

Inicios de Programación en Python.



Brayan Steven Burgos Delgado

brayan.burgos@mail.escuelaing.edu.co

brayanburgos1437@gmail.com

Agenda

- Configuración
- Scripts - Ejecución
- Variables - Identificadores
- Entrada y Salida de datos
- Condicionales
- Condiciones repetitivos
- Listas

Referencias

Programas en Python

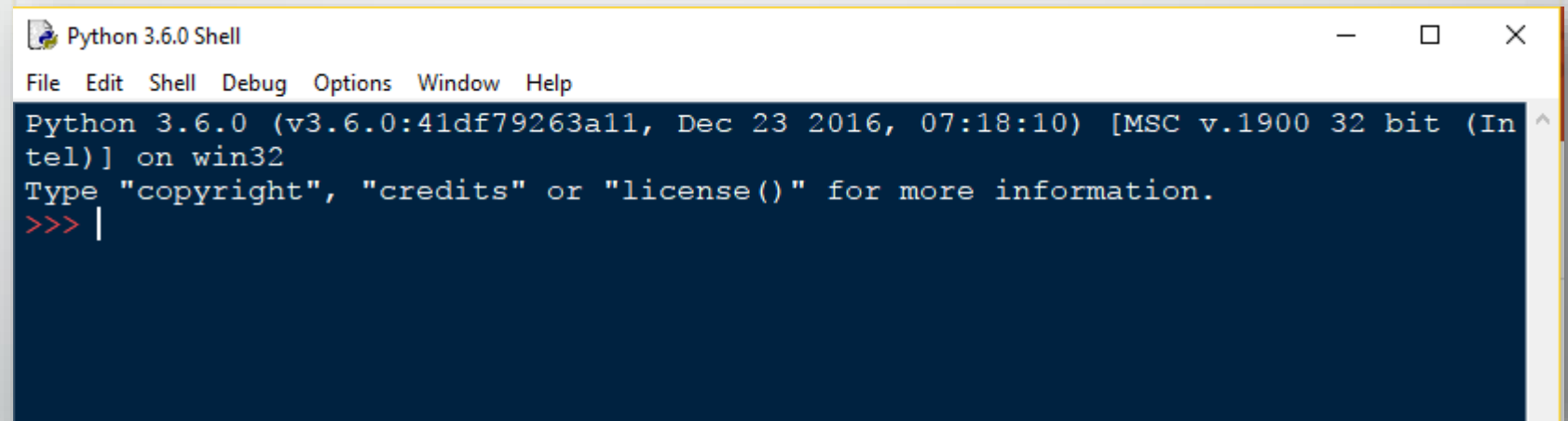
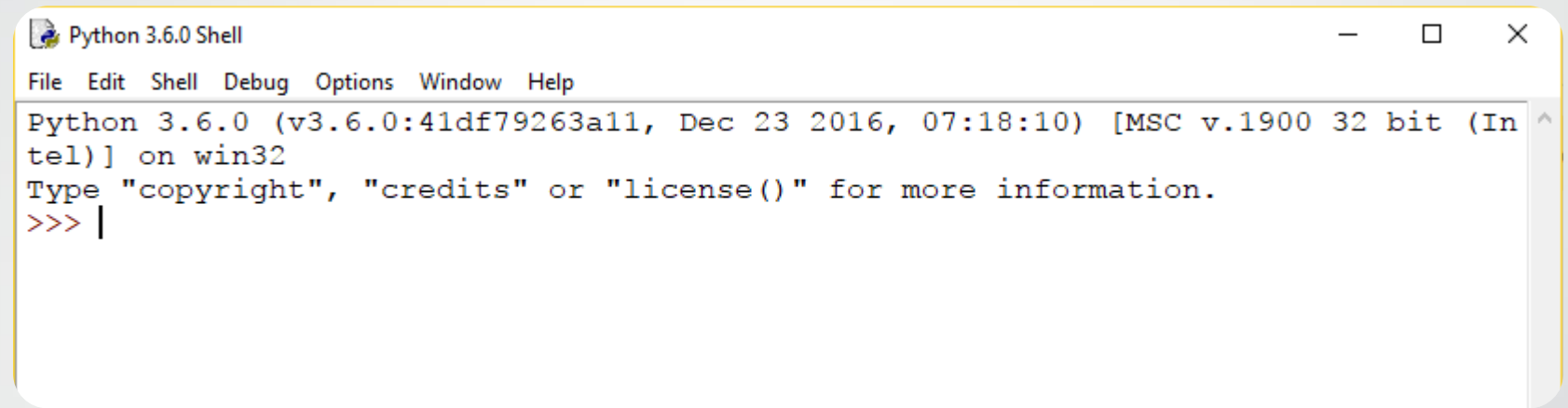
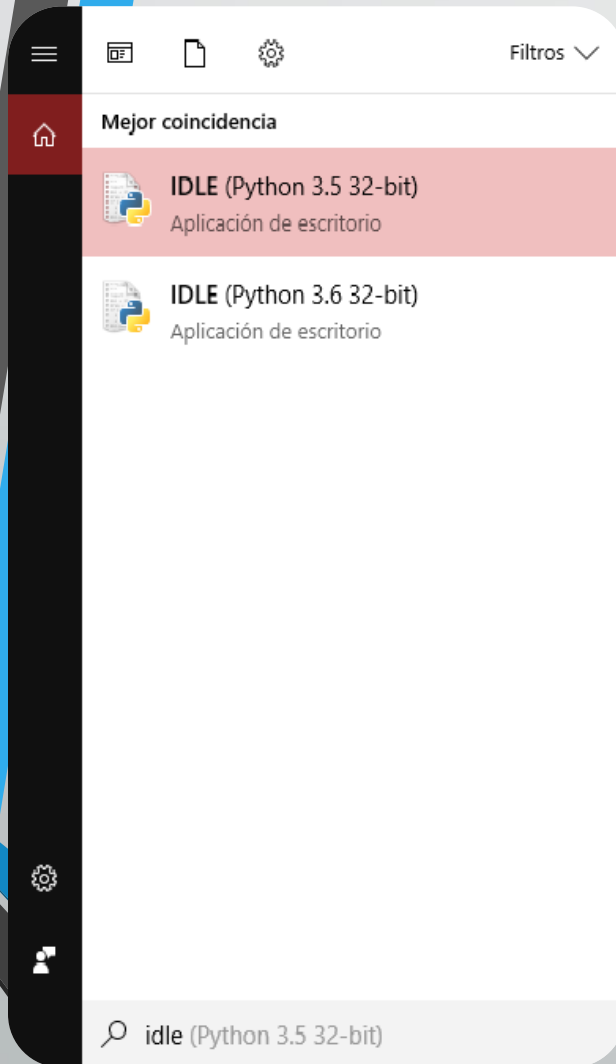
- Programas en Python
- También llamados Script

Un script es una secuencia de definiciones y comandos

Las definiciones son evaluadas y los comandos son ejecutados por el interprete de Python en la consola

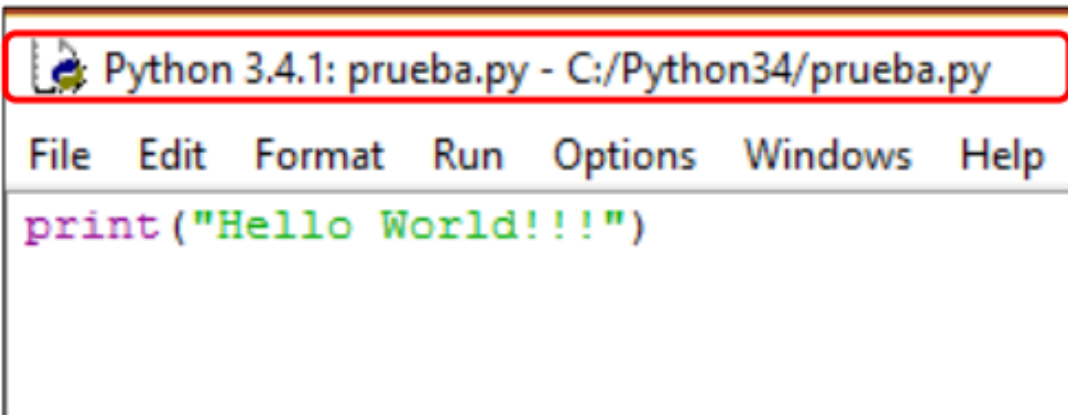
- Los comandos le indica al interprete que tiene que hacer.
- Los scripts se almacenan en un archivo con extensión **.py**

Ejecución



Ejecución de Programas

2. Llamando al archivo .py desde el sistema operativo



```
C:\Python34>python prueba.py
Hello World!!!
```

Variables.

- Son lugares reservados en memoria donde se puede almacenar datos.
- Toda variable tiene Nombre, Tipo y Contenido.
Nombre: Alfanumérico
Tipo: Str, Int, Float, Complex, Bool,....
Contenido: Relacionado con el tipo de la variable
- Python automáticamente reserva el espacio en memoria para cada variable.

TIPOS DE VARIABLES

- Numéricos
INT
FLOAT
COMPLEX

```
>>> x = 8
>>> int(x)
8
>>> float(x)
8.0
>>> complex(4, 0.2)
(4+0.2j)
>>> type(x)
<class 'int'>
```

- Boléanos
BOOL

```
>>> a = True
>>> False != True
True
>>> True != True
False
```

Identificadores o Asignaciones

Son palabras que se define a gusto del programador para diferenciar las variables que se van a utilizar para la realización del programa.


Deben comenzar por una letra.

- No se permiten palabras reservadas: and, global, or, assert, else, if, pass, break, except, import, print, class, exec, in, raise, continue, finally, is, return, def, for, lambda, try, del, from, not, while, int, float, bool...

- En Python no es necesario crear variables.

El signo "=" es utilizado para asignarle un valor a una variable.

```
>>> x=2
>>> y="Hola castillistas"
>>> z=True
>>> a=2,5
>>> b=false
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b=false
NameError: name 'false' is not defined
>>> b=False
```



Reconoce
MAYUSCULAS
y minúsculas

Operaciones Básicas

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	$a + b = 30$
- Subtraction	Subtracts right hand operand from left hand operand.	$a - b = -10$
* Multiplication	Multiplies values on either side of the operator	$a * b = 200$
/ Division	Divides left hand operand by right hand operand	$b / a = 2$
% Modulus	Divides left hand operand by right hand operand and returns remainder	$b \% a = 0$
** Exponent	Performs exponential (power) calculation on operators	$a ** b = 10 \text{ to the power } 20$
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	$9 // 2 = 4$ and $9.0 // 2.0 = 4.0$, $-11 // 3 = -4$, $-11.0 // 3 = -4.0$

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	$(a == b)$ is not true.
!=	If values of two operands are not equal, then condition becomes true.	
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	$(a > b)$ is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	$(a < b)$ is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	$(a >= b)$ is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	$(a <= b)$ is true.

Entrada y Salidas

- 1.TIPO INPUT:
Lectura de información por medio del teclado.
Para ingresar información al programa Input("")

```
print("Este programa te saludara")
nombre=input("Escribe tu nombre primer nombre;")
apellido=input("Escribe tu nombre primer apellido;")

print("BUENAS TARDES",nombre,apellido)
|
```

```
Este programa te saludara
Escribe tu nombre primer nombre;Pepito
Escribe tu nombre primer apellido;Perez
BUENAS TARDES Pepito Perez
>>> |
```

```
from sys import stdin
print("Este programa te saludara")
nombre=str(stdin.readline().strip())
apellido=str(stdin.readline().strip())
print("BUENAS TARDES",nombre,apellido)
```

```
Este programa te saludara
Pepito
Perez
BUENAS TARDES Pepito Perez
>>> |
```

- 2. Lectura de información desde archivos planos. [?] Los problemas de la arena se resuelven por medio de la salida y entrada estándar. Se utiliza la librería sys de Python

CONDICIONALES

- Día a día llevamos a cabo acciones con base en algunas condiciones.
- 1.Si tengo sed, entonces tomo agua
- 2. Si no estudio, entonces tendré malas notas
- 3.Mientras llueva, llevo la sombrilla abierta
- 4.Mientras tengo hambre, debo comer...

Las sentencias terminan en nueva línea
Los bloques son indicados por una
tabulación

obligatorio

```
if expresion:  
    print("identacion o TAB")  
print("fuera del condicional")
```

**Toda orden del condicional, va a
un TAB del condicional**

Ejemplo de los Condicionales

```
x = int(input("Ingresa un número:"))

if x%2==0:
    print("El número", x, " es PAR")
    print("Estuve por ACA")

if x%2!=0:
    print("Estuve por AQUÍ")
    print("El número", x, " es IMPAR")

print("Byee")
```

```
>>>
Ingresa un número:20
El número 20  es PAR
Estuve por ACA
Byee
```

```
>>>
Ingresa un número:11
Estuve por AQUÍ
El número 11  es IMPAR
Byee
```

if-ELSE

- Se utiliza cuando es la ultima opción de tu programa y no hay mas posibilidad sino esa.

```
x = int(input('Ingrese un entero: '))
if x%2 == 0:
    print('Número Par')
else:
    print('Número Impar')

print('Programa realizado con Condicionales')
```

```
>>>
```

```
Ingrese un entero: 25
```

```
Número Impar
```

```
Programa realizado con Condicionales
```

```
>>>
```

```
Ingrese un entero: 58
```

```
Número Par
```

```
Programa realizado con Condicionales
```

If- ELIF- else

- Son condicionales para dar mas de dos opciones para la solución de algún problema: en este caso son tres opciones PERO pueden ser mas de tres opciones

```
x = int(input("Ingresa un número:"))  
  
if x>0:  
    print("El número", x, " es POSITIVO")  
elif x<0:  
    print("El número", x, " es NEGATIVO")  
else:  
    print("El número", x, " es CERO")  
  
print("Byee")
```

```
Ingresa un número:25  
El número 25  es POSITIVO  
Byee
```

```
>>>  
Ingresa un número:0  
El número 0  es CERO  
Byee
```

```
>>>  
Ingresa un número:-2  
El número -2  es NEGATIVO  
Byee
```


Condicionales Repetitivos-WHILE

- Permiten repetir varias veces la misma instrucción
- La cantidad de repeticiones debe ser en algún momento falsa

IMPORTANTE: EL
VALOR BOOL DEBE
SER FALSA EN
ALGUN MOMENTO

DOS
PUNTOS

"://
.

```
while expresion:  
    print("TAB")  
print("programa realizado con WHILE")|
```

EJEMPLO DEL WHILE.

```
sigla=input("ingrese la sigla del curso para continuar")
puntos=0
while sigla=="IPP":
    puntos=puntos+1
    sigla=input("ingrese la sigla del curso para continuar")
print("su puntuacion es",puntos)
```

Hagámoslo en vivo

```
from sys import stdin
print("este programa dividira en 2| tu numero ingresado, mientras sea menor o igual a 100")
numero=int(stdin.readline().strip())
while numero<=100:
    print(numero/2)
    numero=int(stdin.readline().strip())

print("tu numero es mayor a 100,imposible")
```

```
este programa dividira tu numero ingresado, mientras sea menor o igual a 100
1
0.5
10
5.0
35
17.5
50
25.0
100
50.0
101
tu numero es mayor a 100,imposible
>>> |
```

Condicionales repetitivos-FOR

- Se recurre a ellos cuando quieres realizar n veces una instrucción.
- El ejemplo mas sencillo es mostrar n veces la frase ("Programar es muy divertido").
- Tiene infinidad de usos a nivel de programación, como la obsolescencia programada

```
n=int(input("numero de veces que quiere mostrar \"Programar es muy divertido\" "))  
for i in range (n):  
    print("Programar es muy divertido")
```

```
numero de veces que quiere mostrar \"Programar es muy divertido\" 5  
Programar es muy divertido  
Programar es muy divertido  
Programar es muy divertido  
Programar es muy divertido  
Programar es muy divertido  
>>> |
```

Listas

- Son un conjunto de objetos con una cantidad, puede contener cualquier tipo de dato(int, str, float, bool, list, etc)
- Se delimita con “[]” y los elementos van separados por “,” comas.
- ejemplo: Lista=[“Hola”,1234, “Pedro”, 5.32, False]

Tamaño de la
lista es 5

	-5	-4	-3	-2	-1	: Negativos
	“Hola”	1234	“Pedro”	5.32	False	
Index:	0	1	2	3	4	

```
lista=["holaCastillistas", 4, False, 5.6, "seacabolalista"]  
print(lista[0])  
print(lista[1])  
print(lista[-1], "//estas son las posiciones negativas")  
print(lista[-5], "//esto igual a la posicion 0")
```

```
holaCastillistas
```

```
4
```

```
seacabolalista //estas son las posiciones negativas
```

```
holaCastillistas //esto igual a la posicion 0
```

```
>>> |
```

```
print("veremos una forma de usar el bucle FOR con una lista")
lista=["holaCastillistas", 4, False, 5.6, "seacabolalista"]
for i in lista:
    print(i)#una forma de recorrer toda la lista

#for j in range (len(lista)):
#    print(lista[j])
#segunda forma de recorrer un lista
```

```
veremos una forma de usar el bucle FOR con una lista
holaCastillistas
4
False
5.6
seacabolalista
>>> |
```

Operación	Resultado
<code>x in s</code>	Indica si la variable <code>x</code> se encuentra en <code>s</code>
<code>s + t</code>	Concatena las secuencias <code>s</code> y <code>t</code>
<code>s * n</code>	Concatena <code>n</code> copias de <code>s</code>
<code>s[i]</code>	Elemento <code>i</code> de <code>s</code> , empezando por <code>0</code>
<code>s[i:j]</code>	Porción de la secuencia <code>s</code> desde <code>i</code> hasta <code>j</code> (no inclusive)
<code>s[i:j:k]</code>	Porción de la secuencia <code>s</code> desde <code>i</code> hasta <code>j</code> (no inclusive), con paso <code>k</code>
<code>len(s)</code>	Cantidad de elementos de la secuencia <code>s</code>
<code>min(s)</code>	Mínimo elemento de la secuencia <code>s</code>
<code>max(s)</code>	Máximo elemento de la secuencia <code>s</code>

Referencias

- http://librosweb.es/libro/algoritmos_python/capitulo_7/listas.html
- <http://paginaspersonales.deusto.es/dipina/teaching.html>
- <https://www.python.org/>