PAGE NO.:
DATE: / /

PYTHON OOPs CONCEPTS

@the-programming-girl
Telegram- @coders_place

Object Oriented Programming

Python is a multiparadigm programming language. It supports different programming approches.
One of the most popular approches to solve programming problem by creating objects. This is known as a Object oriented programming. (OOP)

OOP has 2 characteristics:
1) Attributes
2) Behavior

Example:- A parrot is an object, as it has following properties.
- name, age, color as attributes.
- singing, dancing as behavior.

The concepts of OOPs in Python focuses on creating reusable code. This concept is also known as DRY (Dont Repeat Yourself)

PAGE NO.:
DATE: / /

Class.

A class is a blueprint for the object. We can think of class as a sketch of parrot with labels. It contains all details about the name, colors, size etc.

ex.

```
Class Parrot :
    pass.
```

Here class keyword define an empty class parrot. From class we construct instances. An instance is a specific object created from a particular class.

Object:-

An object (instance) is an instantiation of a class. When class is defined only the description for the class object is defined. Therfore, no memory or storage is allocated.

ex:-

```
Obj = Parrot()
```

@the-programming-girl

Here obj is an object of class Parrot.

Suppose we have details of parrots. Now we are going to show how to build the class and object of parrots.

We can access the class attribute using __class__.species.

PAGE NO:
DATE: / /

Inheritance :-

Inheritance is a way of creating new class for using details of an existing class without modifying it. The newly formed class is derived class, similarly, the existing class is a base class.

ex. Use of inheritance in Python.

```
class Bird :
    def __init__(self):
        print ("Bird is redy")
    def swim (self):
        print ("swim faster")
class Penguin (Bird):
    def __init(self):
        super().__init__()
    def run (self):
        print ("Run fast")
p = Penguin()
p.swim()
p.run()
```

@the-programming-girl

output

```
swim faster
run fast.
```

We can use the super() function inside the __init__() method. This allows us to run __init__() method

PAGE NO.:
DATE : / /

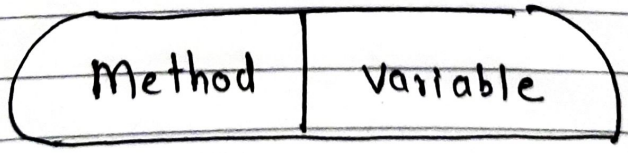
Encapsulation:-

Using OOP in python, we can restrict access to methods and variables. This prevents data from direct modification which is called encapsulation. In python we denote private attribute using _ as the prefix, i.e single _ or double __

```
Class Computer:
    def __init__(self):
        self.__maxprice = 900
    def setMaxPrice(self, price):
        self.__maxprice = price.
C = computer()
C.__maxprice = 1000
C.setMaxprice(100)
```

We used __init__() method to store the maximum selling price of computer.

```
C._maxprice = 1000
```



class

@the-programming-girl

Class member access specifier	Access from own class	Accessible from derived	Accessible from object
Private	Yes	No	No
protected	yes	Yes	No
public	Yes	Yes	No

PAGE NO.:
DATE : / /

Polymorphism :-

Polymorphism is an ability to use a common interface for multiple forms (data types) Polymorphism in python defines methods in the child class that have the same name as the methods in the parent class. It is possible to modify a method in a child class that it has inherited from ~~child~~ class. parent class.

```
Class Parrot :
    def fly (self):
        print ("Parrot can fly")
    def swim (self)
        print ("Parrot can't swim")
Class Penguin :
    def fly (self ):
        print ("Penguin cant fly")
    def swim (self):
        print ("Penguin can swim")
def flying_test (bird):
    bird.fly()

blu = Parrot()
peg = penguin()

flying_test (blu)
flying_test (peg)
```

@the-programming-girl

Output

Parrot can fly.
Penguin can't fly.

PAGE NO.:
DATE : / /

All PDF ARE UPLODED ON TELEGRAM

@the_programming_girl

LINK IN BIO