






¡Comenzando con Python!

Sergio García Prado

29 de septiembre de 2020

Sobre mi



- ¡Hola! Soy Sergio García Prado
- Estudios:
 -  Graduado en Ingeniería Informática
 -  Graduado en Estadística
- Experiencia Laboral:
 -  Software Engineer/ Data Scientist en Unlimiteck
- Sígueme:
 - Mail: sergio@garciparedes.me
 - Website: garciparedes.me
 - GitHub: [@garciparedes](https://github.com/garciparedes)
 - LinkedIn: Sergio García Prado

¿En qué consiste la programación?

“La programación es el proceso utilizado para idear y ordenar las acciones necesarias para realizar un proyecto, preparar ciertas máquinas o aparatos para que empiecen a funcionar en el momento y en la forma deseados o elaborar programas para su empleo en computadoras”

— R.A.E. (2014)



Similitud con recetas de cocina



- La idea de programar sobre un ordenador no es muy diferente de tareas como la de seguir una receta.
- Receta de Bizcocho:
 - 1 Mezclar ingredientes secos.
 - 2 Mezclar ingredientes húmedos.
 - 3 Combinar todos los ingredientes.
 - 4 Encender el horno.
 - 5 Amasar hasta conseguir una masa homogénea.
 - 6 Hornear durante 30 minutos.
 - 7 Dejar reposar.

Pensamiento jerárquico

- El mundo de la programación se cimienta sobre la construcción de abstracciones sobre conceptos básicos, que de manera conjunta forman una base de conocimiento más rica.
- Ejemplo:
 - Hacer bizcocho
 - Mezclar ingredientes secos, ...
 - Mezclar harina, azúcar, levadura, ...
 - Introducir harina en el recipiente, introducir azúcar en el recipiente, ...
 - Abrir paquete de harina, calcular cantidad de harina, ...
 - ...

¡Hola, Mundo!

- El término *Hello, World* consiste en la ejemplificación sobre las sentencias necesarias para un lenguaje de programación determinado, de tal manera que este imprima por consola el texto Hello, World!.
- Hola mundo en *Python*:

```
print("Hello, World!")
```



¿Por qué Python?



- *Python* es un lenguaje de programación interpretado cuya filosofía hace hincapié en la *legibilidad de su código*. bu
- En los últimos años ha sufrido un crecimiento exponencial, entre otros, por el ecosistema de tan heterogéneo que se ha ido construyendo a su alrededor: *Data Science*, *Web Development*, *Embedded Systems*, etc.

Conceptos básicos: Variables

- El concepto más básico e importante para aprender a programar es el de *variable*.
- Una variable consiste en un identificador a partir del cuál se puede acceder a la información que este tiene asignada.
- En *Python*, se puede asignar información a una variable de la siguiente forma: `variable = expression`
 - *variable* representa el identificador.
 - *expression* representa la información que contiene la variable.
- Ejemplos:
 - `x = 3`
 - `y = "Hello"`
 - `x = y`
 - `z = [x, y]`

Conceptos básicos: Tipos de Datos

- Las variables tienen asociado un determinado tipo.
- En *Python* este puede cambiar a lo largo de la ejecución (*tipado dinámico*).
- Existe un valor especial para indicar que la variable no contiene ningún valor (y por tanto tampoco tipo) denominado **None**.
- Algunos de los tipos más populares son los siguientes:
 - Texto (**str**): `x = "Hello!"`
 - Enteros (**int**): `x = 56`
 - Decimales (**float**): `x = -3.14`
 - Booleanos (**bool**): `x = True`
 - Listas (**list**): `x = [1, None, 2, "Hello", False]`
 - Diccionarios (**dict**): `x = {"a": 1, "b": 3, None: False}`

Conceptos básicos: Operaciones Básicas

- La siguiente característica de las variables es que estas poseen operaciones que modifican su estado (de manera individual o conjunta).
- Dichas operaciones dependen del tipo de información que estas contengan.
- Algunos ejemplos:
 - Suma (`int`, `float`, `str`, `list`): `z = x + y`
 - Multiplicación (`int`, `float`, `str`, `list`): `z = x * y`
 - Igualdad (`object`): `z = (x == y)`
 - Negación (`bool`): `y = not x`
 - Adicción (`list`): `x.append(y)`
 - Lectura (`dict`): `x["a"]`

Conceptos básicos: Condicionales

- Además de las operaciones entre variables, existen situaciones en que es necesario tomar decisiones diferentes dependiendo de una determinada condición, lo cuál se traduce en la ejecución de diferentes sentencias de código.
- Para ello, se utilizan sentencias **if** - **elif** - **else** de la siguiente forma:

```
1  if condition1:  
2      ...  
3  elif condition2:  
4      ...  
5      ...  
6  else:  
7      ...
```

- Ejemplo:

```
1  if x < 3:  
2      print("x less than 3")  
3  elif x == 3:  
4      print("x equal to 3")  
5  else:  
6      print("x greater than 3")
```

Conceptos básicos: Bucles

- Cuando se programa, es común repetir una determinada acción durante un número determinado de veces, ya sea debido a una condición dada o para procesar una colección de elementos.
- En *Python*, esto se lleva a cabo a través de sentencias **while** o **for**. Además, existen palabras clave para modificar su comportamiento como **break** o **continue**

```
1 while condition:
2     ...
3
4 for item in iterable:
5     ...
```

- Ejemplo:

```
1 x = 1
2 while x < 56:
3     x *= 2
4
5 for item in ["hello", "bye"]:
6     print(item)
```

Caso Práctico: Recordatorios (I)

- Para ejemplificar cómo usar *Python* así como sus instrucciones básicas se propone construir una aplicación básica de recordatorios.
- Por motivos de simplicidad, por el momento la aplicación permitirá únicamente las siguientes historias de usuario:
 - Añadir nuevos recordatorios.
 - Listar recordatorios existentes.



Caso Práctico: Recordatorios (II)

■ ¿Cómo leer los recordatorios?

```
1 message = input("What to reminder? --> ")
2 seconds = int(input("How many seconds? --> "))
3 reminder = [False, seconds, message]
4 print(f"Created a reminder: {reminder}")
```

■ ¿Dónde almacenar los recordatorios?

```
1 reminders = list()
2 reminders.append(reminder)
```

Caso Práctico: Recordatorios (III)

■ ¿Cómo mostrar los resultados?

```
1 print("Reminders:")
2 print("\tCompleted\tRemaining\tMessage")
3 for reminder in reminders:
4     print(f"\t{reminder[0]:>8}\t{reminder[1]:>9}\t{reminder[2]}")
```

■ ¿Cómo controlar el modo?

```
1 while True:
2     command = input()
3     if command == "q":
4         break
5     elif command == "a":
6         ...
7     elif command == "s":
8         ...
```

Caso Práctico: Recordatorios (IV)

- La aplicación completa tiene la siguiente forma:

```
1 reminders = list()
2 while True:
3     command = input()
4     for reminder in reminders:
5         if reminder[1] - time() >= 0:
6             break
7         reminder[0] = True
8     reminders.sort()
9     if command == "q":
10         break
11     elif command == "a":
12         message = input("What to reminder? --> ")
13         seconds = round(time()) + int(input("How many seconds? --> "))
14         reminder = [False, seconds, message]
15         reminders.append(reminder)
16         print(f"Created a reminder: {reminder}")
17         print()
18     elif command == "s":
19         print("Remainders:")
20         print("\tCompleted\tRemaining\tMessage")
21         for reminder in reminders:
22             print(f"\t{reminder[0]}\t\t\t{reminder[1] - round(time()):>9}\t\t{reminder[2]}")
23         print()
24
25 sleep(0.5)
```


¿Preguntas?

Diapositivas disponibles a través de:

<https://github.com/garciparedes/python-getting-started>