Universidad Autónoma del Estado de Hidalgo
Instituto de Ciencias Básicas e Ingeniería
Academia de Matemáticas y Física
Programming

Palomares Maldonado Héctor Miguel

Learn to program in Python

Índice

1. Input and output data

create a file with extension *.py* in this document it's going to write for the interpreter of *python* show the result
Example

primerprograma.py

For show in screen, you are going to write:

```
print "Esto  se muestra en la patalla"
```

usted verá

```
Esto  se muestra en la patalla
```

Store a data using keyboard

```
print "Enrtrada de datos "
var = input("digite el valor de la varible var : ")
print "la varible  es : ", var
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python prog.py
Enrtrada de datos
digite el valor de la varible var : 24
la varible  es :  24
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python prog.py
Enrtrada de datos
digite el valor de la varible var : "Variable_creada_desde_el_teclado"
la varible  es :  Variable_creada_desde_el_teclado
```

2. Type of Variables

In Python we have type data, integer, floating(decimals), complexes, text string and boolean values: true and false

We are going create a pair of variables by way of **Example :**

```
ent = 10
real = 3.1416
cadena = "hola soy hector"
boolean = True
print "Variable tipo entero = ",ent
print "Variable tipo decimal = ",real
print "Variable tipo cadena = ",cadena
print "Variable tipo boolean =",boolean
```

```
Variable tipo  <type 'int'>   =  10
Variable tipo  <type 'float'>   =  3.1416
Variable tipo cadena   <type 'str'> =  hola soy un programador
Variable tipo boolean   <type 'bool'>  = True
```

As you can see in Python, unlike many other languages, the type of variables is not going to declare. In java we write

```
String Str= "Hello World";\\
int integer= 23;
```

To know what type of variable we write: tpye(name of the variable). how show in the example

3. Arithmetic operators

The numerical values are also the result of a series arithmetic and matematical operators. the syntax of the basic operators in Python are show in the table 1

+	Sum
−	subtraction
*	Multiplication
/	Division
**	Exponent o pow
//	Division integer
%	Modulo

Cuadro 1: Table of Arithmetic Operators

The modulo operator us give the residual of some division, for example, if we divide 7/2 the result is 3 and the modulo is 1 The whole division return the whole part, so it has decimal numbers, rounds the result
Example :

```
# operadores aritmaticos
sumar = 20 + 10 + 10 + 3
resta = 50 - 25
mult = 120 * 4
dividir = 27 / 4
residuo = 27 %4
expo = 5  ** 3
print "La suma es :", sumar
print "La resta es: ",resta
print "La multiplicacion es", mult
print "La divicion es ",dividir
print "5^3 =", expo
print "El residuo (%) es",residuo
```

```
# Calcular la superficie de un triangulo
base = 30
altura = 50
superficie = base * altura / 2
print "El area de un triangulo de altura = 50 y
base = 30 es igual a ", superficie
print "Python es una calculadora "
print "Sumando 5+5=",5+5
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python Prog4.py
La suma es : 43
La resta es:  25
La multiplicacion es 480
La divicion es  6
5^3 = 125
El residuo (%) es 3
El area de un triangulo de altura = 50 y base = 30 es igual a  750
Python es una calculadora
Sumando 5+5= 10
```

```
#operadores
print "10 == 12", 10 == 12
print "12 == 12",12 == 12
print "10 != 10",10 != 10
print "10 != 9",10 != 9
print "azul != Azul","azul" != "Azul"
print "rojo != rojo", "rojo" != "rojo"
print "2 < 3", 2 < 3
print "4 < 3", 4 < 3
print "50 <= 50", 50 <= 50
print "10 <= 9", 10 <= 9
lista_uno = [1,2,3]
lista_dos = [1,2,5]
print "[1,2,3] = [1,2,5]",lista_uno == lista_dos
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python Prog8.py
10 == 12 False
12 == 12 True
10 != 10 False
10 != 9 True
azul != Azul True
rojo != rojo False
2 < 3 True
4 < 3 False
50 <= 50 True
10 <= 9 False
[1,2,3] = [1,2,5] False
```

4. cycles

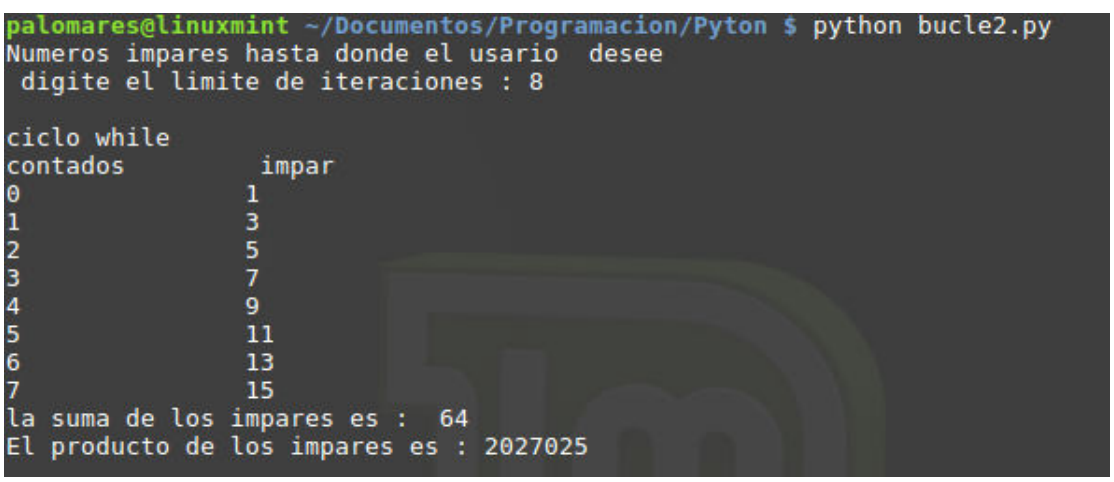
4.1. Cycle For

The sentence *for* of Python iterate on items of any sequence (a list or a string) it has an order.
the structure of the cycle is the following

```
for accountant in range(stard, condition, increase/interation):
    .
    .
    instructions
    .
    .
```

Note : the intructions of cicle will always have a space after the instruction, if the linear has not space, it is not considere in the cycle
Example

```
print "Numeros impares hasta donde el usario  desee"
n = input(" digite el limite de iteraciones : ")
print "\nciclo while \ncontados \t impar"
contador=0
suma=0
producto=1
for contador in range(0,n):
    impar = 2*contador+1
    print contador,"\t\t",impar
    producto=producto*impar
    suma=suma+impar
    contador=contador+1
print "la suma de los impares es : ", suma, "\nEl producto de los impares
es :",producto
```



4.2. Cycle while

In Python has a reserved word name *while* that allows us to make a cycle, these are periodic sequences
The cycle is base on a condition or logical secuence
The struct of cycle *while* is the following
Example:

```
stard. (It could have stard)
while (condition):
    .
    .
    instructions
    .
    .
    increase/interaction
```

Note : As in the cycle *for* the intructions it will have a space after of *while*
Example:

```
print "Numeros impares hasta donde el usario  desee"
n = input(" digite el limite de iteraciones : ")
print "\nciclo while \ncontados \t impar"
contador=0
suma=0
producto=1
while(contador<=n ):
    impar = 2*contador+1
    print contador,"\t\t",impar
    producto=producto*impar
    suma=suma+impar
    contador=contador+1

print "la suma de los impares es : ", suma, "\nEl producto de los impares
es :",producto
print "\n\n"
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python bucle1.py
Numeros impares hasta donde el usario  desee
 digite el limite de iteraciones : 7

ciclo while
contados         impar
0               1
1               3
2               5
3               7
4               9
5               11
6               13
7               15
la suma de los impares es :  64
El producto de los impares es : 2027025
```

Example:

```
1  print "Numeros pares hasta donde el usario  desee (disminuyendo)"
2  n = input(" digite el limite de iteraciones : ")
3  print "\nciclo while \ncontados \t impar"
4  contador=n
5  suma=0
6  producto=1
7  while(contador>0 ):
8      par = 2*contador
9      print contador,"\t\t",par
10     producto=producto*par
11     suma=suma+par
12     contador=contador-1
13 print "la suma de los pares es : ", suma, "\nEl producto de los
14 pares es :",producto
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python bucle3.py
Numeros pares hasta donde el usario  desee (disminuyendo)
 digite el limite de iteraciones : 8

ciclo while
contados         impar
8               16
7               14
6               12
5               10
4               8
3               6
2               4
1               2
la suma de los pares es :  72
El producto de los pares es : 10321920
```

5. Arrays

5.1. List

The list in Python is variables that going to store a variable, is as if going to have small boxes in boxes. it could diferent datas or variables

Example

```
lista = ["Hector", "Miguel", "Palomares"]
print "Mi tupla es : ", mi_tupla
print "Posicion 3 de la tupla izquida a derecha ",mi_tupla[3]
print "Posicion -1 ", mi_tupla[-1]
print "Posicion 4", mi_tupla[-4]
print "Las posiciones del 1 al 3 son ",mi_tupla[1:3] #devulve (2, 3.1)
print "[:3] imprime desde la posicion 0 hasta la 2... ",mi_tupla[:3]
print "[1:] imprime desde la posicion 1 hasta el ultimo ",mi_tupla[1:]
copiarLista = mi_tupla[2:4]
print "copiar las posiciones de la mi_tupla a copiarLista y las posiciones
son [2:4]",copiarLista
print "imprimir lista mediante un Ciclo For\nLista es: ", lista
for nombre in lista:
    print "El nombre es: ", nombre
```

This program was use a function name *for* that is a cycle, and it was explain in the previous section

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python Prog5.py
Mi tupla es :  ('primero', 2, 3.1, 'cuarto')
Posicion 3 de la tupla izquida a derecha  cuarto
Posicion -1  cuarto
Posicion 4 primero
Las posiciones del 1 al 3 son  (2, 3.1)
[:3] imprime desde la posicion 0 hasta la 2...  ('primero', 2, 3.1)
[1:] imprime desde la posicion 1 hasta el ultimo  (2, 3.1, 'cuarto')
copiar las posiciones de la mi_tupla a copiarLista y las posiciones son [2:4] (3.1, 'cuarto')
imprimir lista mediante un Ciclo For
Lista es:  ['Hector', 'Miguel', 'Palomares']
El nombre es:  Hector
El nombre es:  Miguel
El nombre es:  Palomares
```

Example

```
mi_lista = ["numero cero", 20, 10.11, True, [1, 2, 3, 4, 5]]
print "mi_lista es : ",mi_lista
print "mi_lista[3]=",mi_lista[3]
print "mi_lista[-1]=",mi_lista[-1]
print "se cambiara el elemeto mi_lista[1]=20 por False"
mi_lista[1] = False
print "Se muestra la mi_lista Modificada : ",mi_lista
print "mi-lista[4] = ",mi_lista[4]
print "mi_lista tiene ",len(mi_lista),"elemetos"
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python Prog6.py
mi_lista es :  ['numero cero', 20, 10.11, True, [1, 2, 3, 4, 5]]
mi_lista[3]= True
mi_lista[-1]= [1, 2, 3, 4, 5]
se cambiara el elemeto mi_lista[1]=20 por False
Se muestra la mi_lista Modificada :  ['numero cero', False, 10.11, True, [1, 2, 3, 4, 5]]
mi-lista[4] =  [1, 2, 3, 4, 5]
mi lista tiene  5 elemetos
```

5.2. Tuplas

The tuplas worked equal that the list with exception are arrays one-unidimensional and it can not be modified
Example :

```
tupla=(3,2.1416, "cadena", 24, "segunda_cadena", 3.1,  10, 20, "tercera cadena")
print "esta es una tupla : ", tupla
print "tupla[1] = ",tupla[1]
print "tupla[-1] = ",tupla[-1]
print "tupla[:3] = ",tupla[:3]
print "tupla[3:] = ",tupla[3:]
Copiar=tupla
print "Copiar = ",Copiar
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python tupla.py
esta es una tupla :  (3, 2.1416, 'cadena', 24, 'segunda_cadena', 3.1, 10, 20, 'tercera cadena')
tupla[1] =  2.1416
tupla[-1] =  tercera cadena
tupla[:3] =  (3, 2.1416, 'cadena')
tupla[3:] =  (24, 'segunda_cadena', 3.1, 10, 20, 'tercera cadena')
Copiar =  (3, 2.1416, 'cadena', 24, 'segunda_cadena', 3.1, 10, 20, 'tercera cadena')
```

5.3. Diccionarios

The dictionary, that going to define a relation one an one between *keys* and *values* **Example :**

```
dic={'nom':"Hector",'ed':23, 'estat':1.76, 'nac':"mexicano", 5:12.6 }
#crea un diccionario
print "El diccionario es\n",dic
dic['Ciudad']="pachuca" #agrega este elemento
valor=dic.get('ocupacion', "No exite este  elemento") #busca el elemeto "o1cupacion"
print "valor = ",valor
valor1=dic.get('nom', "Encontrado..!") #busca el elemeto "nom"
print "valor1 = ",valor1, "\nel diccionario es :", dic
del dic[5] # elimina el elemento llamado 5
print "el diccionario modificado es :", dic
llaves=tuple(dic.keys())
valores=tuple(dic.values())
print"imrimir los nombres de las variables\n",llaves
print"imrimir los valores de las variables\n",valores
```

```
dic2={'ci':130, 'ocupaciom':"estudiante"}
print "imrimir dic2\n",dic2
dic.update(dic2) # los elementos del dicionario 2 ahora esta en el diccionario 1
print "imrimir dic\n",dic
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python diccionario.py
El diccionario es
{'nom': 'Hector', 'estat': 1.76, 'nac': 'mexicano', 5: 12.6, 'ed': 23}
valor =  No exite este  elemento
valor1 =  Hector
el diccionario es : {'nom': 'Hector', 5: 12.6, 'nac': 'mexicano', 'ed': 23, 'estat': 1.76, 'Ciudad': 'pachuca'}
el diccionario modificado es : {'nom': 'Hector', 'nac': 'mexicano', 'ed': 23, 'estat': 1.76, 'Ciudad': 'pachuca'}
imrimir los nombres de las variables
('nom', 'nac', 'ed', 'estat', 'Ciudad')
imrimir los valores de las variables
('Hector', 'mexicano', 23, 1.76, 'pachuca')
imrimir dic2
{'ci': 130, 'ocupaciom': 'estudiante'}
imrimir dic
{'nom': 'Hector', 'ci': 130, 'nac': 'mexicano', 'ed': 23, 'ocupaciom': 'estudiante', 'estat': 1.76, 'Ciudad': 'pachuca'}
```

6. Functions

A function is a set of lines that makes specific work and it could return a values. The fuctions could take parameters that modify their work, are use to break down big problems in work simple and for don't repeat lines of code that could be use more than once. and is name for other function for perform its work

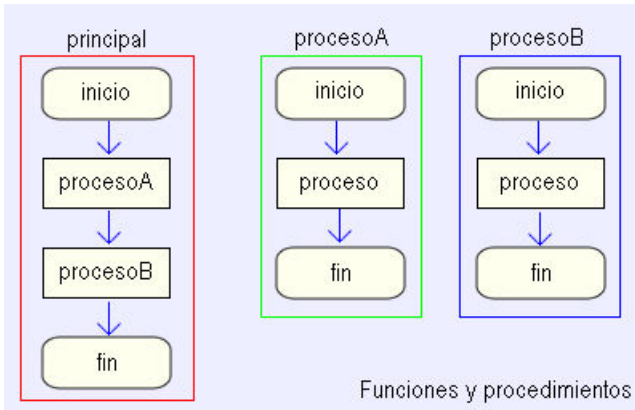


Figura 1: Diagram of a function.

The struct of function is the following, for going to create use the reserve word *def*

```
def name_of_the_function( parameters )
    .
    .
    instructions
    .
    .
    return/yeild
```

Note. The function could has parameters and return.
Parameters : It is data that going to pass of function an other
Return: Is of result or variables that was get in a function and this is going to use in other function

Example 1

```
def suma(num1, num2 ):
    resultado = num1 + num2
    return resultado
print "Funcion Suma con retorno y con parametros "
a = input("digite un numero : ")
b = input("digite un numero : ")
r=suma(a,b)
print "la suma es : ", r
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python funcion1.py
Funcion Suma con retorno y con parametros
digite un numero : 64
digite un numero : 36
la suma es :  100
```

Note that in the function *Sum* had two parameters call *num1* and *num2* this are sum and save in the result variable that is return, this variable could be call in other part of the program , in a few words *return* a result where the function was called
The program asks, you to enter two values a that takes place of the variable *num1* and b takes place of *num2*, in other part program it is called the function, the parameters write order same for example if we would write **sum (b, a)**, b = *num1* and a = *num2*, the variable r take the value of the variable *result*

Example 2

```
def multiplicacion():
    print "Funcion multiplicacion  de dos numeros sin retorno
     y sin parametros "
    num1 = input("digite el primer numero numero : ")
    num2 = input("digite el segundo numero : ")
    resultado = num1 * num2
    print "la multiplicacion es : ", resultado
multiplicacion()
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python funcion2.py
Funcion multiplicacion  de dos numeros sin retorno y sin parametros
digite el primer numero numero : 6
digite el segundo numero : 12
la multiplicacion es :  72
```

This is a program in which the function has not parameters or return, it work in the function and going to call for other function, so run

Example 3

```
def factorial(numero):
    fact=1
    while numero > 0 :
        fact*=numero;
        numero-=1
    return fact
result=factorial(3)
print "factorial de 3 =",result
resulta=factorial(4)
print "factorial de 4 =",resulta
resulta=factorial(5)
print "factorial de 5 =",resulta
resulta=factorial(6)
print "factorial de 6 =",resulta
resulta=factorial(7)
```

```
print "factorial de 7 =",resulta
print "\n\n\n\n"
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python funcion3.py
factorial de 3 = 6
factorial de 4 = 24
factorial de 5 = 120
factorial de 6 = 720
factorial de 7 = 5040
```

This function is like that of example 1. it has only a parameter and this is define in the parentheses, this program is a example as a function can be use repeatedly to get different results, which means that we do not write several times the procedure for each number that you want to calculate the factorial

Example 4

```
def factorial(numero):
    fact=1
    while numero > 0 :
        fact*=numero;
        numero-=1
    print fact
print "factorial de 3 ="
factorial(3)
print "factorial de 4 ="
factorial(4)
print "factorial de 5 ="
factorial(5)
print "factorial de 6 ="
factorial(6)
print "factorial de 7 ="
factorial(7)
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python funcion4.py
factorial de 3 =
6
factorial de 4 =
24
factorial de 5 =
120
factorial de 6 =
720
factorial de 7 =
5040
```

The difference of this program from other is that it does not return
any value, show the result from the function, it only show factorial number of the parameter a new line of the console

Example 5

```
def funcion(**kwargs):
    print(kwargs)
print "\n"
resultado=funcion(valor='Palomares',x=2,y=9,z=True)
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python Funcon_nArgs.py

{'y': 9, 'x': 2, 'z': True, 'valor': 'Palomares'}
```

Note: When we put ** in priciple of the parameters. we pass n parameters in the function. as see in the example, usually the word *kargs* of k-arguments

Example 6

```
def generador(*args):  ##Generador es una funcion y #args = n parametros
    for valor in args:
        yield valor*10  #return
print "Programa funcion generador "
print "Multiiplica los numeros del argumento por 10 ya que yield=valor*10\n"
for valor in generador(1,2,3,4,5,6,7,8,9):
    print "\t",valor
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python generador1.py
Programa funcion generador
Multiiplica los numeros del argumento por 10 ya que yield=valor*10

        10
        20
        30
        40
        50
        60
        70
        80
        90
```

Nota: The generator is a function. and do the same.
Nota: The reserved word *yield* work like *return*

Example 7

```
import random
print "10 numeros aleatorios del 0 al 10\n"
for x in xrange(1,11):
    valor = random.randint(0,10)
    print "numeros",x,"=",valor

lista = [True, "Strings", 1,2,3,"Palomares", False]
aleatorio=random.choice(lista)
print "\n\nLa lista es", lista
print "\n\nSe escogera un elemento de la lista",lista,"\n
```

```
Aleatoriamente y el elemento es : ",aleatorio
random.shuffle(lista)
print "\n\ndesordenar lista....\nLa lista desordenada es:",lista
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python librerias.py
10 numeros aleatorios del 0 al 10

numeros 1 = 5
numeros 2 = 4
numeros 3 = 9
numeros 4 = 1
numeros 5 = 3
numeros 6 = 8
numeros 7 = 4
numeros 8 = 2
numeros 9 = 8
numeros 10 = 8


La lista es [True, 'Strings', 1, 2, 3, 'Palomares', False]


Se escogera un elemento de la lista [True, 'Strings', 1, 2, 3, 'Palomares', False]
Aleatoriamente y el elemento es :  Palomares


desordenar lista....
La lista desordenada es: ['Strings', True, 1, 'Palomares', 3, False, 2]
```

Nota: *import* as its name says it matters functions that it has established *Python*
Nota: radom generates alatorios numbers, example emph random.randomint (start, end) generates an integer where what is in paranthesis is the range that the number can be

7. Modulos

Example 1

```
def suma(num1, num2):
    print "la suma de",num1,"+",num2
    return num1 + num2
def resta(num1, num2):
    print "la resta de",num1,"-",num2
    return num1 - num2
def multiplicacion(num1, num2):
    print "la multiplicacion de",num1,"*",num2
    return num1 * num2
def division(num1, num2):
    print "la division de",num1,"/",num2
    return num1 / num2
```

Example 1.1

```
import Modulo_operacion
resultado = Modulo_operacion.suma(30,45)
```

```
print resultado
resultado = Modulo_operacion.resta(30,45)
print resultado
resultado = Modulo_operacion.multiplicacion(11,12)
print resultado
resultado = Modulo_operacion.division(45,5)
print resultado
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python Modulo_main.py
la suma de 30 + 45
75
la resta de 30 - 45
-15
la multiplicacion de 11 * 12
132
la division de 45 / 5
9
```

Example 1.2

```
from Modulo_operacion import suma
from Modulo_operacion import resta as r
from Modulo_operacion import multiplicacion as m
from Modulo_operacion import division as div
resultado = suma(30,45)
print resultado
resultado = r(30,45)
print resultado
resultado = m(11,12)
print resultado
resultado = div(45,5)
print resultado
```

```
palomares@linuxmint ~/Documentos/Programacion/Pyton $ python Modulo_mainOne.py
la suma de 30 + 45
75
la resta de 30 - 45
-15
la multiplicacion de 11 * 12
132
la division de 45 / 5
9
```