Unidad 4

Manejo de arreglos y funciones en Python

Calculadora

Crear una calculadora sencilla en Python, que permita escoger una de las cuatro operación aritméticas básicas e imprimir el resultado. En menú debe ser como se muestra a continuación.

1. Suma
2. Resta
3. Multiplicación
4. División
5. Salir

NOTA: Solo se puede salir si el usuario ingresa la opción **5. Salir**

Calculadora

```
while True:
    resultado = "No hay resultado disponible"
     print ("Seleccione una de las siguientes opciones\n1. Suma\n2. Resta\n3. Multiplicación\n4. División\n5. Salir")
    op = input()
    print("\n")
    if op == 1:
        num1 = input("Ingrese el 1er numero: ")
        num2 = input("Ingrese el 2do numero: ")
        resultado = num1 + num2
    elif op == 2:
        num1 = input("Ingrese el 1er numero: ")
        num2 = input("Ingrese el 2do numero: ")
        resultado = num1 - num2
    elif op == 3:
        num1 = input("Ingrese el 1er numero: ")
        num2 = input("Ingrese el 2do numero: ")
        resultado = num1 * num2
    elif op == 4:
        num1 = float(input("Ingrese el numerador: "))
        num2 = float(input("Ingrese el denominador: "))
        if num2 == 0:
            print("La operación no se puede realizar. La división por cero no existe")
        else:
            resultado = num1 / num2
    elif op == 5:
        break
    else:
        print("Selecciono una opción no valida. Intente de nuevo")
        continue
    print(resultado)
    print("---------------------------------")
```

Tupla

La TUPLA es otro tipo de arreglo que dispone el Python.
Se maneja como una lista más cerrada porque no puede cambiar elementos individuales, en otras palabras, es **inmutable**.

```
tupla1 = 1, 2, 3, 4, 5
print tupla1
tupla2 = ('Bioingenieros', 'Udea', '2017', 9, 8, 7)
print tupla2
print tupla1[0]
tupla1[0] = "Hola mundo"
```

```
>>>
(1, 2, 3 ,4, 5)
('Bioingenieros', 'Udea', '2017', 9, 8, 7)
1

TypeError: 'tuple' object does not support item assignment
>>>
```

NOTA: aparece un error de asignación no soportada, es el concepto de **inmutabilidad**.

Tupla

También es posible tener elementos de diferente tipo en las tuplas y de manera natural se pueden generar tuplas multidimensionales de igual forma que en las listas.

```
tupla1 = 1, 2, 3, 4, 5
tupla2 = ('Bioingenieros', 'Udea', '2017', 9, 8, 7)
tupla3 = tupla1,tupla2

print tupla3
print tupla3[1][0]
```

```
>>>
((1, 2, 3, 4, 5), ('Bioingenieros', 'Udea', '2017', 9, 8, 7))
Bioingenieros
>>>
```

Tupla

Los arreglos tipo tupla cuentan con menos métodos que las listas. Solo se usan los métodos **count** e **index**.

```
tupla1 = 1, 5, 2, 3 ,4, 5
tupla2 = ('Bioingenieros', 'Udea', '2017', 9, 'Udea', 8, 7, 'Udea')
print (tupla1.count(5))
print (tupla2.count('Udea'))
print (tupla1.index(5))
tupla2.append('Bio')
print tupla2
```

```
>>>
2
3
1
AttributeError: 'tuple' object has no attribute 'append'
>>>
```

Tupla

La tuplas como son arreglos más cerrados, ya que no permiten cambiar directamente sus elementos, sí tienen un mecanismo de desempacado mediante la asignación de sus elementos a variables independientes para cada elemento.

```
tupla1 = 9,8,7
tupla2 = ('Bioingenieros', 'Udea', '2017')
tupla3 = tupla1,tupla2
print tupla3
X,Y = tupla3
print X
print Y
x,y,z = X
print x
print y
print z
```

```
>>>
((9, 8, 7), ('Bioingenieros', 'Udea', '2017'))
(9, 8, 7)
('Bioingenieros', 'Udea', '2017')
9
8
7
>>>
```

Funciones

Además de utilizar predefinidas como print(), max(), min(), len(), se pueden crear funciones propias utilizando la sentencia **def**.

```
def mi_func():
        print ("Hola")
        print ("Bioingenieros")
        print ("UdeA")

mi_func()
```

```
>>>
Hola
Bioingenieros
UdeA
>>>
```

NOTA: El bloque de código dentro de cada función empieza con dos puntos (:) y está **indentado**

Funciones

Las funciones deben ser definidas antes de llamarlas, al igual que se hace con las variables, que se asignan antes de utilizarlas.

```
mi_func()
```

```
def mi_func():
        print ("Hola")
        print ("Bioingenieros")
        print ("UdeA")
```

```
>>>
NameError: name 'mi_func' is not defined
>>>
```

Argumentos

La mayoría de las funciones reciben argumentos. El argumento en la información que va entre paréntesis.

```
def imprimir_algo(palabra):
    print (palabra + "!!!!")

imprimir_algo("Hola")
imprimir_algo("Bioingenieros")
var = str ( input("Ingrese algo: " ) )
imprimir_algo(var)
```

```
>>>
Hola!!!!
Bioingenieros!!!!
Ingrese algo: 3
3!!!!
>>>
```

Argumentos

¿Cuál es el resultado de este código?.

```
def imprimir_mult(x)
      print(2*x)

imprimir_mult(3)
```

6

Argumentos

Se pueden definir funciones con más argumentos, los cuales deben ir separados por comas (,).

```
def imprimir_suma(a,b):
    print (a+b)

x = input("Ingrese el #1: " )
y = input("Ingrese el #2: " )
imprimir_suma(x,y)
```

```
>>>
Ingrese el #1: 4
Ingrese el #2: 5
9
>>>
```

Argumentos

Rellene los espacios en blanco para definir una función que reciba dos argumentos e imprima su multiplicación

```
def imprimir_mult(x,y):
    print(x* y )
```

Argumentos

Los argumentos pueden ser utilizados como variables dentro de la definición de la función, pero, NO pueden ser referenciados fuera de la definición de la función. Aplica también para variables creadas dentro de la función.

```
def funcion(variable):
        variable+=1
        print (variable)

funcion(7)
print(variable)
```

```
>>>
8

NameError: name 'variable' is not defined
>>>
```

Argumentos

Rellene los espacios en blanco para definir una función que imprima “Si”, si su parámetro es un número par y “No” en caso contrario.

```
def par_o_impar(x):
    if x%2 == 0:
        print (“Si”)
    else:
        print(“No”)
```

Devolución de valores de función

Algunas funciones, tales como **int** o **srt**, devuelven un valor para ser utilizado más adelante. Para hacer esto en las funciones que definamos, se puede utilizar la sentencia **return**.

```
def maximo(x,y):
    if x > y:
        print "El numero mayor es: "
        return x
    elif y > x:
        print "El numero mayor es: "
        return y
    else:
        print "Los numero ingresados son iguales"
        return x,y
print maximo(4,7)
z = maximo(input("#1: "),input("#2: "))
print z
```

```
>>>
El numero mayor es:
7
#1: 4
#2: 4
Los numero ingresados son iguales
(4, 4)
>>>
```

Devolución de valores de función

Rellene los espacios en blanco para definir una función que compare las longitudes de sus argumentos y devuelva el más corto.

```
def str_mas_corto(x,y):
    if len(x) <= len (y):
        return x
    else:
        return y
```

Devolución de valores de función

Una vez que devuelva un valor de una función, esta deja de ser ejecutada inmediatamente.

Cualquier código luego de la sentencia **return** nunca se ejecutará

```
def sumar(x,y):
    total = x + y
    return total
    print "Esto no será impreso"

print sumar(4,5)
```

```
>>>
9
>>>
```

Calculadora

Crear una calculadora sencilla en Python, que permita escoger una de las cuatro operación aritméticas básicas e imprimir el resultado. En menú debe ser como se muestra a continuación.

1. Suma
2. Resta
3. Multiplicación
4. División
5. Salir

Condiciones:

* Solo se puede salir si el usuario ingresa la opción **5. Salir**
* Las operaciones se deben hacer utilizando funciones