



Resumen Python 3

Víctor Mardones Bravo

Febrero de 2021

Índice general

1. Introducción a Python	1
1.1. ¿Qué es Python?	1
1.2. El Zen de Python	1
1.3. Hola mundo	2
2. Conceptos básicos	3
2.1. Comentarios	3
2.2. Operaciones aritméticas	3
2.3. La regla PEMDAS	3
2.4. Paréntesis	4
2.5. Floats	4
2.6. Exponenciación	4
2.7. Cociente y resto	5
3. Cadenas de texto	6
3.1. Strings o cadenas de caracteres	6
3.2. Cadena vacía	6
3.3. Caracteres especiales	6
3.4. Secuencias de escape	7
3.5. Caracteres Unicode	8
3.6. Strings multilínea	8
3.7. Comentarios multilínea	9
3.8. Concatenación de strings	9
3.9. Multiplicación de strings	9
3.10. Opciones del método print()	9
4. Variables	10
4.1. Asignación de variables	10
4.2. Nombre de variables válidos	10
4.3. Palabras clave	10
4.4. Operaciones con variables	10
4.5. Entrada	10
4.6. Conversión de tipos de datos	10
4.7. Operadores de asignación	10
5. Declaraciones if	11
5.1. Booleanos	11
5.2. Operadores de comparación	11
5.3. Declaración if	11
5.4. Declaración if-else	11
5.5. Declaración elif	11
5.6. Operadores lógicos	11
5.7. Precedencia de operadores lógicos	11
6. Listas	12
6.1. Creación de listas	12
6.2. Indexación de listas	12
6.3. Lista vacía	12

6.4.	Anidación de listas	12
6.5.	Operaciones con listas	12
6.6.	Funciones de listas	12
6.7.	Copiar listas	12
6.8.	Strings como listas	12
6.9.	Indexación de strings	12
6.10.	Conversión de strings a listas	12
7.	Bucles	13
7.1.	Bucles while	13
7.2.	Declaración break	13
7.3.	Declaración continue	13
7.4.	Bucle for con listas	13
7.5.	Rangos	13
7.6.	Bucle for en rangos	13
8.	Funciones	14
8.1.	¿Qué es una función?	14
8.2.	Definición de funciones	14
8.3.	Llamado de funciones	14
8.4.	Devolución de valores en una función	14
8.5.	Docstring	14
8.6.	Funciones como objetos	14
8.7.	Sobrecarga de funciones	14
8.8.	Anotaciones de tipos	14
9.	Módulos y la biblioteca estándar	15
9.1.	Módulos	15
9.2.	Alias	15
9.3.	La biblioteca estándar	15
9.4.	Módulos externos y pip	15
10.	El módulo math	16
11.	El módulo random	17
12.	Manejo de excepciones	18
12.1.	Excepciones	18
12.2.	Declaración try-except	18
12.3.	Declaración finally	18
12.4.	Levatar excepciones	18
12.5.	Aserciones	18
13.	Pruebas unitarias	19
14.	Manejo de archivos	20
14.1.	Abrir archivos	20
14.2.	Modos de apertura	20
14.3.	Cierre de archivos	20
14.4.	Lectura de archivos	20
14.5.	Escritura de archivos	20
14.6.	Declaración with	20
15.	Módulos time y datetime	21
16.	Iterables	22
16.1.	Objeto None	22
16.2.	Diccionarios	22
16.3.	Indexación de diccionarios	22
16.4.	Uso de in y not en diccionarios	22

16.5. Función <code>get()</code>	22
16.6. Función <code>keys()</code>	22
16.7. Tuplas	22
16.8. Cortes de lista	22
16.9. Cortes de tuplas	22
16.10 Subcadenas	22
16.11 Listas por compresión	22
16.12 Formateo de cadenas	22
16.13 Funciones de cadenas	22
16.14 Funciones <code>all()</code> y <code>any()</code>	22
16.15 Función <code>enumerate()</code>	23
17. Programación funcional	24
17.1. Funciones puras	24
17.2. Lambdas	24
17.3. Función <code>map()</code>	24
17.4. Función <code>filter()</code>	24
17.5. Generadores	24
17.6. Decoradores	24
17.7. Iteración	24
17.8. Recursión	24
18. Conjuntos y estructuras de datos	25
18.1. Conjuntos	25
18.2. Operaciones con conjuntos	25
18.3. Estructuras de datos	25
19. El módulo <code>itertools</code>	26
19.1. Iteradores infinitos	26
19.2. Operaciones sobre iterables	26
19.3. Funciones de combinatoria	26
20. Programación orientada a objetos	27
20.1. Programación orientada a objetos	27
20.2. Clases	27
20.3. Método <code>init</code>	27
20.4. Atributos	27
20.5. Métodos	27
20.6. Atributos de clase	27
20.7. Excepciones de clases	27
20.8. Herencia	27
20.9. Función <code>super()</code>	27
20.10 Métodos mágicos	27
20.11 Sobrecarga de operadores aritméticos	27
20.12 Sobrecarga de operadores de comparación	27
20.13 Métodos mágicos de contenedores	27
20.14 Ciclo de vida de un objeto	27
20.15 Encapsulamiento	28
21. Expresiones regulares	29
22. Empaquetamiento	30
23. Interfaz gráfica	31
24. Algoritmos de ordenamiento	32
25. Algoritmos de búsqueda	33
26. Algoritmos de matrices	34

27.Implementación de estructuras de datos	35
28.La librería NumPy	36

Capítulo 1

Introducción a Python

1.1. ¿Qué es Python?

[Python](#) es un lenguaje de programación de alto nivel, con aplicaciones en numerosas áreas, incluyendo programación web, scripting, computación científica e inteligencia artificial.

Es muy popular y usado por organizaciones como [Google](#), [la NASA](#), [la CIA](#) y [Disney](#).

No hay limitaciones en lo que se puede construir usando Python. Esto incluye aplicaciones autónomas, aplicaciones web, juegos, ciencia de datos, modelos de machine learning y mucho más.

Dato curioso: Según el creador Guido van Rossum, el nombre de Python viene de la serie de comedia británica “El Circo Volador de Monty Python”.

1.2. El Zen de Python

El Zen de Python es una colección de 19 “principios” para escribir programas de computadores que influenciaron el diseño y representan la filosofía de este lenguaje de programación.

El Zen de Python se muestra en pantalla la primera vez que se ejecute la siguiente línea.

```
1 import this
```

Después se mostrará el siguiente texto.

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
```

There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

1.3. Hola mundo

Para mostrar el texto “Hola mundo” en pantalla se puede usar la función `print()`.

```
1 print('Hola mundo')
```

Cada declaración de impresión `print()` genera texto en una nueva línea.

```
1 print('Hola')  
2 print('Mundo')
```

Capítulo 2

Conceptos básicos

2.1. Comentarios

Los comentarios son anotaciones en el código utilizadas para hacerlo más fácil de entender. No afectan la ejecución del código.

En Python, los comentarios comienzan con el símbolo `#`. Todo el texto luego de este `#` (dentro de la misma línea) es ignorado.

```
1 # Este es un comentario
```

Python no soporta comentarios multilínea, al contrario de otros lenguajes de programación.

A lo largo de este resumen, se usarán comentarios para mostrar la salida de algunas funciones, cuando sea posible.

2.2. Operaciones aritméticas

Python tiene la capacidad de realizar cálculos. Los operadores `+`, `-`, `*` y `/` representan suma, resta, multiplicación y división, respectivamente.

```
1 print(1 + 1) # 2
2 print(5 - 2) # 3
3 print(2 * 2) # 4
4 print(4 / 4) # 1.0
```

Los espacios entre los signos y los números son opcionales, pero hacen que el código sea más fácil de leer.

2.3. La regla PEMDAS

Las operaciones en Python siguen el orden dado por la regla PEMDAS:

1. Paréntesis

2. Exponentes
3. Multiplicación y división
4. Adición y Sustracción

2.4. Paréntesis

Se pueden usar paréntesis () para agrupar operaciones y hacer que estas se realicen primero, siguiendo la regla PEMDAS.

```
1 print((2 + 3) * 4)
2 # (2 + 3) * 4
3 # 5 * 4
4 # 20
```

2.5. Floats

Para representar números racionales o que no son enteros, se usa el tipo de dato float o punto flotante. Se pueden crear directamente ingresando un número con un punto decimal, o como resultado de una división.

```
1 print(0.25) # 0.25
2 print(3 / 4) # 0.75
```

Se debe tener en cuenta que los computadores no pueden almacenar perfectamente el valor de los floats, lo cual a menudo conduce a errores.

```
1 print(0.1 + 0.2) # 30000000000000004
```

El error mostrado arriba es un error clásico de la aritmética de punto flotante. Incluso tiene su propio [sitio web](#).

Al trabajar con floats, no es necesario escribir un 0 a la izquierda del punto decimal.

```
1 print(.42) # 0.42
```

Esta notación se asemeja a decir “punto cinco” en vez de “cero punto cinco”.

El resultado de cualquier operación entre floats o entre un float y un entero siempre dará como resultado un float. La división entre enteros también da como resultado un float.

2.6. Exponenciación

Otra operación soportada es la exponenciación, que es la elevación de un número a la potencia de otro. Esto se realiza usando el operador **.

Ejemplo equivalente a $2^5 = 32$.

```
1 print(2**5) # 32
```

2.7. Cociente y resto

La división entera se realiza usando el operador `//`, donde el resultado es la parte entera que queda al realizar la división, también conocida como cociente.

La división entera retorna un entero en vez de un float.

```
1 print(7 // 2) # 3
```

Para obtener el resto al realizar una división entera, se debe usar el operador módulo `%`.

```
1 print(7 % 2) # 1
```

Esta operación es equivalente a 7 mód 2 en aritmética modular.

Este operador viene de la aritmética modular, y uno de sus usos más comunes es para saber si un número es múltiplo de otro. Esto se hace revisando si el módulo al dividirlo por ese otro número es 0.

```
1 print(10 % 2) # 0, 10 es múltiplo de 2
2 print(15 % 3) # 0, 15 es múltiplo de 3
3 print(16 % 7) # 2, 16 no es múltiplo de 7
```

El caso particular módulo 2 también puede usarse para saber si un número es par o no.

Capítulo 3

Cadenas de texto

3.1. Strings o cadenas de caracteres

Las cadenas de caracteres se crean introduciendo el texto entre comillas simples `'` o dobles `''`.

```
1 print('texto')
2 print("texto")
```

Un string debe empezar y terminar con comillas del mismo tipo, no se permiten comillas mixtas.

```
1 # Strings no válidos
2 print('texto")
3 print("texto')
```

3.2. Cadena vacía

A veces es necesario inicializa un string, pero sin agregarle información. Una cadena vacía es definida como `''` o `'''`.

```
1 cadena_vacia1 = ''
2 cadena_vacia2 = '''
```

Estos strings vacíos se inicializan en variables, lo cual se verá en el capítulo siguiente.

3.3. Caracteres especiales

Algunos caracteres no se pueden incluir directamente en una cadena. Para esos casos, se debe incluir la barra diagonal inversa `\` antes de ellos.

```

1 print('\') # '
2 print('\") # "
3 print('\\') # \

```

Los caracteres ' , " y \ son especiales, porque normalmente cumplen funciones especiales dentro de strings.

Si el string se define entre comillas dobles, no es necesario poner ' para ingresar comillas simples dentro de él, y viceversa.

```

1 print("This isn't spanish")
2 print('Hola "mundo"')

```

3.4. Secuencias de escape

Las secuencias de escape también se pueden incluir usando el símbolo \ dentro de cadenas de texto. Su origen viene de las secuencias de escape usadas en las máquinas de escribir.

Algunas de las secuencias de escape más usadas son:

- Nueva línea (new line): Avanza una línea hacia adelante (salto de línea) y deja el cursor al principio de esta línea (retorno de carro). Representado por \n.

```

1 print('Hola\nMundo')
2 # Hola
3 # Mundo

```

Cualquier caracter después de \n queda en la línea siguiente.

- Tabulador horizontal (horizontal tab): Añade un salto de tabulador horizontal. Representado por \t.

```

1 print('Hola\tmundo')
2 # Hola      mundo

```

El salto de tabulador avanza hasta el siguiente “tab stop” de la misma línea.

- Retorno de carro (carriage return): Mueve el “carro” (cursor) al principio de la línea actual, eliminando todos los caracteres de esa línea. Representado por \r.

```

1 print('12345\r67')
2 # 67

```

- Retroceso (backspace): Borra el último carácter y mueve el cursor al carácter anterior. Representado por \b.

```
1 print('123\b45')
2 # 1245
```

Otras secuencias de escape que cada vez se usan menos son:

- Tabulador vertical (vertical tab): Añade un salto de tabulador vertical. Representado por `\v`.

```
1 print('Hola\vmundo')
```

La tabulación vertical avanza hasta la siguiente línea que sea una “tab stop”.

- Salto de página (form feed): Baja a la próxima “página”. Representado por `\f`.

```
1 print('Hola\fmundo')
```

Algunos programadores los usaban para separar distintas secciones de código en “páginas”.

3.5. Caracteres Unicode

Las barras diagonales inversas también se pueden usar para escribir caracteres Unicode arbitrarios. Se escriben como `\u` seguido del código del carácter Unicode (en hexadecimal).

Los códigos Unicode se aceptan sin importar que tengan mayúsculas o minúsculas.

```
1 print('\u00f1') # ñ
2 print('\u00F1') # ñ
```

El [sitio web de Unicode](#) contiene más información sobre estos caracteres y sobre este estándar. [Este sitio web](#) tiene una tabla con los códigos.

3.6. Strings multilinea

Este es un tipo especial de string, que se escribe entre comillas triples `'''` o `"""`, y que reconoce los saltos de línea sin necesidad de usar la secuencia `\n`.

```
1 print("""Esta es
2 una cadena de
3 caracteres
4 multilinea""")
5 # Esta es
6 # una cadena de
7 # caracteres
8 # multilinea
```

3.7. Comentarios multilinea

Los comentarios multilinea no existen formalmente en Python, pero se puede hacer algo parecido usando

3.8. Concatenación de strings

3.9. Multiplicación de strings

3.10. Opciones del método print()

Capítulo 4

Variables

- 4.1. Asignación de variables
- 4.2. Nombre de variables válidos
- 4.3. Palabras clave
- 4.4. Operaciones con variables
- 4.5. Entrada
- 4.6. Conversión de tipos de datos
- 4.7. Operadores de asignación

Capítulo 5

Declaraciones if

5.1. Booleanos

5.2. Operadores de comparación

5.3. Declaración if

5.4. Declaración if-else

5.5. Declaración elif

5.6. Operadores lógicos

5.7. Precedencia de operadores lógicos

Capítulo 6

Listas

- 6.1. Creación de listas
- 6.2. Indexación de listas
- 6.3. Lista vacía
- 6.4. Anidación de listas
- 6.5. Operaciones con listas
- 6.6. Funciones de listas
- 6.7. Copiar listas
- 6.8. Strings como listas
- 6.9. Indexación de strings
- 6.10. Conversión de strings a listas

Capítulo 7

Bucles

7.1. Bucles while

7.2. Declaración break

7.3. Declaración continue

7.4. Bucle for con listas

7.5. Rangos

7.6. Bucle for en rangos

Capítulo 8

Funciones

- 8.1. ¿Qué es una función?
- 8.2. Definición de funciones
- 8.3. Llamado de funciones
- 8.4. Devolución de valores en una función
- 8.5. Docstring
- 8.6. Funciones como objetos
- 8.7. Sobrecarga de funciones
- 8.8. Anotaciones de tipos

Capítulo 9

Módulos y la biblioteca estándar

9.1. Módulos

9.2. Alias

9.3. La biblioteca estándar

9.4. Módulos externos y pip

Capítulo 10

El módulo math

Capítulo 11

El módulo random

Capítulo 12

Manejo de excepciones

12.1. Excepciones

12.2. Declaración try-except

12.3. Declaración finally

12.4. Levatar excepciones

12.5. Aserciones

Capítulo 13

Pruebas unitarias

Capítulo 14

Manejo de archivos

- 14.1. Abrir archivos
- 14.2. Modos de apertura
- 14.3. Cierre de archivos
- 14.4. Lectura de archivos
- 14.5. Escritura de archivos
- 14.6. Declaración with

Capítulo 15

Módulos time y datetime

Capítulo 16

Iterables

16.1. Objeto None

16.2. Dicionarios

16.3. Indexación de diccionarios

16.4. Uso de in y not en diccionarios

16.5. Función get()

16.6. Función keys()

16.7. Tuplas

16.8. Cortes de lista

16.9. Cortes de tuplas

16.10. Subcadenas

16.11. Listas por compresión

16.12. Formateo de cadenas

16.13. Funciones de cadenas

16.14. Funciones all() y any()

16.15. Función `enumerate()`

Capítulo 17

Programación funcional

17.1. Funciones puras

17.2. Lambdas

17.3. Función `map()`

17.4. Función `filter()`

17.5. Generadores

17.6. Decoradores

17.7. Iteración

17.8. Recursión

Capítulo 18

Conjuntos y estructuras de datos

18.1. Conjuntos

18.2. Operaciones con conjuntos

18.3. Estructuras de datos

Capítulo 19

El módulo itertools

19.1. Iteradores infinitos

19.2. Operaciones sobre iterables

19.3. Funciones de combinatoria

Capítulo 20

Programación orientada a objetos

- 20.1. Programación orientada a objetos
- 20.2. Clases
- 20.3. Método `init`
- 20.4. Atributos
- 20.5. Métodos
- 20.6. Atributos de clase
- 20.7. Excepciones de clases
- 20.8. Herencia
- 20.9. Función `super()`
- 20.10. Métodos mágicos
- 20.11. Sobrecarga de operadores aritméticos
- 20.12. Sobrecarga de operadores de comparación
- 20.13. Métodos mágicos de contenedores
- 20.14. Ciclo de vida de un objeto

20.15. Encapsulamiento

Capítulo 21

Expresiones regulares

Capítulo 22

Empaquetamiento

Capítulo 23

Interfaz gráfica

Capítulo 24

Algoritmos de ordenamiento

Capítulo 25

Algoritmos de búsqueda

Capítulo 26

Algoritmos de matrices

Capítulo 27

Implementación de estructuras de datos

Capítulo 28

La librería NumPy