 - Гишүүн функцуудээр дамжуулан (procedure, function)
 - Зан байдлыг тодорхойлох боловч гүйцэтгэлийг нууна

EXAMPLE: [1, 2, 3, 4] has type list



- Жагсаалт нь дотооддоо хэрхэн дүрслэгдсэн байдаг вэ?
Нүднүүдийг холбосон жагсаалт



- Жагсаалтыг хэрхэн удирдах вэ?
 - `L[i]`, `L[i:j]`, `+`
 - `len()`, `min()`, `max()`, `del(L[i])`
 - `L.append()`, `L.extend()`, `L.count()`, `L.index()`,
`L.insert()`, `L.pop()`, `L.remove()`, `L.reverse()`, `L.sort()`
- Дотооддүрслэл нь private хандалттай байх ёстой

Заагч нь дараагийн
элементийг заана

ADVANTAGES OF OOP



- Өгөгдлийг багц түүн дээр ажилладаг функцуудын хамт багц болгож, сайн тодорхойлогдсон интерэйсээр харьцах
- Хуваан захирах хөгжүүлэлт
 - Класс бүрийг тусад нь хэрэгжүүлэх, тестлэх
 - Модулийг нэмэгдүүлэх замаар төвөгтэй байдлыг бууруулах
- Класс нь кодыг дахин ашиглахад хялбар болгодог
 - Паятоны олон модулиуд шинэ классаар тодорхойлогддог
 - Класс бүр тусдаа орчинтой байна
 - Удамшил нь дэл ангиудад дахин тодорхойлох болон өргөтгөх боломж олгодог

CREATING AND USING YOUR OWN TYPES WITH CLASSES



- Класс үүсгэх болон классын тохиолдол үүсгэж ашиглахын хоорондын ялгаа
- Класс үүсгэхэд
 - Классын нэрийг тодорхойлох
 - Классын атрибутыг тодорхойлох
 - Жишээ нь, хэн нэгэн жагсаалт классыг хэрэгжүүлэх код бичих
- Классыг хэрэглэх
 - Шинэ тохиолдол үүсгэх
 - Тухайн тохиолдолдээр үйлдэл хийх
 - Жишээ нь, $L = [1, 2]$ ба `len(L)`

DEFINE YOUR OWN TYPES



- Шинэ төрөл үүсгэхэд class түлхүүр үгийг хэрэглэнэ

```
class Coordinate(object):  
    #define attributes here
```

name/type (pointing to `Coordinate`)
class parent (pointing to `object`)
class definition (pointing to the entire line)

- def -тэй төстэй, классын тодорхойлолтонд нэг индент зай авсанаар классын биеийг илэрхийлнэ
- Coordinate класс нь object классаас удамшиж байна. (түүний бүх атрибут шинж чанарыг авна)
 - Coordinate нь object-ийн хүү класс
 - object нь Coordinate-ийн эцэг класс (superclass)

WHAT ARE ATTRIBUTES?



- Класс-д харъяалагдах өгөгдөл болон функц
- **Data attributes**
 - Өөр объектуудаар илэрхийлэх өгөгдлийн тухай бодох, тэдгээр нь классыг үүсгэнэ
 - Жишээ нь, Coordinate нь хоёр тоон утгатай
- **Methods** (procedural attributes)
 - Method буюу функцын тухай бодно, тэдгээр нь зөвхөн тухайн класстай ажилладаг байна
 - Объекттэй хэрхэн харьцах
 - Жишээ нь, Хоёр координат объектын хоорондын зайг тодорхойлох

DEFINING HOW TO CREATE AN INSTANCE OF A CLASS

- Объектын тохиолдлыг хэрхэн үүсгэхийг эхэлж тодорхойлох
- `__init__` тусгай функцыг зарим аргументтэй дуудаж ашигладаг.

```
class Coordinate(object):  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y
```

special method to create an instance — is double underscore

what data initializes a Coordinate object

parameter to refer to an instance of the class

two data attributes for every Coordinate object

ACTUALLY CREATING AN INSTANCE OF A CLASS



```
c = Coordinate(3, 4)
origin = Coordinate(0, 0)
print(c.x)
print(origin.x)
```

use the dot to
access an attribute
of instance `c`

create a new object
of type
`Coordinate` and
pass in 3 and 4 to
the `__init__`


- Объектын шинж чанарыг объектын хувьсагч гэнэ
- Self-д утга дамжуулахгүй, паятон үүнийг автоматаар хийдэг

WHAT IS METHOD?



- Процедурын шинж чанар, Функц шиг, зөвхөн класс дээр л хэрэгжинэ
- Паятон үргэлжэхний аргументээр нь объектыг өөрийг нь дамжуулдаг.
 - Бүх гишүүн функцын эхний аргумент нь self байна.
- “.” үйлдэл нь ямар нэгэн атрибут руу **хандахад** хэрэглэгддэг.
 - Объектын шинж чанар
 - Объектын гишүүн өгөгдөл

DEFINE A METHOD FOR THE Coordinate CLASS



```
class Coordinate(object):  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y  
    def distance(self, other):  
        x_diff_sq = (self.x - other.x) ** 2  
        y_diff_sq = (self.y - other.y) ** 2  
        return (x_diff_sq + y_diff_sq) ** 0.5
```

use it to refer to any instance

another parameter to method

dot notation to access data

- **Self** болон “.” үйлдэлээс бусад нь яг л функц шиг.
 - (аргумент авна, үйлдэл хийнэ, үрдүн буцаана)

HOW TO USE METHOD

```
def distance(self, other):  
    # code here
```

method def

Using the class:

- conventional way

```
c = Coordinate(3,4)  
zero = Coordinate(0,0)  
print(c.distance(zero))
```

*object to call
method on*

*name of
method*

*parameters not
including self
(self is
implied to be c)*

- equivalent to

```
c = Coordinate(3,4)  
zero = Coordinate(0,0)  
print(Coordinate.distance(c, zero))
```

*name of
class*

*name of
method*

*parameters, including an
object to call the method
on, representing self*

PRINT PRESENTATION OF AN OBJECT




```
>>> c = Coordinate(3,4)
>>> print(c)
<__main__.Coordinate object at 0x7fa918510488>
```

- Объектын дүрслэлийг хэвлэх, анхдагч
- Классад нь **`__str__`** функцыг тодорхойлох
- Паятон нь объектыг хэвлэх үйлдэл хийхэд `__str__` функцыг дууддаг
- Тухайн объектыг хэвлэхэд юу хийхийг та энд тодорхойлж өгнө.

```
>>> print(c)
<3,4>
```

DEFINING YOUR OWN PRINT METHOD



```
class Coordinate(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def distance(self, other):
        x_diff_sq = (self.x-other.x)**2
        y_diff_sq = (self.y-other.y)**2
        return (x_diff_sq + y_diff_sq)**0.5
    def __str__(self):
        return "<" + str(self.x) + ", " + str(self.y) + ">"
```

name of
special
method

must return
a string

WRAPPING YOUR HEAD AROUND TYPES AND CLASSES

-

```
>>> c = Coordinate(3,4)
```

```
>>> print(c)
```

```
<3,4>
```

```
>>> print(type(c))
```

```
<class __main__.Coordinate>
```

return of the `__str__` method
the type of object `c` is a class `Coordinate`

-

```
>>> print(Coordinate)
```

```
<class __main__.Coordinate>
```

```
>>> print(type(Coordinate))
```

```
<type 'type'>
```

a `Coordinate` is a class
a `Coordinate` class is a type of object

```
>>> print(isinstance(c, Coordinate))
```

```
True
```

SPECIAL OPERATORS



- +, -, ==, <, >, len(), print, болон өөр бусад
 - <https://docs.python.org/3/reference/datamodel.html#basic-customization>
- print шиг, класстайгаа ажиллах функцуудыг даран тодорхойлж болно.

<code>__add__(self, other)</code>	→	<code>self + other</code>
<code>__sub__(self, other)</code>	→	<code>self - other</code>
<code>__eq__(self, other)</code>	→	<code>self == other</code>
<code>__lt__(self, other)</code>	→	<code>self < other</code>
<code>__len__(self)</code>	→	<code>len(self)</code>
<code>__str__(self)</code>	→	<code>print self</code>

POWER OF OOP



- Объектыг **багц**аар нь хуваалцана
 - Ерөнхий шинж чанар
 - Шинж чанарууд дээрээ үйлдэл хийх функцууд
- **Хийсвэрлэл** ашиглан объектыг хэрхэн хэрэгжүүлэх, объектыг хэрхэн ашиглахыг хооронд нь ялгаж өгнө
- Объектын хийсвэрлэлийн давхаргы бүтээх, бусад ангилалын объектуудаас зан авирыг өвлөн авах
- Өөрийн объектын ангийг паятоны үндсэн классаас удамшуулах