INTRODUCTION TO PYTHON

- Lec 4 Object Oriented (OOP)

TOPICS

- classes and objects (definition)
- Creating Class and Object in Python
- instance methods
- inheritance
- more...

tekomoro

classes and objects (definition)

- Ref
- Class
  - A class is a blueprint for the object.
  - A “parrot” class will contains all the details about the parrot:
    - name, colors, size etc.
    - Based on these descriptions, we can study about the parrot. Here, a Parrot is a class.
  - class Parrot:
  - pass

tekomoro

classes and objects (definition)

- Object
  - An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.
  -
  - The example for object of parrot class can be:
  - obj = Parrot()

tekomoro

Creating Class and Object in Python

```
class Parrot:
    # class attribute
    species = "bird"

    # instance attribute
    def __init__(self, name, age):
        self.name = name
        self.age = age

# instantiate the Parrot class
blu = Parrot("Blu", 10)
woo = Parrot("Woo", 15)

# access the class attributes
print("Blu is a {}".format(blu.__class__.species))
print("Woo is also a {}".format(woo.__class__.species))

# access the instance attributes
print("{} is {} years old".format( blu.name, blu.age))
print("{} is {} years old".format( woo.name, woo.age))
```

tekomoro

Creating Instance Methods in class

```
class Parrot:

    # instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # instance method
    def sing(self, song):
        return "{} sings {}".format(self.name, song)

    def dance(self):
        return "{} is now dancing".format(self.name)

# instantiate the object
blu = Parrot("Blu", 10)

# call our instance methods
print(blu.sing('"Happy"'))
print(blu.dance())
```

tekomoro

__init__() Function

Most classes (and object) have an initialization function. This is **automatically** run when we create an instance of the class. To define the initialization routine, we name it __init__(self, param)

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

self Parameter

- The word "self" refers to the fact that we are using a variable or function within the specific instance of the class or object.
- The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

tekomoro

Use of Inheritance in Python

```
# parent class
class Bird:

    def __init__(self):
        print("Bird is ready")

    def whoisThis(self):
        print("Bird")

    def swim(self):
        print("Swim faster")

    def fly(self):
        print('Yes, can fly')
```

```
# child class
class Penguin(Bird):

    def __init__(self):
        # call super() function
        super().__init__()
        print("Penguin is ready")

    def whoisThis(self):
        print("Penguin")

    def run(self):
        print("Run faster")

    def fly(self):
            super().fly()
            print("But very limited -
almost never")
```

```
# test

peggy = Penguin()
peggy.whoisThis()
peggy.swim()
peggy.run()
peggy.fly()
```

tekomoro

More on OOP

Python - Object Oriented - tutorialspoint: Deeper intro to OPP - beyond what is needed for the course. Highly recommended:

- Optional class documentation string
- Built-In Class Attributes: __dict__, __doc__, __name__, __module__, __bases__
- multiple inheritance: *class C(A, B): # subclass of A and B*
- Overriding Methods
- Base Overloading Methods

Official Python Reference

tekomoro

Exercises on OOP

HackerRank: https://docs.python.org/3/tutorial/classes.html

Python Practice Book: https://anandology.com/python-practice-book/object_oriented_programming.html

Python Practice

tekomoro

Thank You

^_^

tekomoro