

SEGURIDAD DE LA INFORMACIÓN

INTRODUCCIÓN BÁSICA A PYTHON 3

PROGRAMACIÓN PROCEDURAL

PYTHON

- Lenguaje interpretado y sencillo de alto nivel
 - Sintaxis básica, tipado dinámico, procedural / orientado a objeto, ...
- Muy utilizado
 - El 7º lenguaje más consultado en Stack Overflow
 - Javascript, HTML, CSS, SQL, Java, Shell, Python
 - El 2ª lenguaje más utilizado en Github
 - Análisis de datos, “Machine Learning”, seguridad...

Tipos Primitivos

- Tipado dinámico, “las variables no se definen”
 - Todas las variables en Python son objetos

<u>C</u>	<u>PYTHON</u>
<pre>// Las variables se declaran con un // tipo específico int i, j; i = 1; // Necesitamos terminar cada linea // con el caracter ;</pre>	<pre># Las variables se definen en su # primer uso, sin tipo decimal = 1 Real = 13.42 cadena = 'Linea 1.\nLinea 2. ' # Las líneas de código no necesitan # de terminadores explicitos</pre>

Operadores

- Mismos operadores que C, salvo excepciones

Mismas operaciones aritméticas (+, -, *, /), comparación numérica (>, <, >=, <=, ==, !=), asignación (=), asignación con operadores (+=, -=,...)

C

// Operadores ++ y -- solo en C

// Comparaciones: &&, ||, !
(a && b) || (! (b && c))

PYTHON

división decimal: //
cociente = dividendo // divisor
resto = dividendo % divisor

Comparaciones: and, or, not
(a and b) or (not (b and c))

Constantes booleanas: True, False

Bloques (if)

- Usado de sangría para agrupar declaraciones
 - Toda definición de inicio de bloque acaba con ':'

<u>C</u>	<u>PYTHON</u>
<pre>if (x < 0) { printf("Menor que cero"); } else if (x > 0) { printf("Mayor que cero"); } else { printf("Cero"); }</pre>	<pre># if condición: # ⇒declaraciones # No es necesario el uso de () x = -1 if x < 0: print('Menor que cero') elif (x > 0): print('Mayor que cero') else: print('Cero')</pre>

Bloques (for, while)

- **For** “moderno”: La variable itera dentro de un rango / lista

<u>C</u>	<u>PYTHON</u>
<pre>int i; for (i=0; i<5; i++) { printf("%d %d\n", i, i * i); }</pre> <pre>int i = 0 while (i < 5) { printf("%d\n", i); i++; }</pre>	<pre># for iterador in rango: # ⇒declaraciones for i in range(5): # 0..4 print(i, i * i)</pre> <div>Range(start, stop, [step])</div> <pre># while condición: # ⇒declaraciones i = 0 while i < 5: print(i) i = i + 1</pre>

Funciones

- No es necesario definir los tipos para la entrada y salida

<u>C</u>	<u>PYTHON</u>
<pre>#include <stdio.h> #include <math.h> void printTablaRaices(int n) { int i; for (i=1; i<=n; i++) { printf(" %2d %7.3f\n", i, sqrt(i)); } } int main() { printTablaRaices(10); return 0; }</pre>	<pre>from math import * def localsqrt(n): return sqrt(n) def printTablaRaices(n): for i in range(1,n): # 1..n-1 print("i:", i, "root:", localsqrt(i)) def main(): printTablaRaices(10) main()</pre>

Arrays (Listas) básicas

- Recordatorio: Las variables de Python son objetos

PYTHON

```
cubos = [1, 8, 27, 65, 125]
```

```
cubesList = cubos # ambas variables apuntan al mismo objeto
```

```
cubos[3] = 64 # cubesList también se modifica
```

```
kubikzahlen = cubos[:] # crea una copia
```

```
kubikzahlen[:2] # [inicio..2) = [1,8]
```

```
kubikzahlen[2:] # [2..final] = [27,65,125]
```

```
# Existen varios métodos para insertar/eliminar al principio
```

```
# y al final de un array, facilitando la creación de estructuras de pilas y colas
```

```
# .append(X), .pop(), .popleft()
```


Cadenas

- El contenido de las cadenas puede accederse como un array

PYTHON

"""

Cadena multilínea

Usado tambien como comentario o como documentación en funciones

"""

```
cadenaUno = 'simples'
```

```
cadenaDos = "dobles"
```

```
concatena = 'comillas' + cadenaUno + " y " + cadenaDos
```

```
len(concatena) # 26
```

```
concatena[0] # Primer carácter, c
```

```
concatena[-1] # Último carácter, s
```

```
concatena[2:4] # Caracteres [2...4) = "mi"
```

```
concatena[:2] # Caracteres [0...2) = "co"
```

```
# No podemos modificar una cadena existente! concatena[0] = 'u'
```

comillassimples y doubles

0

-1

... -3 -2 -1

Cadenas

PYTHON

Concatenar cadenas

```
cadena = ','.join(["dato1", "dato2", "dato3"]) # dato1,dato2,dato3
```

import io

```
output = io.StringIO()
```

```
output.write('Linea 1.\n')
```

```
print('Linea 2.', file=output)
```

```
contents = output.getvalue() # Linea 1.\nLinea 2.
```

```
output.close()
```

Manejo de caracteres

```
ord('A') # = 65 – Código Unicode (ASCII y ISO 8859-1 para 0...255)
```

```
chr(65) # ='A' – Carácter Unicode (ASCII y ISO 8859-1 para chr(0)...chr(255))
```

Referencias bibliográficas

Bibliografía básica

- “Python 3 documentation”

<https://docs.python.org/3/tutorial/>