===

Python Programming

Python is a object-oriented, and high-level programming language. It was created
by Guido van Rossum, and first released on February 20, 1991.

===

Pyhton Programming

print() in python -> print() statement is used to write something on the
output screen.
It converts everything into string before writing it on the screen.

Example - print("Hello World!")

Comment in Python -> Comment are the lines which are ignored by compiler
whithout throwing an error. They are used to increase
the program redability or to explain logic of a function.

1. Single line comment -> These are written in a single line by using # symbol.

Example -
```
# This is comment
print("Comment Example")
```

2. Multiline comment -> These are written inside """......."""

Example -
```
""" This
is
comment"""
print("Multiline Comment")
```

Escape sequences -> These are the combination of cahracters which usually
start with character backslash (\).
These are used to design output.

1. \t -> For tab space
2. \n -> New line character
3. \' -> For single quotes
4. \" -> For double quotes
5. \\ -> For backslash

Example -

```
print("Hello\tWorld")
print("Hello\nWorld")
print("Hello\'World\')
print("\"Hello World\"")
print("\\Hello World\\")
```

```
***Assignment Questions***

1. Print Python    Programming
2. Print Python
         Programming
3. Print 'Python Programming'
4. Print "Python Programming"
5. Print \Python Programming\




***Assignment Solution***

1.
print("Python\tProgramming")

2.
print("Python\nProgramming")

3.
print("\'Python Programming\'")

4.
print("\"Python Programming\"")

5.
print("\\Python Pragramming\\")


=================================================================================

***Variable*** -> Variable are the name given to the memory location where we can
                  store any value, character or string.
In python we don't need to declare data type of a variable. Variable are created
autometically when we assign value to them.


Example -
1.
a = 10
b = 15
c = a + b      # a,b and c are integer type variables
print(c)

2.
a = "Hello "
b = "World"
c = a + b     # a,b and c are string type variables
print(c)

3.
a = 70
b = 43
a,b = b,a
```

Assignment Questions

1. Take two variable a and b and print a-b
2. print a*b
3. print a/b
4. print a%b
5. Take two variable of string type and print them in a new line
6. Take two variable and print the added string
7. Take four variable and print product of them

Assignment Solutions

1.
```
a = 92
b = 72
print(a-b)
```

2.
```
a = 5
b = 9
print(a*b)
```

3.
```
a = 77
b = 11
print(a/b)
```

4.
```
a = 83
b = 10
print(a%b)
```

5.
```
a = "Welcome"
b = "in Python"
print(a,"\n",b)
```

6.
```
a = "Welcome"
b = "in Python"
print(a+b)
```

7.
```
a = 4
b = 3
c = 5
d = 6
print(a*b*c*d)
```

==

```
***if statement*** -> It is the conditional statement which is used when we want to
                      do some task only if the condition is true.

Example -

a = 6
if a<10:
   print("a is less then 10")


=================================================================================

***if else*** -> It is used when we need to perform a task if condition is true
                 and another task if condition is false.

Example -

num = 44
if num%11 == 0:
     print("Divisible by 11")
else:
     print("Not divisible by 11")

=================================================================================

***if elif*** -> It is used to check multiple statements at same time and execute
                 one whose condition is satisfied.


Example -

num = 55
if num<30:
    print("Between 1 and 30")

elif num>30 && num<50:
    print("Between 31 and 50")

elif num>50 && num<70:
    print("Between 51 and 70")

else
    print("Greater then 70")
```

Assignment Questions

1. Find least number between two number.
2. Check weather the number is odd or even.
3. Check weather the number is multiple of 5 or not.

4. Check that the number is divisible by 7.
5. Program to print weekday based on given number. 1 - for monday, 2 - for tuesday, 3 - wednesday, ... 7 - for sunday
6. Check weather the number is positive or negative.
7. Check weather the number is greater then 5000 or not.

Assignment Solutions

```
1.
a = 93
b = 84
if a>b:
    print("a is greater")

2.
a = 81
if a%2==1:
    print("Number is odd")
else:
    print("Number is even")

3.
num = 3520
if num%5==0:
    print("Number is multiple of 5")
else:
    print("Not a multiple of 5")

4.
num = 294
if num%7==0:
    print("Divisible by 7")
else:
    print("Not divisible by 7")

5.
day = 3
if day==1:
    print("Monday")
elif day==2:
    print("Tuesday")
elif day==3:
    print("Wednesday")
elif day==4:
    print("Thursday")
elif day==5:
    print("Friday")
elif day==6:
    print("Saturday")
elif day==7:
    print("Sunday")
else:
    print("Invalid choice")
```

6.
```
num = -42
if num>0:
    print("Number is positive")
else:
    print("Number is negative")
```

7.
```
n = 5002
if n>5000:
    print("Greater then 5000")
else:
    print("Less then 5000")
```

===

Loops in Python

Loop is a iterative process which runs continiously till the condition is true.

while loop -> It is used to execute a block of statements till the condition is true.

Example -

```
i=1
while i<11:      # Declearation of while loop
    print(i)
    i=i+1
```

Assignment Questions

1. Print all even numbers from 1 to 30.
2. Print all odd numbers from 1 to 30.
3. Print the table of 5.
4. Find sum of first 10 natural numbers.
5. Find the sum of first 10 even numbers.
6. Find the sum of first 10 odd numbers.
7. Print negative numbers from -10 to -1.
8. Print factorial of 6.
9. Print the sum of digit of a 1234.

Assignment Solutions

1.

```
num = 2
while num<=30:
     print(num)
     num = num+2
```

2.
```
num = 1
while num<=30:
     print(num)
     num = num+2
```

3.
```
table = 5
while table<=50:
      print(table)
      table=table+5
```

4.
```
i = 1
sum=0
while i<=10:
    sum=sum+i
print(sum)
```

5.
```
n = 2
sum = 0
while n<=20:
    sum =sum +n
    n+=2
print(sum)
```

6.
```
n = 1
sum = 0
while n<=20:
    sum =sum +n
    n+=2
print(sum)
```

7.
```
number = -10
while number<0:
    print(number)
    number+=1
```

8.
```
sum = 0
i = 10
while i>0:
    sum = sum + i
    i=i-1
print(sum)
```

9.
```
sum = 0
num = 12345
while num>0:
    rem = num%10
    sum = sum + rem
    num = num//10
print(sum)
```

==

for loop -> It is used to execute block of statement till the condition is
                  true.

range( ) function - It returns the sequence of numbers which starts from 0 by
default,
                    increment it by 1 by default and stops just before the
specified
number.

Example -

```
1. for i in range(5):
       print(i,end=' ')


2. for i in range(5,10):
       print(i,end=' ')


3. for i in range(1,10,2):
       print(i,end=' ')
```

Assignment Questions

1. Print a table of 7 using for loop
2. Print numbers from 10 to 0
3. Print numbers from -10 to 10
4. print square of each number from 1 to 10
5. print square of all even number between 1 to 10
6. print cube of all odd number between 1 to 20
7. print sum of numbers from 1 to 20
8. print sum of square of each number from 1 to 10

Assignment Solutions

1.
```
for num in range(7,71,7):
    print(num)
```

2.
```
for i in range(10,0,-1):
    print(i)
```

3.

```
for i in range(-10,0):
    print(i)

4.
for n in range(1,11):
    print(n*n)

5.
for i in range(2,11,2):
    print(i**2)

6.
for n in range(1,20,2):
    print(n**3)

7.
sum = 0
for n in range(1,21):
    sum = sum+n
print(sum)

8.
sum = 0
for i in range(1,11):
    sum = sum + i**2
print(sum)
```

===

```
***list in python*** -> List is used to store multiple values at the same time in
                        single variable.

List is created by putting all the elements inside [] and are seperated by ,
List can have any type of value integer,float,string or character.
List is mutable.

Example -

1.
li = [3,5,8,2,9]
for val in li:
    print(val)

2.
list = [8,'a',55.3,"Python",9]
for x in list:
    print(x)

3.
list = [7,2,9]
list.append(60)
print(list)

length = len(list)
print(length)

list.sort()
print(list)
```

```
list.pop()
print(list)
```

Assignment Questions

1. create a list and print it using for loop
2. find sum of all elements of list
3. find product of all element of list
4. find greatest element in list
5. find smallest element in list
6. find square of each number in list
7. find sum of cube of all element in list
8. find sum of square of each element in list
9. sort all element in list

Assignment Solutions

1.
```
li = [ 1,3,'s',"welcome"]
for x in li:
    print(x)
```

2.
```
li = [1,2,4,5,6]
sum = 0
for x in li:
    sum = sum + x
print(sum)
```

3.
```
li = [1,2,4,5,6]
product = 1
for x in li:
    product *= x
print(product)
```

4.
```
li = [8,2,6,3,9]
mx = max(li)
print(mx)
```

5.
```
li = [8,2,6,3,9]
```

```
mn = min(li)
print(mn)
```

6.
```
li = [1,2,3,4,5]
for x in li:
    print(x*x)
```

7.
```
li = [1,2,3,4,5]
sum = 0
for x in li:
    sum = sum + x**3
print(sum)
```

8.
```
li = [1,2,3,4,5]
sum = 0
for x in li:
    sum = sum + x**2
print(sum)
```

9.
```
li = [75,28,13,99,123]
li.sort()
print(li)
```

==

Tuple -> It is used to store multiple values in a single variable. It is similar to list but it is immutable.

Elements in the tuple can be accessed by using variable name and [position]

Example -

```
tup = (2,4,5,"python")
print(tup)

tup = (4,"hello",'a')
print(tup[0],tup[1],tup[2])
```

Assignment Questions

1. create a tuple and print it using for loop
2. find sum of all elements of tuple
3. find product of all element of tuple
4. find greatest element in tuple
5. find smallest element in tuple
6. find square of each number in tuple
7. find sum of cube of all element in tuple

Assignment solutions

1.
```
tup = (4,"hello",'a')
for x in tup:
    print(x)
```

2.
```
tup = (4,7,34,63)
sum = 0
for x in tup:
    sum+=x
print(sum)
```

3.
```
tup = (4,7,34,63)
pro = 0
for x in tup:
    pro*=x
print(pro)
```

4.
```
tup = (4,7,34,63)
print(max(tup))
```

5.
```
tup = (4,7,34,63)
print(min(tup))
```

6.
```
tup = (1,2,3,4,5)
for x in tup:
    print(x*x)
```

7.
```
tup = (1,2,3,4,5)
for x in tup:
    print(x**3)
```

==

Dictionary -> It is a collection of key:values pair. In dictionary every keys must be unique.
Dictionary can be created by placing a sequence of elements within curly {} braces,

separated by ‘comma’.

Example -
1.
```
Dict = {1: 'Rohit', 2: 'Rahul', 3: 'Ravi'}
print(Dict)
```

2.
```
Dict = {1: 'Rohit', 2: 'Rahul', 3: 'Ravi'}
print(type(Dict))
```

```
3.
Dict = {1: 'Rohit', 2: 'Rahul', 3: 'Ravi'}
print(Dict[1])

4.
Dict = {1: 'Rohit', 2: 'Rahul', 3: 'Ravi'}
for key,values in Dict.items():
    print(key,values)
```

Assignment Questions

1. Create a grocery dictionary and use price of values(print in different line)
2. Create a dictionary contain subject name and marks(print in same line)
3. Dictionary for name of student and their roll number(print it)
4. Add new element to the dictionary and print updated dictionary
5. Delete element from dicionary and print updated dictionary
6. Find to total of each subject marks from dictionary.

Assignment Solutions

```
1.
dic = {"Maggi":30,"Soap":50,"VegOil":150,"Rice":200}
for x,y in dic.items():
    print(x,y)

2.
dic = {"English":69,"Physics":89,"Maths":94,"Hindi":73}
for x,y in dic.items():
    print(x,y,end=" | ")

3.
dic = {"Shubham":9,"Harshit":5,"Rishabh":4,"Neelesh":1}

print(dic)

4.
dic = {"Shubham":9,"Harshit":5,"Rishabh":4,"Neelesh":7}

dic["Ayush"]=1
dic["Dheeraj"]=2
print(dic)

5.
dic = {"Shubham":9,"Harshit":5,"Rishabh":4,"Neelesh":7}

dic["Ayush"]=1
dic["Dheeraj"]=2

del dic["Ayush"]
del dic["Shubham"]
print(dic)
```

6.
```
dic = {"English":69,"Physics":89,"Maths":94,"Hindi":73}
total=0
for x in dic.values():
    total+=x
print(total)
```

==

Function in Python

Function are the collection of statments which are grouped to perform any task.
Functions run only when they are called.

Why we use function?
1. Function are used for reusability of a code.
2. To reduce complexity of a program.
3. To increase the redability of a code.

Example -

1.
```
def fun():
    print("Hello World")

fun()
```

2.
```
def fun(a):
    return a

print(fun("Python"))
```

Assignment Questions

1. Write a function to calculate the addition of two numbers.
2. Write a function to print if a number is even or odd.
3. Write a function to print weather the number is posititve or negative.
4. function to check he/she is elegible to vote.
5. function to take 2 parameters and find average.
6. function to find sum of numbers from 1 to the given number
7. Write a function to return a factorial of given number.

Assignment Solutions

1.
```
def fun(a,b):
    return a+b

print(fun(73,27))

2.
def ev_od(num):
    if num%2 is 0:
        print("Number is even")
    else:
        print("Number is odd")

ev_od(83)

3.
def neg_pos(num):
    if num<0:
        print("Number is Negative")
    else:
        print("Number is Positive")

neg_pos(83)

4.
def vote(age):
    if age>=18:
        print("Vote")
    else:
        print("Cannot vote")

vote(32)

5.
def avg(x,y):
    return (x+y)/2

print(avg(49,51))

6.
def sum(num):
    sm=0
    for i in range(1,num+1):
        sm+=i
    return sm
print(sum(10))

7.
def fact(num):
    fa=1
    for i in range(num,0,-1):
        fa*=i
    return fa
print(fact(5))
```

===

Lambda Function in Python -> A lambda function is a small anonymous function.
A lambda function can take any number of arguments, but can only have one
expression.

Example -

1.
```
x = lambda a : a + 10
print(x(5))
```

2.
```
y=lambda x,y,z:x+y+z
print(y(3,4,5))
```

3.
```
sum = lambda x:x+x
li = [1,2,3,4,5]
for x in li:
    print(sum(x))
```

Assignment Questions

1. Lambda function that add 7 to given number
2. Lambda function that multiply 7 to given number
3. Lambda function to find double of all element of list
4. Lambda function to find product of each with itself in list
5. Lambda function to find sum of square of 3 numbers
6. Lambda function to find product of cube of 3 numbers

Assignment Solution

1.
```
y=lambda x:x+7
print(y(7))
```

2.
```
y=lambda x:x*7
print(y(7))
```

3.
```
sum = lambda x:x+x
li = [1,2,3,4,5]
for x in li:
    print(sum(x))
```

4.
```
sum = lambda x:x*x
li = [1,2,3,4,5]
for x in li:
    print(sum(x))
```

5.
```
sum = lambda x,y,z:x*x+y*y+z*z
print(sum(1,2,3))
```

6.

```
a = lambda x,y,z:(x**3)*(y**3)*(z**3)
print(a(1,2,3))
```

==

String in Python -> String is a collection of many alphabets. It is written inside single quotes or double quotes.

Square bracket can be used to access element of string.

Example -

1.
```
s1 = "Hello"
s2 = 'World'
print(s1,s2)
```

2.
```
s1 = "Hello"
s2 = 'World'
for x in s1:
    print(x)
print(s2[0],s2[1])
```

String Slicing

3.
```
s1 = "Hello World"
print(s1[0:5])
```

String function

4.
```
s1 = "hello world"
print(len(s1))
print(s1.isalpha())
print(s1.isalnum())
print(s1.islower())
print(s1.count('l'))
```

5.
```
s1 = "hello world"
print(s1.upper())
print(s1.lower())
print(s1.split())
print(" ".join(s1))
```

***Assignment Questions**

"Hello, How are You"

1. Print given string using for loop
2. Print using slicing
3. Print How using slicing
4. Find length of given string
5. Find Count of H and l

6. Print given string in upper case
7. Print given string in lower case
8. Join given string using '.'

Assignment Solution

```
s = "Hello, How are You"
for x in s:
    print(x,end="")
print()
print(s[0:])
print(s[7:10])
print(len(s))
print(s.count('H'))
print(s.count('l'))
print(s.upper())
print(s.lower())
print(".".join(s))
```

===

OOP in Python -> This is called as Object-Oriented Programming.
                       The concept of OOP in Python focuses on creating reusable
code.
This concept is also known as DRY (Don't Repeat Yourself).

In Python, the concept of OOP follows some basic principles:

++

Class -> A class is a blueprint for the object. We can define class by using
             "class".

Example ->

```
class Student:
    pass
```

++
+

Object -> An object is instance of class. Means when class is defined only
              description of object is defined, not object itself.

```
rahul = Student()
```

++
+

Constructor -> The task of constructors is to assign values to the data members
of
                   the class when an object of the class is created.

Syntax ->

```
def __init__(self):
    # body of the constructor
```

Example ->

```
class Student:
    def __init__(self,n,s):
        self.name = n
        self.std = s

rahul = Student("Rahul","5th")
ravi = Student("Ravi","7th")
print(ravi.name, ravi.std)
```

Assignment Questions

1. Create a Vehicle class with max_speed and mileage attributes and print
2. Create a empty vehical class
3. Create math class with methods sum,sub,pro,div
4. Create student class with methods getPercent and find percent of objects
5. Create Employee class with salary atribute and print salary of 3 objects

Assignment Solution

1.
```
class Vehicle:
    def __init__(self, max_speed, mileage):
        self.max_speed = max_speed
        self.mileage = mileage

modelX = Vehicle(240, 18)
print(modelX.max_speed, modelX.mileage)
```

2.
```
class Vehical:
    pass
```

3.
```
class Math:
    def __init__(self,a,b):
        self.a = a
        self.b = b

    def sum(self):
        return self.a+self.b
    def sub(self):
        return self.a-self.b
    def pro(self):
        return self.a*self.b

calculation = Math(10,6)
```

```
print(calculation.sum())

4.
class Student:
    def __init__(self,m,e,s):
        self.math = m
        self.eng = e
        self.sci = s
    def getPercent(self):
        return (self.math+self.eng+self.sci)/3

Rohan = Student(70,70,60)
print(Rohan.getPercent())

5.
class Employee:
    def __init__(self,salary):
        self.salary = salary

    def getSalary(self):
        return self.salary

Rohan = Employee(50000)
print(Rohan.getSalary())
Mayank = Employee(30000)
print(Mayank.getSalary())
```

+++
+

Method -> Methods are functions defined inside the body of a class.

Example ->

```
class Math:
    def __init__(self,a,b):
        self.first = a
        self.second =b

    def add(self):
        print(self.first+self.second)

    def sub(self):
        return self.first-self.second

first = Math(7,7)
first.add()

second = Math(100,3)
print(second.sub())
```

+++
+

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

Example ->

```
class Animal:

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

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

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

# child class
class Fish(Animal):

    def __init__(self):
        # call constructor of base class
        Animal.__init__(self)
        print("Fish is ready")

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

F = Fish()
F.whoisThis()
# A = Animal()
# A.whoisThis()
# F.swim()
```

+++
+

Encapsulation -> Encapsulation is a techniqe to prevent data from direct
modification.
To prevent data from changing outside the class we need to create
protected variable.

Example ->

```
class Base:
    def __init__(self):

        # Protected member
        self._a = 2
        self._b = 3

class Derived(Base):
    def __init__(self):

        Base.__init__(self)
        print("Calling protected member of base class: ")
        print(self._a)
        print(self._b)

obj1 = Derived()

obj2 = Base()
```

```
print(obj2.a)
```

+++
+

Arrays in Python -> An array is a collection of same type of data items stored at
                        contiguous memory locations.

Array in Python can be created by importing array module.

module.array(data_type, value_list) is used to create array in python.

Example ->
1.
```
import array as arr
a = arr.array('i', [2, 4, 6, 8])
print("First element:", a[0])
print("Second element:", a[1])
print("Second last element:", a[-1])
```

2.
```
import array as arr
numbers = arr.array('i', [1, 2, 3, 5, 7, 10])

# changing first element
numbers[0] = 0
print(numbers)

# changing 3rd to 5th element
numbers[2:5] = arr.array('i', [4, 6, 8])
print(numbers)
```

Some useful method of array

1. append()
2. insert()
3. pop()
4. remove()
5. index()
6. reverse()

Example ->

```
import array as arr
num = arr.array('i', [1, 2, 3])

num.append(9)
print(num)

num.insert(1,3)
print(num)

num.pop()
print(num)
```

```
num.remove(3)
print(num)

print(num.index(1))

num.reverse()
print(num)
```

Assignment Question

1. Create array of 5 numbers and print them in different line.
2. Create array of 5 element and add elements from index 1 to 3
3. Create array of 10 float values and print in different line.
4. Create array and remove last element from it
5. Create array and reverse all elements of that array

Assignment Solution

1.
```
import array as arr

num = arr.array('i',[1,2,3,4])
for x in num:
    print(x)
```

2.
```
import array as arr

num = arr.array('i',[1,2,3,4,5])

num[1:4]=arr.array('i',[9,8,7])
print(num)
```

4.
```
import array as arr

num = arr.array('i',[1,2,3,4,5])

num.pop()
print(num)
```

5.
```
import array as arr

num = arr.array('i',[1,2,3,4,5])
```

```
num.reverse()
print(num)
```

==
=

Exception Handling -> Try and except statements are used to catch and handle
                          exceptions in Python. Statements that can raise
exceptions
are kept inside the try and the statements that handle the exception are written
inside except.

Example ->

1.
```
li = [1,2,3,4]
try:
    print(li[3])
    print(li[4])
except:
    print("Invalid Index used")
```

2.
```
li = [1,2,3,4]
try:
    print(li[3])
    #print(li[4])
    #print(l[1])
    #print(li[2]/0)

except IndexError:
    print("Invalid Index used")

except NameError:
    print("Invalid name used")

except ZeroDivisionError:
    print("Cannot be divided by 0")
```

==
=

Main function in Pyhton -> It is like a execution starting point of a program.

Example ->

1.
```
def fun():
    print("Hello World")

if __name__ == "__main__":
    fun()
```

2.
```
def add(a,b):
    return a+b
```

```
def sub(a,b):
    return a-b

def pro(a,b):
    return a*b

if __name__ == "__main__":
    print(add(83,17))
    print(pro(158,10))


==================================================================================
=
```