

# *Contenidos*

- Introducción a la programación
  - Python
  - Hola Mundo
  - Tipos de Datos
  - Declaración de variables
  - Estructuras de control y repetición
  - Ejemplos de programación
- 
-

# ***Introducción a la programación***

- Se llama **Programación** a la implementación de un algoritmo en un determinado lenguaje de programación, para resolver un problema
  - **Algoritmo** es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema
  - **Programa** (Software en inglés) es una secuencia de instrucciones que una computadora puede interpretar y ejecutar
- 
-

# *Lenguaje de programación*

- **Lenguaje de programación** es el idioma utilizado para controlar el comportamiento de una máquina
  - Consiste en un **conjunto de símbolos y reglas sintácticas y semánticas** que definen su estructura y el significado de sus elementos y expresiones
  - En la actualidad los lenguajes de programación están escritos para ser comprensibles por el ser humano, a este código se le llama **código fuente**, pero no es comprendido por la máquina ya que esta sólo maneja el **lenguaje binario**
- 
-

# *Lenguaje de programación*

- La **compilación** es el proceso de traducir un programa en código fuente a programa ejecutable. Ejem: C, pascal
- Los **lenguajes interpretados** necesitan de un programa que traduzca el código fuente a instrucciones de la plataforma en la que se ejecutan. Ejem: Python, Visual Basic, Script.

# *Python*

- Lenguaje interpretado o de script
- Tipado dinámico
- Multiplataforma: UNIX, Solaris, Linux, DOS, Windows, android, Mac OS, etc
- Orientado a objetos
- ¿Por qué python?

# *Python*

- Instalación:

- <http://recursospython.com/guias-y-manuales/instalar-python-3-en-windows-8/>

<http://recursospython.com/guias-y-manuales/instalar-python-3-en-windows-8/>

- Primer Programa

```
print ("Hola Mundo")  
input()
```

# *Variables*

- Una **variable** es un espacio para almacenar datos modificables, en la memoria de un ordenador. En Python, una variable se define con la sintaxis:
  - nombre\_de\_la\_variable = valor\_de\_la\_variable  
a = 23
- Cada variable, tiene un nombre y un valor, el cual define a la vez, el tipo de datos de la variable

# ***Tipos básicos***

- Números, como pueden ser: 7 (entero), 15.57 (de coma flotante) o  $7 + 5j$ (complejos)
  - Cadenas de texto: "Hola Mundo"
  - Valores booleanos: True (verdadero) y False (falso)
- 
-

# *Tipos básicos*

- Ejemplo:

```
# esto es una cadena
```

```
c = "Hola Mundo"
```

```
# y esto es un entero
```

```
e = 23
```

```
# podemos comprobarlo con la función type
```

```
type(c)
```

```
type(e)
```

---

---

# ***Tipos básicos***

- En Python, a diferencia de muchos otros lenguajes, no se declara el tipo de la variable al crearla. En Java, por ejemplo, escribiríamos:

```
String c = "Hola Mundo";  
int e = 23;
```

# *Tipos básicos*

- Números
  - Enteros: int y long
  - Reales (decimales): float

- Operaciones

Operador	Descripción	Ejemplo
+	Suma	<code>r = 3 + 2</code> # r es 5
-	Resta	<code>r = 4 - 7</code> # r es -3
-	Negación	<code>r = -7</code> # r es -7
*	Multipliación	<code>r = 2 * 6</code> # r es 12
**	Exponente	<code>r = 2 ** 6</code> # r es 64
/	División	<code>r = 3.5 / 2</code> # r es 1.75
//	División entera	<code>r = 3.5 // 2</code> # r es 1.0
%	Módulo	<code>r = 7 % 2</code> # r es 1

# *Tipos básicos*

- Cadenas de texto
  - Concatenación

```
a = "uno"
```

```
b = "dos"
```

```
c = a + b # c es "unodos"
```

```
print(c)
```

```
c = a * 3 # c es "unounouno"
```

```
print(c)
```

---

---

# *Tipos básicos*

- Booleanos: True, False
  - Importantes en expresiones
  - Realmente representan 0 y 1

Operador	Descripción	Ejemplo
and	¿se cumple a y b?	<code>r = True and False</code> # r es False
or	¿se cumple a o b?	<code>r = True or False</code> # r es True
not	No a	<code>r = not True</code> # r es False

- Mostrar tablas de valores

# *Tipos básicos*

- Booleanos: True, False

Operador	Descripción	Ejemplo
==	¿son iguales a y b?	<code>r = 5 == 3 # r es False</code>
!=	¿son distintos a y b?	<code>r = 5 != 3 # r es True</code>
<	¿es a menor que b?	<code>r = 5 &lt; 3 # r es False</code>
>	¿es a mayor que b?	<code>r = 5 &gt; 3 # r es True</code>
<=	¿es a menor o igual que b?	<code>r = 5 &lt;= 5 # r es True</code>
>=	¿es a mayor o igual que b?	<code>r = 5 &gt;= 3 # r es True</code>

# ***Tipos básicos***

- Ejercicio:
  - Crear un programa llamado tiposBasicos.py probando todo lo visto hasta ahora

# *Control de flujo: Sentencias condicionales*

- Si un programa no fuera más que una lista de órdenes a ejecutar de forma secuencial, una por una, no tendría mucha utilidad
  - Los condicionales nos permiten que nuestro programa se comporte de una forma u otra dependiendo de una condición
  - Aquí es donde cobran importancia el tipo booleano y los operadores lógicos
- 
-

# *Control de flujo: Sentencias condicionales*

- 3 sentencias: if, if-else, if-elif-else
- No hay switch-case en python
- La forma más simple de un estamento condicional es un if (del inglés si) seguido de la condición a evaluar, dos puntos (:) y en la siguiente línea e indentado, el código a ejecutar en caso de que se cumpla dicha condición

# *Control de flujo: Sentencias condicionales*

```
nombre = "alberto"
```

```
# si (if) nombre es igual a "alberto"
```

```
if nombre == "alberto":
```

```
    print ("igual!")
```

```
    print ("Gracias")
```

```
input()
```

!!!OJO CON LA INDENTACION!!!

---

---

# *Control de flujo: Sentencias condicionales*

```
nombre = "alberto"  
# si (if) nombre es igual a "alberto"  
if nombre == "alberto":  
    print ("igual!")  
print ("Gracias")  
  
input()
```

!!!El comportamiento cambia!!!

---

---

# *Control de flujo: Sentencias condicionales*

if-else

```
nombre = "alberto"
# si (if) nombre es igual a "alberto"
if nombre == "alberto":
    print ("igual!")
    print ("Gracias")
if nombre != "alberto":
    print ("distinto!")
input()
```

---

---

# *Control de flujo: Sentencias condicionales*

if-else

```
nombre = "alberto"
# si (if) nombre es igual a "alberto"
if nombre == "alberto":
    print ("igual!")
    print ("Gracias")
else:
    print ("distinto!")
input()
```

---

---

# *Control de flujo: Sentencias condicionales*

if-elif-else

```
compra=150
```

```
if compra <= 100:
```

```
    print ("Pago en efectivo")
```

```
elif compra > 100 and compra < 300:
```

```
    print ("Pago con tarjeta de débito")
```

```
else:
```

```
    print ("Pago con tarjeta de crédito")
```

```
input()
```

---

---

# *Control de flujo: Sentencias de repetición*

- 2 sentencias: while, for-in
- No hay do-while en python
- while:
  - Este bucle, se encarga de ejecutar una misma acción "mientras que" una determinada condición se cumpla. Ejemplo: Mientras que año sea menor o igual a 2013, imprimir la frase "Informes del Año año".

# *Control de flujo: Sentencias de repetición*

```
anio = 2001
while anio <= 2012:
    print ("Informes del Año", str(anio) )
    anio = anio + 1
input()
```

---

---

# *Control de flujo: Sentencias de repetición*

```
anio = 2001
while anio <= 2012:
    print ("Informes del Año", anio)
    anio = anio + 1
input()
```

---

---

# *Mezclando Sentencias de repetición y decisión*

```
while True:  
    entrada = input("> ")  
    if entrada == "adios":  
        break  
    else:  
        print (entrada)
```

---

---

# *Ejercicios*

- Imprime todos los números del 1 al 100
  - Imprime sólo los números pares del 1 al 100
  - Imprime los números del 1 al 100 de 10 en 10. Usar la función `str()` para generar cadenas con los números
  - Generar 100 nombres distintos y mostrarlos
  - Generar 100 nombres y apellidos distintos y mostrarlos
  - Leer dos números desde el teclado y mostrar el mayor.  
Ayuda: `x = int(input("Enter a number: "))`
  - Leer un número desde el teclado e imprimir si es par o impar
- 
-