Ejercitación № 1

Cátedra Programación II Agosto 2016

1. *Conociendo un Poquito el Lenguaje Python*

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

```
>>> 2+3
>>> 5*7
>>> 2+3*7
>>> (2+3)*7
>>> 10/4
>>> 5**2
>>> 53//24
>>> 53 % 24
>>> -5
>>> --5
>>> ---5
```

Código 1: Operaciones Artiméticas –

Tengan presente que en Phyton 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.

```
>>> 'Hola Mundo!'
>>> 'abcd'+'efgh'
>>> '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 - .

```
>>> x = 8
>>> x
>>> y = x*x
>>> 2*y
>>> 2*Y
>>> lenguaje = 'Python'
>>> 'Estoy programando en ' + lenguaje
>>> lenguaje = 'C'
```

```
>>> 'Estoy programando en ' + lenguaje
>>> pi = 3.141592653589793
>>> pi
>>> 15tambores = 'batucada'
>>> more@ = 100
>>> 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?

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

Código 4: Funciones Standart

```
>>> def hola_marta():
>>>     return "Hola Marta! estoy programando en Phyton!!"
>>> hola_marta()

>>> def hola_pablo():
>>>     return "Hola Pablo! estoy programando en Phyton!!"
>>> hola_pablo()

>>> def hola(alguien):
>>>     return "Hola "+ alguien + "! estoy programando en Phyton!!"

>>> hola(Ana)
>>> hola(Pablo)
>>> saludo = 'Mundo'
>>> hola(saludo)

>>> def cuadrado(n):
        return n*n

>>> 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:

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

Código 6: Primera Solución

Solución 2:

```
>>> def f2 (elegido):
>>>     n = elegido * 2
>>>     n = n + 6
>>>     n = n / 2
>>>     n = n - elegido
>>>     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.

```
>>> def suma_5_cuadrados():
...     suma = 0 suma = suma + cuadrado(1)
...     suma = suma + cuadrado(2)
...     suma = suma + cuadrado(3)
...     suma = suma + cuadrado(4)
...     suma = suma + cuadrado(5)
...     return suma
>>>
>>> suma_5_cuadrados()
```

Código 8: Primer Intento

Ciclo Definido

```
for X in range (N1, N2):
    <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.

```
>>> def suma_5_cuadrados():
...     suma = 0
...     for x in range(1, 6):
...         suma = suma + cuadrado(x)
...     return suma

>>> suma_5_cuadrados()
```

Código 9: Segundo Intento

Solución 3: Pensemos en Grande! Generalicemos .

```
>>> def suma_cuadrados(n):
...     suma = 0
...     for x in range(1, n+1):
...         suma = suma + cuadrado(x)
...     return suma

>>> suma_cuadrados(5)
>>> suma_cuadrados(100)
```

Código 10: Tercer Intento

1.3. Ayuda!

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

```
>>> help()
>>> help(abs)
>>> help(range)
>>> help('for')
>>> 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
```

2.2. Segundo Programa: Interacción con el Ususario

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:

```
$ python saludar.py
```

2.3. Tercer Programa: Algo más Interesante!

Generar el siguiente programa:

Nombre: **cuadrados.py**
Descripción: *Imprime los cuadrados comprendidos en el intervalo* $[n_1, n_2]$

```
        def imprimir_cuadrados():
            print("Se calcularán cuadrados entre dos números ingresados")

            n1 = int (input ("Ingrese un número entero: "))
            n2 = int (input ("Ingrese otro número entero:  "))

            for x in range (n1, n2):
                print ( x*x )

            print ("Es todo por ahora")

        imprimir_cuadrados()
```

3. Convención: La Estructura de Nuestros Programas

Recomendaciones al realizar las prácticas:

a) Generales:

- Sea claro y prolijo. Es muy importante que el código sea lo más claro y legible posible.
- Es muy importante que los identificadores de funciones y variables sean coherentes. El identificador debe ser suficientemente descriptivo.
- 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.
- Los ejercicios marcados con el símbolo $\star$ son más difíciles y no son de resolución obligatoria.

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.

4. 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.

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

a)

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

b)

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

c)

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

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:

```
>>> print ("tres", "tristes", "tigres", sep="-")
>>> print ("tres", "tristes", "tigres", sep="\t")
>>> print ("tres", "tristes", "tigres", sep="\n")
>>> print ("tres", "tristes", "tigres", sep=",")
>>> print ("tres", "tristes", "tigres", sep=",", end='\n')
>>> print ("tres", "tristes", "tigres", sep="---")

>>> print ("tres", "tristes", "tigres", end="---")
>>> print ("tres", "tristes", "tigres", end="")
>>> 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.

5. Entrega de Ejercicios

La entrega y corrección de ejercicios se harán a través de la plataforma:
`http://comunidades.campusvirtualunr.edu.ar/`
Oportunamente se les indicará cuáles serán los ejercicios a entregar y el formato de presentación en el que deberán hacerlo.

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 Phyton como herramienta, Rosita Wachenchauzer et.al., 2016, (sin publicar).