Introducción a Python

El primer programa: Hola mundo

Un clásico... Hola Mundo en Python

In [ ]:

```
#-----------------------------------------------------
# Programa Hola Mundo que utiliza la funcion print
# para mostrar un mensaje en pantalla
#-----------------------------------------------------

print("Hola Mundo")     # Esto es facil!
```

Tipos de datos y operadores aritméticos

Python dispone de un conjunto de tipos numericos para realizar las operaciones matemáticas básicas (sumas, restas, multiplicación, división). Utilice la celda inferior para realizar calculos numéricos sencillos (No olvide presionar la combinación de teclas *Shift* + *Enter* para ejecutar sus instrucciones).

In [ ]:

```
1 + 2
```

In [ ]:

```
5 - 8
```

In [ ]:

```
2 * 6
```

In [ ]:

```
10 * 2.5
```

In [ ]:

```
10 / 3
```

In [ ]:

```
12 + 6 / 2
```

Python tiene como otros lenguajes de programación una jerarquia en el orden de ejecución de intrucciones. Puede utilizar () para modificar esta regla.

Operación	Operador	Orden
Paréntesis	()	1
Potenciación	**	2
División	/	3
División Entera	//	3
Multiplicación	*	3
Módulo	%	3
Suma	+	4
Resta	-	4

In [ ]:

```
(12 + 6) / 2
```

In [ ]:

```
12 + 4 / (6 + 2) - 4
```

Los errores en Python se muestran como un mensaje devuelto por el interprete. Lor errores incluyen un texto que indica el tipo de error, una flecha (---->) que indica donde esta el error y el nombre de esta Excepción que es como se suele denominar a los tipos de errores. Intente realizar una división entre cero en la celda inferior.

In [ ]:

```
7 / 0
```

Python incluye la posibilidad de trabajar con numeros complejos, utiliando el simbolo j. Ingrese en la celda inferior un número complejo como 2 + 3j.

In [ ]:

```
2 + 3j
```

Otras instrucciones aritméticas disponibles en Python son módulo (%) que puede manejar números con decimales, y la operación potenciación (*) asi como la division entera (//)

In [ ]:

```
2 ** 3
```

In [ ]:

```
10 // 3
```

In [ ]:

```
10 % 3
```

Python reconoce el tipo de dato ingresado por la forma como este se ha escrito: valores sin decimales como enteros (int), valores con decimales como flotantes (float) y valores complejos por contener el caracter j (complex). Se puede utilizar la función type() para revisar el tipo de datos.

In [ ]:

```
type(5)
```

In [ ]:

```
type(1.2)
```

In [ ]:

```
type(2+3j)
```

In [ ]:

```
type('Hola')
```

In [ ]:

```
type('C')
```

Se pueden utilizar las instrucciones int, float y complex para convertir entre tipos de datos, aunque no todas las conversiones de datos estan permitidas

In [ ]:

```
int(2.9)
```

In [ ]:

```
float(10)
```

In [ ]:

```
str(3.1415)
```

In [ ]:

```
float('1.7172')
```

In [ ]:

```
complex(3)
```

In [ ]:

```
int(3+3j)
```

In [ ]:

```
int('Hola')
```

In [ ]:

```
float(1+2j)
```

El tipo de dato string (str) tiene muchas formas de definirse: pueden utilizarse comillas simples ('), comillas dobles ("), comillas simple triples (''') o comillas dobles triples (""").

In [ ]:

```
'Una golondrina no hace verano'
```

In [ ]:

```
"Una golondrina no hace verano"
```

In [ ]:

```
'''Tres golondrinas... esas sin hacen verano'''
```

In [ ]:

```
"""...pero cuatro ya son multitud"""
```

Las definición de un string con triples comillas (ya sean simples o dobles) permite insertar una nueva línea en el string (lo que se muestra con el caracter de escape \n)

In [ ]:

```
'''Este texto esta compuesto por una linea...
y otra linea'''
```

Pero en la impresión, esta caracter \n se interpretara correctamente como una nueva linea

In [ ]:

```
print('''Este texto esta compuesto por una linea...
y otra linea''')
```

Esta combinación de comillas simples y dobles permite incluir estas mismas comillas dentro de un string

In [ ]:

```
print('El nombre cientifico de la golondrina es "Hirundo Rustica"')
```

Hay dos operaciones aritméticas que afectan a un string. La multiplicación repite una cadena por el factor de multiplicación:

In [ ]:

```
print("Varios Holas " * 3)
```

Mientras que el operador "+" concatena (es decir, une) dos strings

In [ ]:

```
print("Hola" + ' ' + 'Mundo')
```

Asignacion de variables en Python

La asignación de variables en Python se realiza con el operador =

In [ ]:

```
var = "Soy una variable"
```

Las asignaciones no se muestran en la pantalla, como sucede como otras herramientas interpretadas, como MATLAB u Octave. Para visualizar el contenido de una variable sera necesario volver a escribir el nombre de la variable.

In [ ]:

```
var
```

Es interesante notar que las variables no requieren tipos (como sucede en lenguajes compilados como C o C++). Esto es porque Python es de tipo dinamico. Pruebe asignar varios tipos de datos a la misma variable y verifique con la funcion type() que el tipo de datos reconocido por Python es diferente.

In [ ]:

```
var = 10
type(var)
```

In [ ]:

```
var = 3.1416
type(var)
```

In [ ]:

```
var = "Hola"
type(var)
```

In [ ]:

```
var = 2+3j
type(var)
```

In [ ]:

```
var = (10, 12, 14, 16)
type(var)
```

In [ ]:

```
var = [10, 12, 14, 16]
type(var)
```

In [ ]:

```
var = {1, 2, 3, 4, 5}
type(var)
```

In [ ]:

```
var = {'Juan': 987654334, 'Maria': 986353733}
type(var)
```

Es posible asignar varios valores a varias variables a la vez utilizando "," para separar las variables y los valores a asignar.

In [ ]:

```
var1, var2 = 1, 3
print(var1, var2)
```

Operadores de comparación

Los operadores de comparación son ==, !=, <, <=, >, >=

In [ ]:

```
3.0 == 3.0
```

In [ ]:

```
1 == 1.0
```

In [ ]:

```
12 > 10
```

In [ ]:

```
20.10 != 20.00
```

In [ ]:

```
12 <= 12
```

Nota: Las celdas en Jupyter solo imprimen el resultado de la ultima instruccion, a menos que se utilice la instruccion print para especificar la impresion de los resultados.

In [ ]:

```
12 + 3
12 % 7
1 + 2
```

In [ ]:

```
print(12 + 3)
print(14 % 7)
print(1 + 2)
```

Reglas de nombres de variables

Las variables es Python pueden ser arbitrariamente largas. Pueden contener letras y numeros, pero deben de empezar con una letra o con un caracter de subrayado (_). Asi también, aunque se pueden utilizar letras mayusculas, por convención no se utilizan. En Python, *var* y *Var* hacen referencia a variables diferentes

In [ ]:

```
12cervezas = "Una caja"
```

In [ ]:

```
tres% = "Tres por ciento"
```

In [ ]:

```
class = "Seccion A"
```

Todos los ejemplos anteriores son ejemplos de variables con nombres inválidos, porque no inicia con una letra o subrayado, porque utilizan un caracter ilegal o porque utiliza como nombre de variable una palabra reservada (class). Python tiene más de 30 palabras reservadas (este número puede cambiar según las versiones de Python):

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		

Ayuda en Python

Para concluír... ¿y donde encuentro la ayuda en el mismo sistema Python? La ayuda en consola se puede obtener utilizando la instruccion *help()* (para salir es necesario ingresar "q").

En Jupyter y en muchos IDEs se puede obtener ayuda presionando la tecla [Tab] mientras se va ingresando una instrucción.

Toda la documentación de Python se puede consultar en la ayuda en linea (https://docs.python.org/3/library/index.html).