



Python Programming

Лекц 8

Багш Ж.Золжаргал
Х6Хулан

IMPLEMENTING THE CLASS

vs

USING THE CLASS



- Кодыг хоёр өөр өнцөгөөс бичих

Шинэ объектын төрлийг
классаар **хэрэгжүүлэх**

- Класс **тодорхойлох**
- **Шинж чанар** тодорхойлох
(Ямар объект вэ?)
- **Гишүүн функц**
тодорхойлох (яаж ашиглах
вэ?)

Кодонд шинэ объектын
төрөл **хэрэглэх**

- Объектын төрлийн
тохиолдол үүсгэх
- Тэдэнтэй **үйлдэл** хийх

IMPLEMENTING THE CLASS



vs

USING THE CLASS

Классын нэр нь төрөл

```
class Coordinate(object)
```

Класс нь бүх тохиолдол
дээр байдаг өгөгдөл болон
гишүүн функцыг ерөнхийд
нь тодорхойлдог.

Тодорхой нэг объект нь
тохиолдол юм.

```
coord = Coordinate(1,2)
```

Тохиолдол нь классын
бүтэцтэй байдаг.

WHY USE OOP AND CLASSES OF OBJECTS?

- Бодит амьдралыг дуурайх
- Нэг төрлийн өөр объектуудыг бүлэглэх



Jelly
1 year old
brown



5 years old
brown



Tiger
2 years old
brown



Bean
0 years old
black



2 years old
white

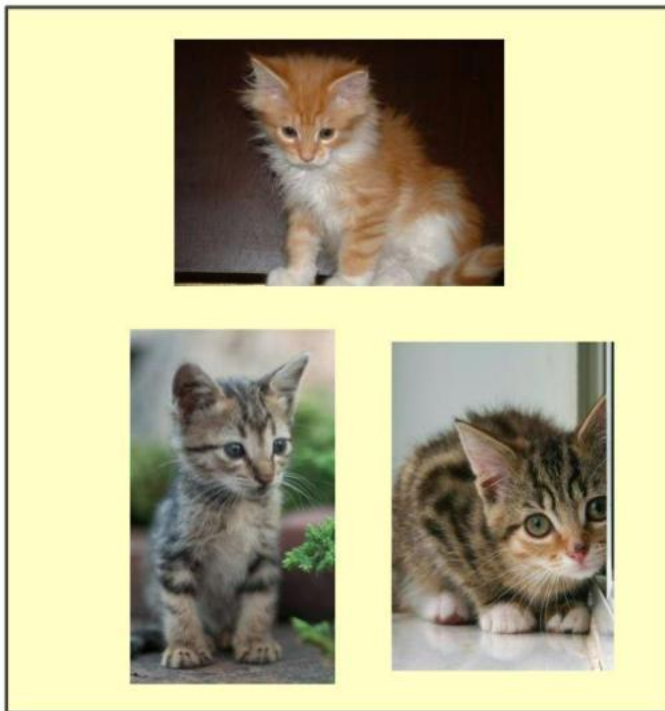


1 year old
b/w

WHY USE OOP AND CLASSES OF OBJECTS?



- Бодит амьдралыг дуурайх
- Нэг төрлийн өөр объектуудыг бүлэглэх

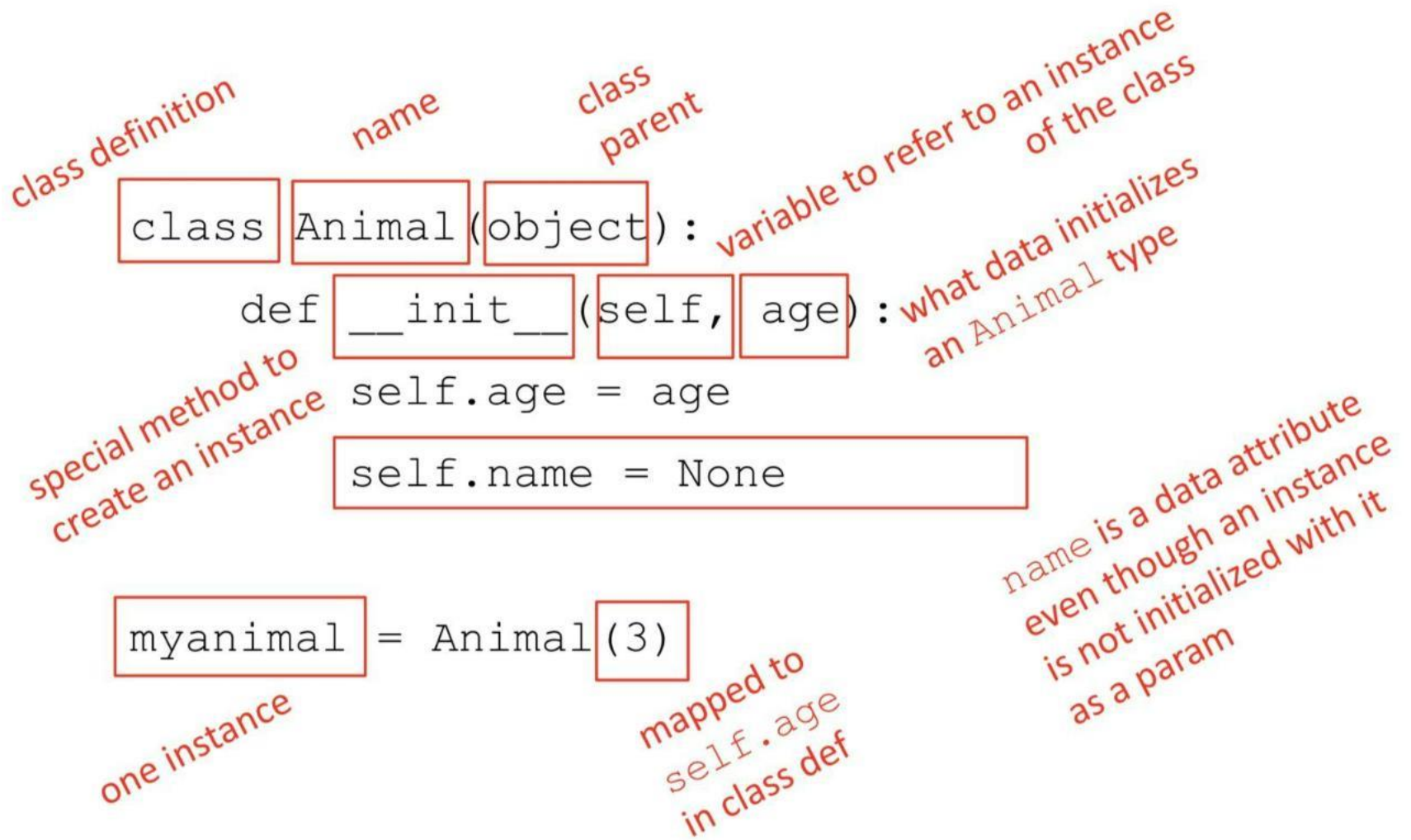


GROUPS OF OBJECTS HAVE ATTRIBUTES



- **Data attributes**
 - Объектоо өгөгдлөөр хэрхэн төлөөлж чадах вэ?
 - **Энэ юу вэ?**
 - Координатын хувьд: x ба y -ийн утга
 - Амьтаны хувьд: нас, нэр
- **Procedural attributes** (behavior/operations/methods)
 - Хэн нэгэн объекттэй хэрхэн харьцаж болох вэ?
 - **Энэ юу хийх вэ?**
 - Координатын хувьд: хоёр цэгийн хоорондох зайг олох
 - Амьтаны хувьд: дуу гаргах

HOW TO DEFINE A CLASS



The diagram illustrates the components of a Python class definition and its instantiation. It features two code snippets with various parts highlighted by red boxes and annotated with red text labels.

Class Definition:

```
class Animal(object):  
    def __init__(self, age):  
        self.age = age  
        self.name = None
```

Annotations for Class Definition:

- class definition:** Points to the `class` keyword.
- name:** Points to the `Animal` class name.
- class parent:** Points to the `(object)` parent class.
- variable to refer to an instance of the class:** Points to the `self` parameter in the `__init__` method.
- special method to create an instance:** Points to the `__init__` method name.
- what data initializes an Animal type:** Points to the `age` parameter in the `__init__` method.

Instantiation:

```
myanimal = Animal(3)
```

Annotations for Instantiation:

- one instance:** Points to the `myanimal` variable.
- mapped to self.age in class def:** Points to the `age` parameter in the `__init__` method.
- name is a data attribute even though an instance is not initialized with it as a param:** Points to the `self.name = None` line in the class definition.

GETTER AND SETTER METHODS



```
class Animal(object):
    def __init__(self, age):
        self.age = age
        self.name = None

    def get_age(self):
        return self.age
    def get_name(self):
        return self.name

    def set_age(self, newage):
        self.age = newage
    def set_name(self, newname=""):
        self.name = newname

    def __str__(self):
        return "animal:" + str(self.name) + ":" + str(self.age)
```

getter

setter

- Классын гаднаас шинж чанар руу хандахад ашиглана

AN INSTANCE and DOT NOTATION



- Классын тохиолдолыг үүсгэх

```
a = Animal(3)
```

- Цэг тэмдэглэгээ нь шинж чанар руу хандах (data, methods) ашигладаг. Илүү сайн арга нь өгөгдөл рүү хандахад getter, setter-ийг ашиглах юм.

```
a.age
```

```
a.get_age()
```

- access method
- best to use getters
and setters

- access data attribute
- allowed, but not recommended

INFORMATION HIDING



- Классыг тодорхойлсон хүн нь магадгүй **шинжчанаарын хувьсагчийн нэрийг сольж** болно.

*replaced age data
attribute by years*

```
class Animal(object):  
    def __init__(self, age):  
        self.years = age  
    def get_age(self):  
        return self.years
```

- Хэрэв классын гаднаас шинж чанар руу хандахад, **классын тодорхойлолт нь өөрчлөгдсөн** бол алдаа
- Классын гаднаас getter ба setter-ийг **a.get_age()** хэрэглэх, **a.age** биш
 - Сайн хэв маяг, кодыг засварлахад хялбар, алдаанаас урьдчилан сэргийлэх

PYTHON NOT GREAT AT INFORMATION HIDING



- Классын гаднаас өгөгдөлд хандах боломжыг олгоно
- `print(a.age)`
Классын гаднаас өгөгдөл рүү бичилт хийхийг зөвшөөрнө
- `a.age = 'infinite'`
Класс тодорхойлолтын гаднаас объектод шинж чанар нэмэхийг зөвшөөрнө.
- `a.size = "tiny"`
Эдгээрийн аль нэгийг нь хийх нь **тийм сайн загвар биш**

DEFAULT ARGUMENTS



- Бодит аргумент өгөөгүй үед, албан ёсны аргументыг ашигладаг

```
def set_name(self, newname="") :  
    self.name = newname
```

- Анхдагч аргумент энд ашиглаж байна

```
a = Animal(3)
```

```
a.set_name()
```

```
print(a.get_name())
```

prints ""

- Дамжуулсан аргумент энд хэрэглэж байна

```
a = Animal(3)
```

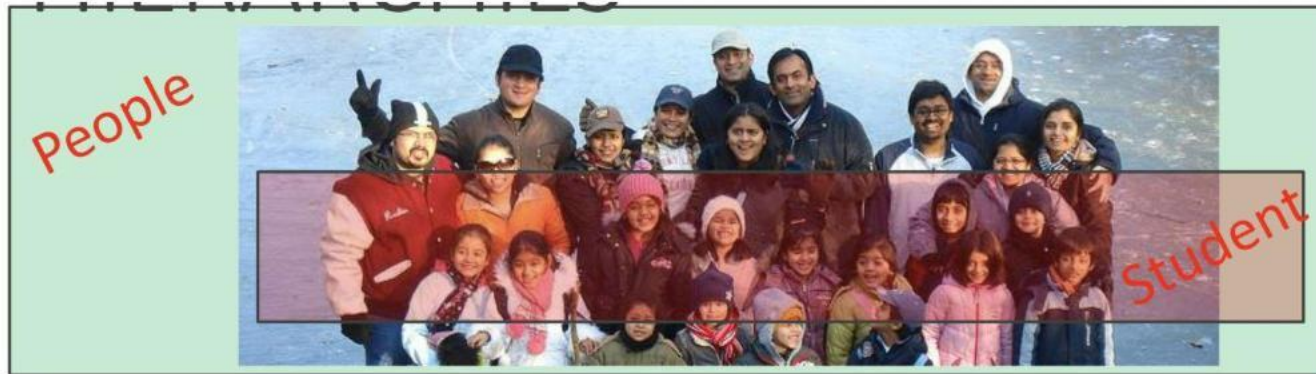
```
a.set_name("fluffy")
```

```
print(a.get_name())
```

prints "fluffy"

HIERARCHIES

Animal



Cat



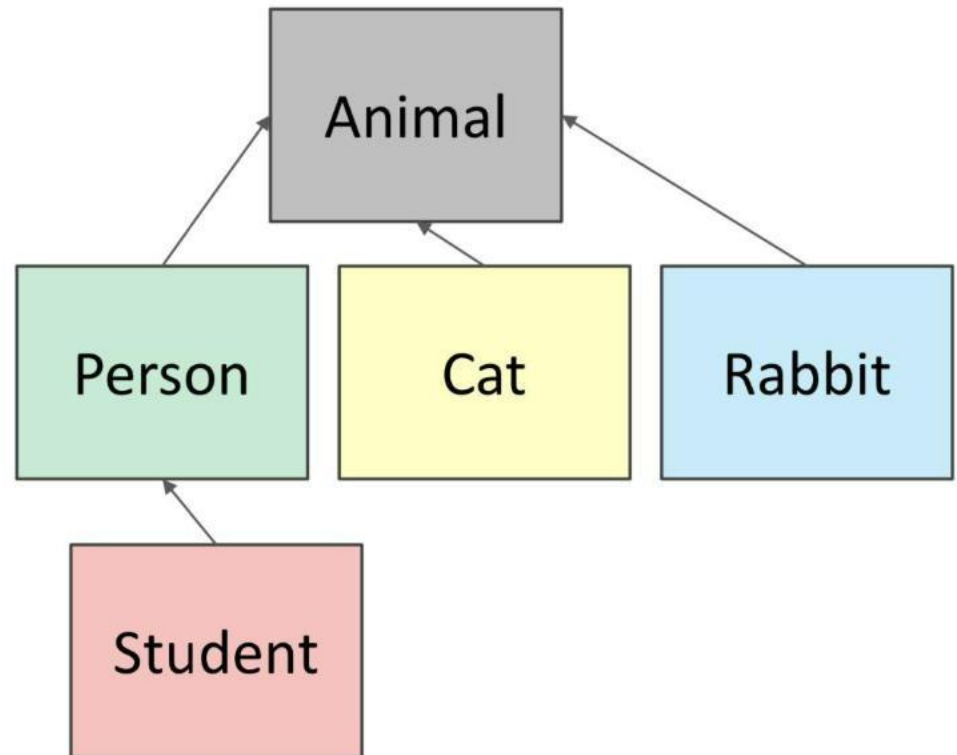
Rabbit



HIERARCHIES



- **Parent class** (superclass)
- **Child class** (sub class)
 - Эцэг классын бүх шинж чанар болон үйлдлүүд удамшина
- **Мэдээлэл нэмнэ**
- **Үйлдэл нэмнэ**
- **Зан араншинг даран тодорхойлно**



INHERITANCE: PARENT CLASS



```
class Animal(object):  
    def __init__(self, age):  
        self.age = age  
        self.name = None  
    def get_age(self):  
        return self.age  
    def get_name(self):  
        return self.name  
    def set_age(self, newage):  
        self.age = newage  
    def set_name(self, newname=""):  
        self.name = newname  
    def __str__(self):  
        return "animal:" + str(self.name) + ":" + str(self.age)
```

- everything is an object
- class object
implements basic
operations in Python, like
binding variables, etc

OBJECTS

inherits all attributes of Animal:
__init__()
age, name
get_age(), get_name()
set_age(), set_name()
__str__()

add new
functionality via
speak method

```
class Cat(Animal):
```

```
    def speak(self):
```

```
        print("meow")
```

```
    def __str__(self):
```

```
        return "cat:" + str(self.name) + ":" + str(self.age)
```

overrides __str__

- Гишүүн функц нэмэх speak()
 - Cat төрлийн тохиолдолоор шинэ аргыг дуудах
 - Animal төрлийн тохиолдлоор шинэ арга дуудвал алдаа болно
- __init__ бол Animal-ийн хувьд бас ашиглагдаж болно

WHICH METHOD TO USE?



- Хүү класс нь эцэг классыхтэй **ижил нэртэй гишүүн функц**тэй байж болно
- Классын тохиолдолын хувьд, **тухайн классын тодорхойлолтоо**соо хайна
- Хэрэв олдохгүй бол, **дээд шаталсан бүтцээс** функцын нэрээр хайна (эцэг, өвөг эцэг, гэх мэт)
- Шаталсан бүтэцийн аль түрүүлж олдсон функц хэрэглэнэ

```
class Person(Animal):
```

```
    def __init__(self, name, age):
```

```
        Animal.__init__(self, age)
```

```
        self.set_name(name)
```

```
        self.friends = []
```

```
    def get_friends(self):
```

```
        return self.friends
```

```
    def add_friend(self, fname):
```

```
        if fname not in self.friends:
```

```
            self.friends.append(fname)
```

```
    def speak(self):
```

```
        print("hello")
```

```
    def age_diff(self, other):
```

```
        diff = self.age - other.age
```

```
        print(abs(diff), "year difference")
```

```
    def __str__(self):
```

```
        return "person:" + str(self.name) + ":" + str(self.age)
```

parent class is Animal

call Animal constructor
call Animal's method
add a new data attribute

new methods

override Animal's
__str__ method

```
import random
```

```
class Student(Person):
```

```
    def __init__(self, name, age, major=None):
```

```
        Person.__init__(self, name, age)
```

```
        self.major = major
```

```
    def change_major(self, major):
```

```
        self.major = major
```

```
    def speak(self):
```

```
        r = random.random()
```

```
        if r < 0.25:
```

```
            print("i have homework")
```

```
        elif 0.25 <= r < 0.5:
```

```
            print("i need sleep")
```

```
        elif 0.5 <= r < 0.75:
```

```
            print("i should eat")
```

```
        else:
```

```
            print("i am watching tv")
```

```
    def __str__(self):
```

```
        return "student:" + str(self.name) + ":" + str(self.age) + ":" + str(self.major)
```

bring in methods
from random class

inherits Person and
Animal attributes

adds new data

- I looked up how to use the
random class in the python docs
- random() method gives back
float in [0, 1)

CLASS VARIABLES AND THE Rabbit SUBCLASS

- **Классын хувьсагч**ууд, классын бүх тохиолдлууд хоорондоо хуваалцах хувьсагч

```
class Rabbit(Animal):  
    tag = 1  
    def __init__(self, age, parent1=None, parent2=None):  
        Animal.__init__(self, age)  
        self.parent1 = parent1  
        self.parent2 = parent2  
        self.rid = Rabbit.tag  
        Rabbit.tag += 1
```

parent class

class variable

instance variable

access class variable

incrementing class variable changes it for all instances that may reference it

- Rabbit-ийн бүх тохиолдолд tag нь **давхардахгүй нэршил** байх ёстой


Rabbit GETTER METHODS

```
class Rabbit(Animal):
    tag = 1
    def __init__(self, age, parent1=None, parent2=None):
        Animal.__init__(self, age)
        self.parent1 = parent1
        self.parent2 = parent2
        self.rid = Rabbit.tag
        Rabbit.tag += 1
    def get_rid(self):
        return str(self.rid).zfill(3)
    def get_parent1(self):
        return self.parent1
    def get_parent2(self):
        return self.parent2
```

method on a string to pad
the beginning with zeros
for example, 001 not 1


- getter methods specific
for a Rabbit class
- there are also getters
get_name and get_age
inherited from Animal

WORKING WITH YOUR OWN TYPES



```
def __add__(self, other):  
    # returning object of same type as this class  
    return Rabbit(0, self, other)
```

recall Rabbit's `__init__(self, age, parent1=None, parent2=None)`



- Хоёр rabbit-ийн тохиолдлыг хооронд нь **+ үйлдэл** тодорхойлох
 - Үүн шиг: $r4 = r1 + r2$ энэ тохиолдолд $r1, r2$ нь rabbit-ийн тохиолдол
 - $R4$ нь 0 настай шинэ rabbit
 - $R4$ нь `self` болон `other` гэсэн эцгүүдтэй
 - `__init__` -д **`parent1, parent2` нь Rabbit-ийн төрлүүд**

SPECIAL METHOD TO COMPARE TWO Rabbits



- Хэрэв 2 туулай нь эцэг эх нь ижилхэн бол тухайн 2 туулайг тэнцүү гэж үзье

booleans

```
def __eq__(self, other):  
    parents_same = self.parent1.rid == other.parent1.rid \  
                   and self.parent2.rid == other.parent2.rid  
    parents_opposite = self.parent2.rid == other.parent1.rid \  
                      and self.parent1.rid == other.parent2.rid  
    return parents_same or parents_opposite
```

- Эцэг эхүүдийн id-г харьцуулж шалгана (давхардахгүй)
- Объектыг шууд харьцуулах боломжгүй
 - Жишээ нь: `self.parent1 == other.parent1`
 - Энэ нь `__eq__` функц цаанаа дуудагдана

OBJECT ORIENTED PROGRAMMING



- Өөрийн **өгөгдлийн цуглуулгыг** бий болгох
- Мэдээллийг **зохион байгуулах**
- Ажлын **хувиарлалт**
- Мэдээллийг **тогтмол** байдлаар авах
- Нарийн төвөгтэй байдалд **давхрага** нэмэх
- Функц шиг, класс нь програмчлалд **хуваан задлах** болон **хийсвэрлэлийн** механизм юм.