

# GUÍA DE TRABAJO Y ACTIVIDADES Nº 1

Cátedra Programación II

Marzo 2024

## 1. *Conociendo un Poquito el Lenguaje Python*

Arranquemos tipeando algunos comandos vistos en clase para acostumbrarnos a la notación del lenguaje y sus sutilezas.

```
1 >>> 2+3
2 >>> 5*7
3 >>> 2+3*7
4 >>> (2+3)*7
5 >>> 10/4
6 >>> 5**2-
7 >>> 53//24
8 >>> 53 % 24
9 >>> -5
10 >>> --5
11 >>> ---5
```

Código 1: Operaciones Artiméticas –

Tengan presente que en Python el punto indica la precisión decimal, y los paréntesis permiten alterar el orden de precedencia natural (potencia, negación;  $-$ producto, división, división entera, y resto $-$ , y suma y resta). Los operadores que tienen igual precedencia, como por ejemplo la suma y la resta, son evaluados de izquierda a derecha.

```
1 >>> 'Hola Mundo!'
2 >>> 'abcd'+'efgh'
3 >>> 'a'*3
```

Código 2: Operaciones con Cadenas de Texto –

Recordar que las cadenas de texto pueden ser indicadas como tales utilizando comillas simples - ' - o comillas dobles - " - .

```
1 >>> x = 8
2 >>> x
3 >>> y = x*x
4 >>> 2*y
5 >>> 2*Y
6 >>> lenguaje = 'Python'
7 >>> 'Estoy programando en ' + lenguaje
8 >>> lenguaje = 'C'
```

```
9 >>> 'Estoy programando en ' + lenguaje
10 >>> pi = 3.141592653589793
11 >>> pi
12 >>> 15tambores = 'batucada'
13 >>> more@ = 100
14 >>> for = 'Programacion II'
```

### Código 3: Variables

No todo lo que se nos ocurra puede ser un nombre de variable. ¿Detectaron cuáles son los problemas? ¿Qué identificadores son inválidos y porqué?

```
1 >>> abs(10)
2 >>> abs(-10)
3 >>> max(5,9,-3)
4 >>> min(5,9)
5 >>> min(-1)
6 >>> len ("PYTHON")
7 >>> n = 17
8 >>> print(n)
9 >>> lenguaje = 'Python'
10 >>> print(lenguaje)
```

### Código 4: Funciones Standart

```
1 >>> def hola_marta():
2 >>>     return "Hola Marta! estoy programando en Phyton!!"
3 >>> hola_marta()
4
5 >>> def hola_pablo():
6 >>>     return "Hola Pablo! estoy programando en Phyton!!"
7 >>> hola_pablo()
8
9 >>> def hola(alguien):
10 >>>     return "Hola "+ alguien + "! estoy programando en Phyton!!"
11
12 >>> hola("Ana")
13 >>> hola("Pablo")
14 >>> saludo = 'Mundo'
15 >>> hola(saludo)
16
17 >>> def cuadrado(n):
18 >>>     return n*n
19
20 >>> cuadrado(5)
```

### Código 5: Funciones Propias

## 1.1. Un Problema Dos Soluciones

*Problema:* Pensar un número, duplicarlo, sumarle seis, y dividirlo por dos, y restarle el número elegido al comienzo. El número que queda es siempre tres.

*Solución 1:*

```
1 >>> def f1 (elegido):  
2 >>>     return ((elegido * 2) + 6) / 2 - elegido
```

Código 6: Primera Solución

*Solución 2:*

```
1 >>> def f2 (elegido):  
2 >>>     n = elegido * 2  
3 >>>     n = n + 6  
4 >>>     n = n / 2  
5 >>>     n = n - elegido  
6 >>>     return n
```

Código 7: Segunda Solución

Es importante aclarar que ambas soluciones son equivalentes. Probar que siempre devuelven el valor tres. Utilizar los siguientes valores: {9, 4, 118, 165414606, 0, -15}


## 1.2. El Ciclo Definido

*Problema* Supongamos que queremos calcular la suma de los primeros cinco números cuadrados. Reutilizaremos la función cuadrado ya definida en los ejemplos anteriores.

*Solución 1:* Fuerza Bruta.

```
1 >>> def suma_5_cuadrados():  
2     suma = 0  
3     suma = suma + cuadrado(1)  
4     suma = suma + cuadrado(2)  
5     suma = suma + cuadrado(3)  
6     suma = suma + cuadrado(4)  
7     suma = suma + cuadrado(5)  
8     return suma  
9  
10 >>> suma_5_cuadrados()
```

Código 8: Primer Intento



### Ciclo Definido

```

1  for X in range (N1, N2):
2      <hacer algo con X>

```

- Genera la secuencia de valores enteros del intervalo  $[N1, N2)$
- Para cada valor se repite el conjunto de instrucciones del cuerpo del ciclo.
- La sangría de las instrucciones del cuerpo del ciclo deben ser superior a la del encabezado.

*Solución 2:* Usando un Ciclo Definido.

```

1  >>> def suma_5_cuadrados():
2      suma = 0
3      for x in range(1, 6):
4          suma = suma + cuadrado(x)
5      return suma
6
7  >>> suma_5_cuadrados()

```

Código 9: Segundo Intento

*Solución 3:* Pensemos en Grande! Generalicemos .

```

1  >>> def suma_cuadrados(n):
2      suma = 0
3      for x in range(1, n+1):
4          suma = suma + cuadrado(x)
5      return suma
6
7  >>> suma_cuadrados(5)
8  >>> suma_cuadrados(100)

```

Código 10: Tercer Intento

### 1.3. Ayuda!

Podemos solicitar ayuda al interprete mediante la función `help()`. Tipee los siguientes comandos en el interprete y analice los resultados:

```

1  >>> help()
2  >>> help(abs)
3  >>> help(range)
4  >>> help('for')
5  >>> help('range')

```

Código 11: Comando help

## 2. Programas

### 2.1. Primer Programa: Imprimir en Pantalla

Generar el siguiente programa.

Nombre: **cuad100.py**

Descripción: *Imprime la suma de los primeros 100 números cuadrados*

```
1      def cuadrado (n):  
2          return n*n  
3  
4      def suma_cuadrados (n):  
5          suma = 0  
6          for x in range (1, n+1):  
7              suma = suma + cuadrado (x)  
8  
9          return suma  
10  
11     print (" La suma de los primeros 100 cuadrados es: ", ↵  
          suma_cuadrados(100))
```

En la consola del interprete ejecutar el siguiente comando:

```
1 >>> help('print')
```

En la consola del sistema operativo (shell) ejecutar:

```
1 $ python cuad100.py
```

#### ¿Qué destacamos de este programa?

- Organización y/o estructuración.
- División del problema en partes.
- Visualización de una ejecución.

### 2.2. Segundo Programa: Interacción con el Usuario

Generar el siguiente programa:

Nombre: **saludar.py**

Descripción: *Solicita el nombre al usuario y luego lo saluda*

```
1      def hola(nombre):
2          return "Hola " + nombre + "!"
3
4      def saludar():
5          nombre = input("Por favor ingrese su nombre: ")
6          saludo = hola (nombre)
7          print (saludo)
8
9      saludar()
```

En la consola del sistema operativo (shell) ejecutar:

```
1 $ python saludar.py
```

## 2.3. Tercer Programa: Algo más Interesante!

Generar el siguiente programa:

Nombre: **cuadrados.py**

Descripción: *Imprime la suma de los cuadrados comprendidos en el intervalo  $[n_1, n_2)$*

```
1      def imprimir_cuadrados():
2          print("Se calculará la suma de los cuadrados entre dos nú↵
3              meros ingresados")
4
5          n1 = int (input ("Ingrese un número entero: "))
6          n2 = int (input ("Ingrese otro número entero: "))
7          suma=0
8          for x in range (n1, n2):
9              suma= suma + x*x
10
11          print ("La suma de los cuadrados entre ", n1, "y ", n2, " ↵
12              es: ", suma)
13          print ("Es todo por ahora!")
14
15      imprimir_cuadrados()
```

## 3. Convención: La Estructura de Nuestros Programas

Recomendaciones al realizar las prácticas:

**a) Generales:**

- Sea claro/a y prolijo/a. Es muy importante que el código sea lo más claro y legible posible. Debe ser la traducción de nuestra *estrategia de solución*.
- Los identificadores (nombres) de funciones y variables sean coherentes. El identificador debe ser suficientemente descriptivo. Mayúsculas sólo para constantes.
- Ponga una línea en blanco entre las definiciones de función para simplificar la lectura del programa.
- Las expresiones matemáticas complejas pueden representarse en varios pasos.
- No use variables globales.

**b) Documentación:**

- Documente correctamente las funciones y módulos que desarrolle.
- Documente partes del código cuyo significado pudiera no quedar del todo claro.
- No documente en exceso, pero tampoco ahorre documentación necesaria. La documentación debe ser breve y concisa. La experiencia nos hace maestras/os de la documentación.

## 4. Un ejemplo completo

```
1  #-----
2  #Este programa calcula el area de un triangulo dado sus 3 lados
3  #Catedra Programación II, 2C
4  #Rosario, 05/03/2024
5  #-----
6
7  def calculo_A_Triangulo(base, altura):
8      '''
9      Diseño de datos:
10     Base (cm): float
11     Altura (cm): float
12     Area (cm): float
13     Signatura:
14     Area_Triangulo: float float -> float
15     Proposito:
16     Calcula el area de un triangulo en base a los datos dados
17     Ejemplo:
18     '''
19     Area_T = ((base * altura)/ 2)
20     return Area_T
21
22 def test_calculo_A_Triangulo():
```

```
23 assert (round (calculo_A_Triangulo(3.16, 4.43), 4) == 6.9994)
24 assert (round (calculo_A_Triangulo(2, 2), 1) == 2.0)
25 assert (round (calculo_A_Triangulo(5.1, 2.75), 15) == ↵
    7.012499999999999)
26
27 def main():
28     print("Programa para calcular Area de Triangulo")
29     base = float (input("Ingrese numero de la base: "))
30     altura = float (input("Ingrese numero de la altura: "))
31     print("El numero de area es: ", calculo_A_Triangulo(base, altura))
32
33 if __name__ == '__main__':
34     main()
```

## 5. Ejercitación

EJERCICIO 1. Correr tres veces el programa cuadrados .py con valores de entrada (3, 5), (3, 3) y (5, 3) respectivamente. ¿Qué sucede en cada caso?

EJERCICIO 2. La salida del programa cuadrados .py es poco informativa. Modificar el programa para que ponga el número junto a su cuadrado.

EJERCICIO 3. Escribir un programa que pregunte al usuario:

1. su nombre, y luego lo salude.
2. dos números, y luego muestre el producto.

EJERCICIO 4. Implementar algoritmos que permitan:

- a) Calcular el perímetro de un rectángulo dada su base y su altura.
- b) Calcular el área de un rectángulo dada su base y su altura.
- c) Calcular el perímetro y el área de un rectángulo (alineado con los ejes  $x$  e  $y$ ) dadas sus coordenadas  $x1, x2, y1, y2$ .
- d) Calcular el perímetro de un círculo dado su radio. *Ayuda:* Considerar a  $PI$  como 3,141592
- e) Calcular el área de un círculo dado su radio.
- f) Calcular el volumen de una esfera dado su radio.
- g) Dados los catetos de un triángulo rectángulo, calcular su hipotenusa.

*Ayuda:* Definir la función `raiz_cuadrada` primero. No utilice el módulo `math`

EJERCICIO 5. Mostrar el resultado de ejecutar estos bloques de código en el intérprete de python:



a)

```
1 >>> for i in range(5):
2 ...     print(i * i)
```

b)

```
1 >>> for i in range(2, 6):
2 ...     print(i, 2 ** i)
```

c)

```
1 >>> for i in range(6, 2):
2 ...     print(i* i)
```

d)

```
1 >>> for i in range(-5):
2 ...     print(i* i + 1)
```

e)

```
1 >>> for i in range(-5, -1):
2 ...     print(i* i, end="\t")
```

f)

```
1 >>> for i in range(-1, -5):
2 ...     print(i, end=" ")
```

g)

```
1 >>> for i in range(5,5):
2 ...     print("valor: ",i , end="\t")
```

h)

```
1 >>> for i in [1,3,5,7,9]:
2 ...     print("valor: ",i , end="\t")
```

i)

```
1 >>> for i in [-1,33,34,2,1]:
2 ...     print("valor: ", i, " cuadrado: ", i*i)
```

j)

```
1 >>> for i in [-1,0,-1,0,-1,0]:
2 ...     print("valor: ", i)
```

Luego de transcribir los resultados, analice los ciclos que no mostraron nada , y describa con sus palabras cuál es el problema desde el punto de vista del testing, que genera este tipo de situaciones.

EJERCICIO 6. Implementar un algoritmo que, dado un número entero  $n$ , permita calcular su factorial.

EJERCICIO 7. Implementar algoritmos que resuelvan los siguientes problemas:

a) Dados dos números, imprimir la suma, resta, división y multiplicación de ambos.

b) Dado un número entero  $n$ , imprimir su tabla de multiplicar.

EJERCICIO 8. Escribir un programa que le pida una palabra al usuario para luego imprimirla 1000 veces, en una única línea, con espacios intermedios.

*Ayuda:* Investigar acerca de los parámetros `end` y `sep` de la función `print`. Comenzar probando en el interprete los siguientes comandos:

```
1 >>> print ("tres", "tristes", "tigres", sep="-")
2 >>> print ("tres", "tristes", "tigres", sep="\t")
3 >>> print ("tres", "tristes", "tigres", sep="\n")
4 >>> print ("tres", "tristes", "tigres", sep=",")
5 >>> print ("tres", "tristes", "tigres", sep=",", end='\n')
6 >>> print ("tres", "tristes", "tigres", sep="---")
7
8 >>> print ("tres", "tristes", "tigres", end="---")
9 >>> print ("tres", "tristes", "tigres", end="")
10 >>> print ("tres", "tristes", "tigres", end=""); print("tomaban", "tres", "tragos", end↵
    ="\n")
```

EJERCICIO 9. Modificar el programa anterior para que el usuario también ingrese:

- a) la cantidad de veces que desea que se repita la palabra dada.
- b) el separador entre cada repetición de la palabra.

## 6. Entrega

El **ejercicio NRO 7** de esta práctica deberá ser entregado impreso a las/los docente de la cátedra el **JUEVES 14/03/2024**. La entrega será individual. La versión impresa puede ser escrita a mano o puede ser una impresión del código. Se evaluará:

- Documentación: Receta y Comentarios
- Solución algorítmica: Estrategia y Modularización
- Legibilidad y Claridad del código

## Referencias

- [1] Think Python: How to Think Like a Computer Scientist, Allen B. Downey, 2nd Edition, Version 2.2.18.
- [2] Algoritmos y Programación I, Aprendiendo a programar usando Python como herramienta, Rosita Wachenchauzer et.al., 2016, (sin publicar).