* Python supports OOPS as well as scripting.

27th - June

Python - tutorial 5

OOPS : Object Oriented Programming System

[class - blue print of real world object (classification)
object - real world entity on which we can manipulate. (variable of class = object).

ex: (car - class, MANO - object)

```
i = 10
print(type(i)) → <class 'int'>
```

(∵ i is variable of class integer)

(Python is designed by using OOPS concept.

[OOP concept brings structure to a code.]

Note: (Its upto us whether we want to use scripting / OOPS, for larger codes OOPS is recommended.)

- Class:

```
class class_name:
    pass
```

(∵ when you are not interested to write body of anything just write pass.)

- Declaration of class, object, its variables:

(Method-1) → (not accepted way...).

```
class test:
    pass
object ← a = test()
print(type(a)) → <class '__main__.test'>
```

(∵ a is variable of class 'test').

[__init__ is predefined function basically used to initialise a variable or class.]

[self - whenever we call any function inside class, we are suppose to write that function starting with pointer (self / any other name)

```
class test():
    pass
b = test()
b.name = "MADHU"
b.age = 22
b.dob = 250299

c = test()
c.name = "MUSKAN"
c.age = 23
c.surname = "JAIN"
```

[b.age → 22
c.surname → 'JAIN

[Method-2] → (init method) (Note: self is not a keyword, we can use any other name too)

```
class Person:
    def __init__(self, name, surname, dob):
        self.name = name
        self.surname = surname
        self.dob = dob
```

(first argument always behaves like a pointer).

(understood by class)

a = Person() (; a = object of class Person)

→ error (__init__ method is looking for (_,_,_) three arg

when we create, no need
will always be a 'pointer',
to define pointer.

a = Person("MADHU", "JAIN", 250299)
(name, surname, dob)

a.name → 'MADHU' (object a has variable name value 'MADHU')

b = Person("MUSKAN", "PINCHA", 300422)

b.surname → 'PINCHA'

print(a) → <main.Person object at 0x0234....>
(∵ so here we don't get whole value of a, later we will use str)

(*)
class Person:
    def __init__(a, name, surname, dob): (init is predefined function)
        (pointer name can be same or diff. within a class)
        a.name = name
        a.surname = surname
        a.dob = dob → (age is user-defined)
    def age(a, current_year): (a or b)
        return current_year - a.dob
            (pointer.variable = calling variable inside class).
    def __str__(a): (__ means inbuilt fun.)
        (a → pointer)
        return "%s %s was born in %d."
            %(a.name, a.surname, a.dob)

alec = Person("Alec", "Baldwin", 1996) (3-arg. goes in init & str is output)
print(alec) (output by using __str__ func.)
print(alec.age(2014)) (2014 argument goes in age func.)
→ Alec Baldwin was born in 1996.
→ 18 (∵ 2014 - 1996)

Note: (__str__ is an inbuilt function which returns only strings.)

[Calling functions or variables:
inside class -> pointer.variable
outside class -> object.variable/function(argument)

Note: (we can use same/different pointers for diff. methods/functions of a class, but remember, first argument will always be a pointer when you are defining any function inside a class.)

(*)
```
class Person:
    def __init__(a, name):
        a.name = name
    def __init__(a, name, surname, dob):
        a.name = name
        a.surname = surname
        a.dob = dob
    def __init__(a, name, surname):
        a.name = name
        a.surname = surname
    def __str__(a)
        return "%s %s born in ____" %(a.name, a.surname)
```

(Multiple __init__ method: last one will be preferred).

(active)

```
alec1 = Person()
alec2 = Person("MADHU", "JAIN", 25021998)
alec3 = Person("MADHU")
alec4 = Person("MADHU", "JAIN")
```

error (No arg.)
error (3 arg.)
error (1 arg.)
MADHU JAIN born in ____ (2 arg. as last __init__ have 2 arg.)

Note: __init__ method is optional, used to initialize variable.

Method-3) → (less preferred)

```
class Person:
    def set_name(self, name):
        self.name = name
    def set_surname(self, surname):
        self.surname = surname
    def set_dob(self, dob):
        self.dob = dob
    def age(self, current_year):
        return current_year - self.dob
    def __str(self):
        return "%s %s was born in %d." \
        % (self.name, self.Surname, self.dob)

president = Person()
president.set_name("MAOHU")
president.set_surname("JAIN")
president.surname
```