Capacitarte

Material Imprimible

Curso de Python

Módulo 1

Qué es programar

Si buscamos una descripción en pocas palabras sobre qué es programar, encontraríamos frases como: «crear software usando un lenguaje de programación», «darle instrucciones al ordenador» o «enseñarle al ordenador a hacer algo».

Este es un curso práctico y creemos que es mejor que vayas descubriendo en qué consiste programar, realizando precisamente esa actividad. En esta sección de introducción hablaremos de forma breve sobre algunos conceptos esenciales, algo así como el «abc» que nos permita comenzar a andar.

Algoritmos, programas y lenguajes de programación

Para ayudar a entender la programación a un nivel básico se suele utilizar símiles, como las instrucciones de montaje de un mueble o una receta de cocina. En ellas explicamos cómo realizar algo a través de una serie de pasos detallados. Por ejemplo, al escribir una receta, primero hemos tenido que descomponer mentalmente el proceso de cocinar un plato en una serie de tareas con un orden lógico:

Limpiar el pescado
Echarle dos pizcas de sal
Picar 20 gr. de cebolla
Calentar 2 cucharas de aceite en una sartén
Dorar la cebolla
etc…

Luego escribiremos esos pasos. Podría ser en español, en inglés o cualquier otro idioma, pero las instrucciones seguirían siendo las mismas.

Pues bien, al desglose de un proceso en pasos detallados y ordenados le denominamos algoritmo y el fichero donde transcribimos estas instrucciones usando un lenguaje de programación concreto (Javascript, PHP, Python, Java…) para que pueda ser ejecutado por un ordenador, le llamamos programa (*).

La sintaxis de estos lenguajes de programación es bastante más simple que nuestros idiomas y utilizan un vocabulario y un conjunto de reglas mucho más reducido. Eso sí, son muy estrictas y debemos seguirlas a rajatabla para que el ordenador pueda interpretarlas sin que produzca un error.

En resumen, estos programas son un conjunto de sentencias escritas en un lenguaje de programación que le dicen al ordenador qué tareas debe realizar y en qué orden, a través de una serie de instrucciones que detallan completamente ese proceso sin ambigüedad.

Saber más (*): hay lenguajes interpretados y compilados.

En los primeros, como Javascript, un programa llamado intérprete ejecuta las sentencias a la vez que las lee del fichero de texto donde están escritas. En estos casos, a los programas también se le suele denominar scripts o guiones.

En un compilado, como Java, debemos previamente convertir el fichero de texto a una 'traducción' mediante un programa llamado compilador. Ese fichero resultante es el que se ejecutará en el ordenador.

Instalaciones

Python para Windows

https://www.python.org/downloads/windows/

Descargar instalador ejecutable de Windows x86-64

Comprobando la instalación y la versión.

Abrimos el intérprete de comando colocando en el buscador o lupa de la barra de tareas las siglas CMD y presionamos ENTER

Escribimos **python --version** y presionamos ENTER

Instalación de sublime text3

Necesitamos una aplicación para correr el programa, puede ser cualquier interprete de comandos como visual code studio o sublime text3 como en el ejemplo.

https://www.sublimetext.com/3

https://python-para-impacientes.blogspot.com/2017/02/instalar-python-paso-paso.html (Para Windows 7)

Descargamos la versión que va para nuestro sistema operativo e instalamos

Una vez el programa abierto abrimos un nuevo archivo y modificamos las herramientas, seleccionando Build System y tildando Python.

Luego nuevamente en Tools seleccionamos Build y seleccionamos la opción Python que me da la ventana al abrir.

¿Qué es Python?

Python es un lenguaje de programación de alto nivel popular. Fue creado por Guido van Rossum, y lanzado en 1991.

Se utiliza para:

- desarrollo web (lado servidor),
- desarrollo de software,
- Matemáticas
- scripting del sistema.
- Computo científico
- Inteligencia artificial

Es muy popular y es utilizado por organizaciones como Google, NASA, la CIA y Disney

Python es procesado a tiempo de ejecución por un interpretador. No hay necesidad de compilar tu programa antes de ejecutarlo

¿Qué puede hacer Python?

- Python se puede utilizar en un servidor para crear aplicaciones web.
- Python se puede utilizar junto con el software para crear flujos de trabajo.

- Python puede conectarse a sistemas de base de datos. También puede leer y modificar archivos.
- Python se puede utilizar para manejar big data y realizar matemáticas complejas.
- Python se puede utilizar para la creación rápida de prototipos o para el desarrollo de software listo para la producción.

¿Por qué Python?

- Python funciona en diferentes plataformas (Windows, Mac, Linux, Raspberry Pi, etc.).
- Python tiene una sintaxis simple similar al idioma inglés.
- Python tiene una sintaxis que permite a los desarrolladores escribir programas con menos líneas que algunos otros lenguajes de programación.
- Python se ejecuta en un sistema de intérprete, lo que significa que el código se puede ejecutar tan pronto como se escribe. Esto significa que la creación de prototipos puede ser muy rápida.
- Python se puede tratar de una manera procedimental, orientada a objetos o de una manera funcional.

Sintaxis de Python en comparación con otros lenguajes de programación

- Python fue diseñado para la legibilidad, y tiene algunas similitudes con el idioma inglés con la influencia de las matemáticas.
- Python utiliza nuevas líneas para completar un comando, a diferencia de otros lenguajes de programación que a menudo usan punto y coma o paréntesis.
- Python se basa en la sangría, utilizando espacios en blanco, para definir el ámbito; como el ámbito de bucles, funciones y clases. Otros lenguajes de programación a menudo utilizan llaves para este propósito.

Sangría de Python

La sangría hace referencia a los espacios al principio de una línea de código.

Cuando en otros lenguajes de programación la sangría en el código es sólo para legibilidad, la sangría en Python es muy importante.

Python utiliza sangría para indicar un bloque de código.

```
if 5 > 2:
  print("Cinco es mas grande que dos!")
```

Python le dará un error si omite la sangría:

```
if 5 > 2:
print("Cinco es mas grande que dos!")
```

El número de espacios depende de usted como programador, pero tiene que ser al menos uno.

```
if 5 > 2:
 print("Cinco es mas grande que dos!")
if 5 > 2:
        print("Cinco es mas grande que dos!")
```

Usted tiene que utilizar el mismo número de espacios en el mismo bloque de código, de lo contrario Python le dará un error:

```
if 5 > 2:
 print("Cinco es mas grande que dos!")
        print("Cinco es mas grande que dos!")
```

Python Variables

En Python, las variables se crean al asignarle un valor:

```
x = 5
y = "¡Hola, mundo!"
print(x)
print(y)
```

Python no tiene ningún comando para declarar una variable.

Comentarios

Python tiene capacidad de comentarios con el propósito de documentación en código.

Los comentarios comienzan con un valor de tipo (#), y Python representará el resto de la línea como un comentario:

```
#Esto es un comentario.
```

```
print("Hola, Mundo!")
```

Los comentarios se pueden colocar al final de una línea, y Python ignorará el resto de la línea:

```
print("Hola, mundo!") #Esto es un comentario.
```

Los comentarios no tienen que ser texto para explicar el código, también se puede utilizar para evitar que Python ejecute código:

```
#print("Hola, mundo!")
print("Salud, compañero!")
```

Comentarios de varias líneas

Python realmente no tiene una sintaxis para comentarios de varias líneas.

Para agregar un comentario de varias líneas, puede insertar un para cada línea:#

```
#Esto es un comentario
#Escribiendo en
#más de una línea de código
print("Hola, mundo!")
```

O, no exactamente como se pretendía, puede usar una cadena de varias líneas.

Dado que Python omitirá los literales de cadena que no están asignados a una variable, puede agregar una cadena de varias líneas (comillas triples) en el código y colocar el comentario dentro de ella:

```
"""
Esto es un comentario
Escribiendo en más de una línea de código
"""
print("Hola, mundo!")
```

Mientras la cadena no se asigne a una variable, Python leerá el código, pero luego lo ignorará y ha realizado un comentario de varias líneas.

Variables

Creación de variables

Las variables son contenedores para almacenar valores de datos.

A diferencia de otros lenguajes de programación, Python no tiene ningún comando para declarar una variable.

Se crea una variable en el momento en que se le asigna un valor por primera vez.

```
x = 5 #x es de tipo int o entero
y = "John"#y es de tipo str o cadena de caracteres
print(x)
print(y)
```

Las variables no necesitan declararse con ningún tipo en particular e incluso pueden cambiar el tipo después de que se hayan establecido.

```
x = 4
x = "Sally"
print(x)
```

Las variables de cadena se pueden declarar mediante comillas simples o dobles:

```
x = "John"
print(x)
#comillas dobles al igual que comillas simples:
x = 'John'
print(x)
```

Nombres de variables

Una variable puede tener un nombre corto (como x e y) o un nombre más descriptivo (edad, nombre, total_volumen). Reglas para variables de Python:

- Un nombre de variable debe comenzar con una letra o el carácter guion bajo
- Un nombre de variable no puede comenzar con un número
- Un nombre de variable solo puede contener caracteres alfanuméricos y guiones bajos (A-z, 0-9 y _)

- Los nombres de variables distinguen mayúsculas de minúsculas (age, Age y AGE son tres variables diferentes)

#Legal variable names:
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"

#Illegal variable names:
2myvar = "John"
my-var = "John"
my var = "John"

Recuerde que los nombres de variables distinguen mayúsculas de minúsculas

Asignar valor a varias variables

Python le permite asignar valores a varias variables en una línea:
x, y, z = "Orange", "Banana", "Cherry"

print(x)
print(y)
print(z)

Y puede asignar el *mismo* valor a varias variables en una línea:
x = y = z = "Orange"

print(x)
print(y)
print(z)

Variables de salida

La instrucción print Python se utiliza a menudo para generar variables.

Para combinar texto y una variable, Python utiliza el carácter:+

```
x = "Asombroso"
print("Python es " + x)
```

También puede utilizar el carácter + para agregar una variable a otra variable:

```
x = "Python es "
y = "asombroso"
z = x + y
print(z)
```

Para los números, el signo + funciona como un operador matemático:

```
x = 5
y = 10
print(x + y)
```

Si intenta combinar una cadena y un número, Python le dará un error:

```
x = 5
y = "John"
print(x + y)
```

Variables globales

Las variables que se crean fuera de una función (como en todos los ejemplos anteriores) se conocen como variables globales.

Las variables globales pueden ser utilizadas por todos, tanto dentro de las funciones como fuera.

```
x = "Asombroso"

def myfunc():
  print("Python es " + x)
```

```
myfunc()
```

Si crea una variable con el mismo nombre dentro de una función, esta variable será local y solo se puede utilizar dentro de la función. La variable global con el mismo nombre permanecerá como estaba, global y con el valor original.

```
x = "Asombroso"
def myfunc():
  x = "Fantastico"
  print("Python es " + x)

myfunc()

print("Python es " + x)
```

La palabra clave global

Normalmente, cuando se crea una variable dentro de una función, esa variable es local y solo se puede utilizar dentro de esa función.

Para crear una variable global dentro de una función, puede utilizar la palabra clave.global

```
def myfunc():
  global x
  x = "fantastico"

myfunc()

print("Python es " + x)
```

Además, utilice la palabra clave global si desea cambiar una variable global dentro de una función.

```
x = "Asombroso"
```

```
def myfunc():
  global x
  x = "fantastico"

myfunc()

print("Python es " + x)
```

Tipos de datos de Python

Tipos de datos integrados

En la programación, el tipo de datos es un concepto importante.

Las variables pueden almacenar datos de diferentes tipos, y diferentes tipos pueden hacer cosas diferentes.

Python tiene los siguientes tipos de datos integrados de forma predeterminada, en estas categorías:

Tipo de texto:	str
Tipos numéricos:	int, float, complex
Tipos de secuencia:	list, tuple, range

Tipo de asignación :	dict
Tipos de conjunto:	set, frozenset
Tipo booleano:	bool
Tipos binarios:	bytes, byte, arraymemoryview

Obtención del tipo de datos

Puede obtener el tipo de datos de cualquier objeto utilizando la función:type()

```
x = 5

print(type(x))
```

Configuración del tipo de datos

En Python, el tipo de datos se establece al asignar un valor a una variable:

Ejemplo	Tipo de dato
x = "Hola mundo "	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["Manzana", "banana", "Frutilla"]	list
x = ("Manzana", "banana", "Frutilla")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict

x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Configuración del tipo de datos específico

Si desea especificar el tipo de datos, puede utilizar las siguientes funciones constructoras:

Example	**Data Type**
x = str("Hola Mundo")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("manzana", "banana", "frutilla"))	list
x = tuple(("manzana", "banana", "frutilla "))	tuple
x = range(6)	range
x = dict(name="John", age=36)	dict
x = set(("manzana", "banana", "frutilla "))	set
x = frozenset(("manzana", "banana", "frutilla "))	frozenset
x = bool(5)	bool
x = bytes(5)	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Números de Python

Hay tres tipos numéricos en Python:

- int
- float
- complex

Las variables de tipos numéricos se crean al asignarles un valor:

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

Para verificar el tipo de cualquier objeto en Python, utilice la función:type()

```
x = 1
y = 2.8
z = 1j
print(type(x))
print(type(y))
print(type(z))
```

Int

Int, o entero, es un número entero, positivo o negativo, sin decimales, de longitud ilimitada.

```
x = 1
y = 35656222554887711
z = -3255522
print(type(x))
print(type(y))
print(type(z))
```

Flotante

Flotar, o "número de punto flotante" es un número, positivo o negativo, que contiene uno o más decimales.

```
x = 1.10
y = 1.0
z = -35.59
print(type(x))
print(type(y))
print(type(z))
```

El flotador también puede ser notación científica con una "e" para indicar la potencia de 10.

```
x = 35e3
y = 12E4
z = -87.7e100
print(type(x))
print(type(y))
print(type(z))
```

Complejo

Los números complejos se escriben con una "j" como parte imaginaria:

```
x = 3+5j
y = 5j
z = -5j
print(type(x))
print(type(y))
print(type(z))
```

Conversión de tipos

Puede convertir de un tipo a otro con los métodos , , y:int()float()complex()

```
#convertir de int a float:
x = float(1)

#convertir de float a int:
y = int(2.8)

#convertir de int a complejo:
z = complex(x)
print(x)
print(y)
print(z)
```

Nota: No puede convertir números complejos en otro tipo de número.

Número aleatorio

Python no tiene una función para hacer un número aleatorio, pero Python tiene un módulo incorporado llamado random que se puede utilizar para hacer números aleatorios:random()

```
import random
print(random.randrange(1, 10))
```

Conversión de Python

Especificar un tipo de variable

Puede haber ocasiones en las que desee especificar un tipo en una variable. Esto se puede hacer con el casting. Python es un lenguaje orientado a objetos y, como tal, utiliza clases para definir tipos de datos, incluidos sus tipos primitivos.

Por lo tanto, la conversión en Python se realiza mediante funciones constructoras:

- int() - construye un número entero a partir de un literal entero, un literal float (redondeando hacia abajo al número entero anterior) o un literal de cadena (siempre que la cadena representa un número entero)
- float() - construye un número flotante a partir de un literal entero, un literal float o un literal de cadena (siempre que la cadena representa un float o un entero)
- str() - construye una cadena a partir de una amplia variedad de tipos de datos, incluyendo cadenas, literales enteros y literales flotantes

```
x = int(1)
y = int(2.8)
z = int("3")
print(x)
print(y)
print(z)
```

Cadenas de Python

Literales de cadena

Los literales de cadena en Python están rodeados por comillas simples o comillas dobles.

'hola' es lo mismo que "hola".

Puede mostrar un literal de cadena con la función:print()

```
print("Hola")
print('Hola')
```

Asignar cadena a una variable

La asignación de una cadena a una variable se realiza con el nombre de la variable seguido de un signo igual y la cadena:

```
a = "Hola"
print(a)
```

Cadenas multilínea

Puede asignar una cadena de varias líneas a una variable utilizando tres comillas:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

O tres comillas simples:

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

Nota: en el resultado, los saltos de línea se insertan en la misma posición que en el código.

Las cadenas son matrices

Al igual que muchos otros lenguajes de programación populares, las cadenas en Python son matrices de bytes que representan caracteres unicode.

Sin embargo, Python no tiene un tipo de datos de carácter, un solo carácter es simplemente una cadena con una longitud de 1.

Los corchetes se pueden utilizar para acceder a los elementos de la cadena.

```
a = "Hola, Mundo!"
print(a[1])
```

Rebanar

Puede devolver un intervalo de caracteres mediante la sintaxis de sector.

Especifique el índice inicial y el índice final, separados por dos puntos, para devolver una parte de la cadena.

```
b = " Hola, Mundo!"
print(b[2:5])
```

Indexación negativa

Utilice índices negativos para iniciar el sector desde el final de la cadena:

```
b = " Hola, Mundo!"
print(b[-5:-2])
```

Longitud de la cadena

Para obtener la longitud de una cadena, utilice la función.len()

```
a = " Hola, Mundo!"
print(len(a))
```

Métodos de cadena

Python tiene un conjunto de métodos integrados que puede usar en cadenas

El método strip() elimina cualquier espacio en blanco del principio o del final de forma visual:

```
a = " Hola, Mundo! "
print(a.strip())
```

El método lower() devuelve la cadena en minúsculas:

```
a = " Hola, Mundo!"
print(a.lower())
```

El método upper() devuelve la cadena en mayúsculas:

```
a = " Hola, Mundo!"
print(a.upper())
```

El método replace() reemplaza una cadena por otra cadena:

```
a = " Hola, Mundo!"
print(a.replace("H", "J"))
```

El método split() divide la cadena en subcadenas si encuentra instancias del separador:

```
a = " Hola, Mundo!"
b = a.split(",")
print(b)
```

Comprobar cadena

Para comprobar si una determinada frase o carácter está presente en una cadena, podemos usar las palabras clave in o not in

```
txt = " La lluvia en España se mantiene principalmente en la llanura. "
x = "uvi" in txt
print(x)
```

Compruebe si la frase "uvi" NO está presente en el siguiente texto:

```
txt = " La lluvia en España se mantiene principalmente en la llanura. "
x = "uvi" not in txt
print(x)
```

Concatenación de cadenas

Para concatenar o combinar dos cadenas puede usar el operador +.

```
a = "Hola"
b = "Mundo"
c = a + b
```

```
print(c)
```

Para añadir un espacio entre ellos, agregue un:" "

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

Formato de cadena

Como aprendimos con las Variables de Python, no podemos combinar cadenas y números como este:

```
age = 36
txt = "My name is John, I am " + age
print(txt)
```

Pero podemos combinar cadenas y números mediante el uso del método!format()

El método format()toma los argumentos pasados, les da formato y los coloca en la cadena donde están los marcadores de posición: {}

Utilice el método format() para insertar números en cadenas:

```
age = 36
txt = "Mi nombre es Juan, y tengo {} años"
print(txt.format(age))
```

El método format() toma un número ilimitado de argumentos y se colocan en los marcadores de posición respectivos:

```
cantidad = 3
item= 567
precio = 49.95
myorder = "Yo quiero {} piezas del item {} por {} pesos."
```

print(myorder.format(cantidad, item, precio))

Carácter de escape

Para insertar caracteres que son ilegales en una cadena, utilice un carácter de escape.

Un carácter de escape es una barra diagonal invertida seguida del carácter que desea insertar.\

Un ejemplo de un carácter ilegal es una comilla doble dentro de una "cadena que está rodeada de comillas dobles":

txt = "Somos los llamados" vikingos "del norte"

Obtendrá un error si usa comillas dobles dentro de una cadena que está rodeada por comillas dobles:

Para solucionar este problema, utilice el carácter de escape :\"

txt = "Somos los llamados\" vikingos \" del norte"

print(txt)

Otros caracteres de escape utilizados en Python:

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

Métodos de cadena

Python tiene un conjunto de métodos integrados que puede usar en cadenas.

Nota: Todos los métodos de cadena devuelven nuevos valores. No cambian la cadena original.

Método	Descripción
capitalize()	Convierte el primer carácter a mayúsculas
casefold()	Convierte la cadena en minúsculas
center()	Devuelve una cadena centrada
count()	Devuelve el número de veces que se produce un valor especificado en una cadena
encode()	Devuelve una versión codificada de la cadena.
endswith()	Devuelve verdadero si la cadena termina con el valor especificado
expandtabs()	Establece el tamaño de la pestaña de la cadena
find()	Busca en la cadena un valor especificado y devuelve la posición donde se encontró
format()	Formatea los valores especificados en una cadena
format_map()	Formatea los valores especificados en una cadena
index()	Busca en la cadena un valor especificado y devuelve la posición donde se encontró
isalnum()	Devuelve True si todos los caracteres de la cadena son alfanuméricos
isalpha()	Devuelve True si todos los caracteres de la cadena están en el alfabeto.
isdecimal()	Devuelve True si todos los caracteres de la cadena son decimales
isdigit()	Devuelve True si todos los caracteres de la cadena son dígitos
isidentifier()	Devuelve True si la cadena es un identificado
islower()	Devuelve True si todos los caracteres de la cadena son minúsculas
isnumeric()	Devuelve True si todos los caracteres de la cadena son numéricos
isprintable()	Devuelve True si todos los caracteres de la cadena son imprimibles
isspace()	Devuelve True si todos los caracteres de la cadena son espacios en blanco
istitle()	Devuelve True si la cadena sigue las reglas de un título
isupper()	Devuelve True si todos los caracteres de la cadena son mayúsculas
join()	Une los elementos de un iterable al final de la cadena
ljust()	Devuelve una versión justificada a la izquierda de la cadena
lower()	Convierte una cadena en minúsculas
lstrip()	Devuelve una versión de corte izquierda de la cadena
maketrans()	Devuelve una tabla de traducción para ser utilizada en las traducciones.
partition()	Devuelve una tupla donde la cadena se divide en tres partes
replace()	Devuelve una cadena donde un valor especificado se reemplaza con un valor especificado

rfind()	Busca en la cadena un valor especificado y devuelve la última posición donde se encontró
rindex()	Busca en la cadena un valor especificado y devuelve la última posición donde se encontró rjust() Returns a right justified version of the string
rpartition()	Devuelve una tupla donde la cadena se divide en tres partes
rsplit()	Divide la cadena en el separador especificado y devuelve una lista
rstrip()	Devuelve una versión de corte derecha de la cadena
split()	Divide la cadena en el separador especificado y devuelve una listasplitlines()
listasplitlines()	Divide la cadena en los saltos de línea y devuelve una lista
startswith()	Devuelve verdadero si la cadena comienza con el valor especificado
strip()	Devuelve una versión recortada de la
cadenaswapcase()	Cambia los casos, las minúsculas se convierten en mayúsculas y viceversa.
title()	Convierte el primer carácter de cada palabra a mayúsculas
translate()	Devuelve una cadena traducida
upper()	Convierte una cadena en mayúsculas
zfill()	Llena la cadena con un número especificado de valores 0 al principio

Fuentes: https://www.digitallearning.es/intro-programacion-js/que-es-programar.html

https://ichi.pro/es/sintaxis-variables-y-tipos-de-datos-de-python-numeros-cadenas-booleanos-1551802064344

https://programmerclick.com/article/3285355443/