Conociendo Python (Primera parte)
MSc. Jimena A. Timaná P.

PYTHON

- Lenguaje de programación muy versátil, utilizado para desarrollar cualquier tipo de programa (de escritorio, web, etc); creado por Guido van Rossum en el año 1990.
- Python puede integrarse con .NET y Java, entre otros entornos.
- Algunas empresas que utilizan Python son Yahoo, Google, Walt Disney, Nokia, NASA (Más información: https://www.python.org/about/success/)
- Página Oficial: http://www.python.org/

Características

- Python es un lenguaje de programación multiparadigma (POO, programación estructurada, programación funcional).
- Lenguaje de alto nivel.
- Es un lenguaje Interpretado o de script.
- Tipado dinámico.
- Fuertemente tipado.
- Independiente de la plataforma: El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.
- Lenguaje bastante sencillo a la hora de crear programas. Un programa escrito en Python podría tener menos líneas de código que su equivalente en Java o C.

Ejemplo:

JAVA	class Holamundo
	{
	  public static void main(string [] args)
	  {
	  System.out.println("hola mundo");
	  }
	}

C	void main()
	{
	  printf ("Hola mundo");
	}

PYTHON	print "hola mundo" versiones 2.x print (“hola mundo”) versiones 3.x

- Python es *Case Sensitive* (distingue entre mayúsculas y minúsculas). A pesar de que se pueden usar acentos ni caracteres propios de otros idiomas como por ejemplo la **"ñ"**.

- Python dispone de un intérprete por línea de comandos en el que se pueden escribir instrucciones, una por una, en una especie de *shell*; el resultado puede verse y probarse de inmediato.

- Existen otros programas, tales como IDLE, que añaden funcionalidades extra al modo convencional, como el auto-completar código y el coloreado de la sintaxis del lenguaje.

Declaración de variables

En Python los *tipos básicos* se dividen en:

Tipos Básicos	Ejemplo
Números	
• Entero	3
• Coma flotante	5.38
• complejos	2 + 7j
Cadenas	"sistemas"
Booleanos	True (cierto) y False (falso).

En Python, una variable puede almacenar cualquier cosa: una letra, un digito, una cadena, un número grande, etc. Para su creación solo basta con definirla, es decir, asignarle un valor.

Ejemplos:

```
n = 17
n = "lenguaje"
n = 10.2
num = 100000
nombre = "juan"
x = 'n'
dias_semana = ["lunes","martes", "miercoles", "jueves", "viernes", "sabado", "domingo"]
```

Una característica especial de Python y que aprovecha el hecho de no contar con tipos de datos (**typeless**), es que se pueden definir varias variables de una sola vez.

Ejemplos:

```
num1, num2, num3 = 10, 30, 50
x = y =90
a,b,c = "sistemas", [6, "junio"], 8.5
```

Los nombres de las variables pueden tener una longitud arbitraria, pueden incluir letras y números, pero **deben** comenzar con una letra. Por convención no suele inicializar una variable con mayúscula. Si lo hace, recuerde que la distinción es importante: Num y num son dos variables totalmente diferentes. Recuerde además que no puede darle a las variables, nombres con palabras reservadas como class, if, entre otras.

Python maneja las siguientes palabras reservadas:

and	continue	else	for	import	not	raise
assert	def	except	from	in	or	return
break	del	exec	global	is	pass	try
class	elif	finally	if	lambda	print	while

También se puede utilizar notación científica, y añadir una e (de exponente) para indicar un exponente en base 10.

Ejemplo:

```
w= 0.1e-3          sería equivalente a 0.1 x 10-3 = 0.1 x 0.001 = 0.0001
```

Para mostrar o imprimir los datos almacenados en las variables, solo basta escribir el nombre de la misma (dentro del IDLE). Por ejemplo, si tenemos en cuenta el código anterior y escribimos en el editor la variable *w* obtendremos como salida el valor de la variable que es 0.0001.

Además se puede imprimir en una sola línea más de dos variables al mismo tiempo.

```
>>> a = "hola"
>>> c = 50.5
>>> a,c
('hola', 50.5)
```

Para las cadenas, se puede utilizar comillas dobles o comillas simples.

Las cadenas admiten operadores como el + para realizar concatenaciones y el * que repite la cadena el número de veces que se indique en el segundo operando.

Ejemplo:

```
>>> a = “hola ” + “ chicos”
hola chicos
```

Para comentariar texto utilice el carácter **#**.

Python funciona como una calculadora simple: se introduce la expresión numérica e inmediatamente la realiza:

```
>>> 2+3
5
>>> 2 +  4 + 3j
(6+3j)
```

Operadores Aritméticos

Los símbolos +, -, /, y el uso de los paréntesis para el agrupamiento, se usan todos de la misma forma que en matemáticas. El – también es utilizado para la negación. El asterisco * es el signo de multiplicación, ** el símbolo para la exponenciación, // para la división entera, % para el módulo.

Cuando aparece más de un operador en una expresión matemática, el orden de evaluación depende de las *reglas de precedencia*.

Paréntesis: tienen la precedencia más alta y pueden usarse para forzar una expresión evaluada, en un orden deseado. Puesto que las expresiones entre paréntesis se evalúan primero, 2 * (3-1) es igual a 4, y (1+1)**(5-2) es igual a 8.

Exponenciación: tiene la siguiente precedencia más alta; así pues 2**1+1 es igual a 3 y no a 4 y 3*1**3 es igual a 3 y no a 27.

La **Multiplicación** y la **División** tienen la misma precedencia, que es más alta que la de la **Adición** y la **Sustracción**, que tienen también la misma precedencia. Por tanto 2*3-1 devuelve 5 y no 4.

Los operadores que tienen la misma precedencia se evalúan primero de izquierda a derecha.

Tenga en cuenta que operaciones entre enteros da como resultados enteros. Si se quiere resultados con decimales, al menos uno de los operandos debe ser real.

Operadores Lógicos

Los operadores lógicos los utilizamos principalmente cuando realizamos trabajos con valores booleanos. Estos son: and, or, not.

Si no recuerda el tipo de dato básico que tiene un valor determinado, puede preguntarlo con **type.**

Ejemplos:

```
>>> type("Bienvenidos")
<class 'str'>

>>> type(82)
< class 'int'>
```

```
>>> type(5.3)
< class 'float'>

>>> type(5.7 + 6j)
< class 'complex'>

>>> x = True
>>> type(x)
< class 'bool'>
```

Qué resultado arroja:

```
>>> type('3.5')
```

Cuando cree enteros largos **NO** utilice comas ni puntos para separar los grupos de dígitos.

Forma incorrecta:

1.000.000 ó 178,476,352

Ya que de esta última manera está creando una lista de números:

```
>>> c = 178,476,352
>>> c
(178, 476, 352)
```

Entrada y Salida

Si se quiere presentar al usuario mensajes o imprimir el valor de una variable dentro de un programa, no basta con escribir la variable y presionar Enter, es necesario usar la función **print**

Ejemplo:

```
>>>nombre, salario = "Gaby", 5000000
>>> print("la Señorita ", nombre, " se gana ", salario, " mensualmente")
La Señorita Gaby se gana  5000000  mensualmente
```

Ejercicio: realice la misma impresión del mensaje usando la concatenación con el símbolo +

Para forzar el casteo o la transformación de algunas variables a un "tipo de dato distinto" se pueden usar funciones como *int*, *float*, str

Ejemplos:

```
>>> a = 10.68
>>> print(int(a))
10
```

```
>>> b = 6
>>> print(float(b))
6.0
```

Pregunta: *¿Es posible castear un entero a complejo y viceversa?*

Para la entrada de datos (leer desde el teclado) se utiliza la función **input.**

Ejemplos:

```
>>> edad = input("que edad tienes ?: ")
que edad tienes ?: 25
>>> print(edad)
25
```

```
# Se puede escribir en un archivo aparte las siguientes sentencias de código
print ("escriba su nombre: ")
nombre = input()
print ("escriba su cédula: ")
cedula = input()
print ("escriba su edad: ")
edad = input()
```

```
# Otra versión del ejercicio
nombre = input("escriba su nombre: ")
cedula = input("escriba su cedula: ")
edad = input("escriba su edad: ")
```

Condicionales

La sentencia requerida para realizar condicionales es:

if *condición* **:**
    # ordenes que se van a ejecutar si la condición es correcta
else:
    # lo que se ejecuta si la condición es falsa.

Ejemplo:

```
salario = int(input("cual es tu salario ?:"))
if salario > 500000:
    print ("ganas más del salario mínimo !")
else:
    print ("estás ganando muy poco y el contrato que tienes es ilegal !!")
```

Si se necesita elegir entre varias opciones se puede elegir la siguiente estructura de decisión:

if condición_1:
  bloque 1
elif condición_2:
  bloque 2
else:
  bloque 3

Operadores de comparación:

Operador	Descripción
x != y	x no es igual a y
x == y	x es igual a y
x < y	x es menor que y
x > y	x es mayor que y
x >= y	x es mayor o igual que y
x <= y	x es menor o igual que y

Ejemplos:

```
edad = int(input('¿Ingrese su edad?:'))
if edad <= 0:
   print( '¿Aún no has nacido?')
elif edad<16:
   print( 'Legalmente todavía no puedes trabajar')
elif edad<65:
   print( 'Supongo que estás trabajando!!!')
elif edad<100:
   print( 'Supongo que estarás ya jubilado(a), verdad?')
else: print( '¿Seguro que tienes', edad, 'años?')
```

Si necesita utilizar algún módulo preestablecido en python, utilice la cláusula **import** al principio de su programa.

Ejemplos: import math

Si desea utilizar alguna función matemática como raíz cuadrada, coseno, etc, invóquela así: math.cos() , math.sin(), etc.

Iteración WHILE

while condicion:
  Cuerpo del bucle, que se repetirá mientras la condición sea cierta.

Ejemplos:

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

Iteración FOR

los **iterables** son objetos que pueden ser iterados o accedidos con un índice. Algunos ejemplos de iterables en Python son las listas, tuplas, cadenas o diccionarios.

Por lo tanto, cuando utilicemos un **for**, lo que va luego de la palabra **in** debe ser un iterable.

for *variable* in *Objetoiterable*:
    # aquí su código

Ejemplo:

```
for i in "jimena":
    print(i)
```

```
j
i
m
e
n
a
```

El ciclo **for** también trabaja en conjunto con la función range () en la que, por lo general, internamente se define un valor inicial, un valor final y el paso. Sin embargo, la función range(), solo necesita como mínimo el valor final donde por defecto el paso es de uno en uno y se inicia desde cero.

Tenga en cuenta que el ciclo se hará hasta el valor final **-1**.

Si se quiere que se haga una cuenta regresiva, el valor del paso deberá ser negativo.

for *indice* **in range(**valorInicial**,** valorFinal**,** paso**):**
        sentencias a ejecutar dentro del cuerpo del bucle

Ejemplos:

```
for i in range(1,10,1):
    print(i)
```

```
for i in range(10):
    print(i)
```