¡Bienvenida al mundo de Python!

¡Hola Mar!

Soy Alejandro Martínez, y tengo el honor de invitarte a un viaje fascinante: **aprender el lenguaje de programación Python.**

Sé que, posees una mente inquieta y curiosa, siempre buscando expandir tus horizontes y adquirir nueva habilidades. **Python es la herramienta perfecta para potenciar tus habilidades y abrir un sinfín de posibilidades.**

Imagina poder crear sitios web que cautiven a miles de personas, analizar datos para descubrir patrones ocultos, automatizar tareas tediosas para optimizar tu tiempo o incluso desarrollar aplicaciones de inteligencia artificial que revolucionen el mundo. ¡Con Python, todo esto y mucho más son posible!

Este curso está especialmente diseñado para ti, Mar. En él, encontrarás un ambiente cálido y acogedor donde podrás aprender desde cero, sin importar si no tienes experiencia previa en programación.

Avanzaremos paso a paso, a tu ritmo y con un enfoque práctico. Juntos construiremos proyectos reales que te permitirán aplicar tus conocimientos y ver el impacto tangible de Python.

En el camino, te acompañaré en cada paso del proceso. Resolveremos dudas juntos, celebraremos tus logros y superaremos los obstáculos que se presenten.

Estoy seguro de que, posees la capacidad, la determinación y la creatividad para dominar Python. Este curso te brindará las herramientas y el apoyo que necesitas para convertirte en un programador Python experto.

¡No pierdas esta oportunidad de descubrir el poder transformador de Python!

¡Te espero en este emocionante viaje de aprendizaje!

Con entusiasmo y mucho cariño,

Alejandro Martínez.

Contenido

Introducción a Python

¿Qué es Python?

Es un lenguaje de programación de **alto nivel**. Sus instrucciones están muy cercanas al **lenguaje natural** en inglés y se hace hincapié en la **legibilidad del código**.

Características del lenguaje

- Es un lenguaje de programación **interpretado y multiplataforma** cuya filosofía es una sintaxis que favorezca un **código legible.**
- Se trata de un lenguaje de programación **multi paradigma,** ya que **soporta orientación a objetos, programación imperativa** y, en menor medida, programación funcional.

Instalando Python en tu computadora

Escribiendo tus primero programas en Python

Conceptos básicos de programación

Los programas están formados por **código y datos.** A nivel de la memoria del ordenador no son más que una secuencia de bits. La interpretación de estos bits depende del lenguaje de programación, que almacena en la memoria no sólo el puro dato sino, distintos metadatos.

Tipos de datos

Nombre	Tipo	Ejemplos
Booleano	bool	`True`, `False`
Entero	int	`21`, `34500`, `34_500`
Flotante	float	`3.14`, `1.5e3`
Complejo	complex	`2j`, `3 + 5j`
Cadena	str	`'tfn'`, `'''tenerife - islas canarias'''`
Tupla	tuple	`(1, 3, 5)`
Lista	list	`['Chrome', 'Firefox']`
Conjunto	set	`set([2, 4, 6])`
Diccionario	dict	`{'Chrome': 'v79' , 'Firefox': 'v71'}`

Variables

Las variables son fundamentales ya que permiten definir **nombres** para los **valores** que tenemos en memoria y que vamos a usar en nuestros programas.

```
nombre = valor
```

Reglas para nombrar variables

En Python existen una serie de reglas para los nombres de variables:

1. Solo pueden los siguientes caracteres:
    a. Letras minúsculas.
    b. Letras mayúsculas.
    c. Dígitos.
    d. Guiones bajos.
2. Deben **empezar con una letra o guion bajo**, nunca con un dígito.
3. No pueden ser una **palabra reservada** del lenguaje.

Para conocer más de las palabras reservadas lo podemos ver de la siguiente manera:

```
Type "help", "copyright", "credits" or "license" for more information.
>>> help('keywords')

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not

>>>
```

Ejercicio

Utilizando la consola interactiva de Python, realiza las siguientes tareas.

1. Asigna un valor entero **2001** a la variable ***space_odyssey*** y muestra su valor.
2. Descubre el tipo del literal **'Que la fuerza te acompañe'**.
3. Identifica el tipo de litera ***True***.
4. Asigna la expresión ***10*3.0*** a la variable ***result*** y muestra su tipo

Mutabilidad

Las variables son nombres, no lugares. El nombrar a una variable lo que hacemos es **apuntar** a una zona de memoria en el que se representa el objeto.

```
>>> a = 5
```

Si ahora *copiamos* el valor de **a** en otra variable **b** se podría esperar que hubiera otro espacio en memoria para dicho valor, pero no, siguen siendo referencias a la memoria.

```
>>> b = a
```


La función `id()` permite conocer la dirección de memoria de un objeto en Python. A través de ella podemos comprobar que los dos objetos que hemos creado *apuntan* a la misma zona de memoria.

```
>>> id(a)
140727950232120
>>> id(b)
140727950232120
```

Cuando la zona de memoria que ocupa el objeto se puede modificar hablamos de tipos de datos **mutables**. En otro caso hablamos de tipos de datos **inmutables**.

Un ejemplo son las **listas**, son un tipo de dato **mutable** ya que podemos modificar su contenido

Inmutable	Mutable
`bool`	`list`
`int`	`set`
`float`	`dict`
`str`	
`tuple`	

Estructuras de control

Los programas se ejecutan en forma secuencial de *arriba* a *abajo.* Este orden constituye el llamado **flujo de programa**.

Ancho de código

En caso de que se quiera **romper una línea de código**, tenemos dos opciones.

```
>>> factorial = 4 * 3 * 2 * 1
```

1. Usa la barra invertida `\`

```
>>> factorail = 4 * \
... 3 * \
... 2 * \
... 1
```

2. Usar los paréntesis (…)

```
>>> factorial = (4*
... 3*
... 2*
... 1)
>>>
```


La sentencia `if`

Es una sentencia condicional. En su escritura debemos añadir una **expresión de comparación** terminando con dos puntos al final de la línea. Por ejemplo:

```
>>> temperature = 40

>>> if temperature > 35:
...     print('Aviso por alta temperatura')
...
Aviso por alta temperatura
```

> Nota
> Nótese que en Python no es necesario incluir paréntesis al escribir condiciones. Hay veces que es recomendable por claridad o por establecer prioridades.

En el caso anterior se puede ver claramente que la condición se cumple. Pero podría no ser así. Para controlar ese caso existe la sentencia ***else***.

```
>>> temperature = 20

>>> if temperature > 35:
...     print('Aviso por alta temperatura')
... else:
...     print('Parámetros normales')
...
Parámetros normales
```

Podríamos tener incluso condiciones dentro de condiciones, lo que se viene a llamar técnicamente **condiciones anidadas.**

```
>>> temperature = 28

>>> if temperature < 20:
...     if temperature < 10:
...         print('Nivel azul')
...     else:
...         print('Nivel verde')
... else:
...     if temperature < 30:
...         print('Nivel naranja')
...     else:
...         print('Nivel rojo')
...
Nivel naranja
```


Python nos ofrece una mejora en la escritura de condiciones anidadas cuando aparecen consecutivamente un *else y un if*

Construcción de la sentencia `elif` #

Ejemplo:

```
>>> temperature = 28

>>> if temperature < 20:
...     if temperature < 10:
...         print('Nivel azul')
...     else:
...         print('Nivel verde')
... elif temperature < 30:
...     print('Nivel naranja')
... else:
...     print('Nivel rojo')
...
Nivel naranja
```

Asignaciones condicionales

Supongamos que queremos asignar un nivel de riesgo de incendio en función de la temperatura. De manera clásica escribiríamos:

```
>>> temperature = 35

>>> if temperature < 30:
...     fire_risk = 'LOW'
... else:
...     fire_risk = 'HIGH'
...

>>> fire_risk
'HIGH'
```


Sin embargo, esto lo podríamos abreviar con una **asignación condicional de una única línea.**

```
>>> fire_risk = 'LOW' if temperature < 30 else 'HIGH'

>>> fire_risk
'HIGH'
```

Funciones en Python

Explorando estructuras de datos

Listas en Python: Almacenando y organizando colecciones de datos

Tuplas en Python: Inmutables y ordenadas

Diccionarios en Python: Almacenamiento de datos en clave-valor

Conjuntos en Python: Colecciones únicas y no ordenadas.

Prácticas avanzadas y aplicaciones

Módulos y paquetes en Python: Importando código para reutilizarlo

Manejo de archivos en Python: Leyendo y escribiendo datos.

Programación orientada a objetos en Python: Creando objetos y clases

Excepciones y manejo de errores en Python: Manejando situaciones inesperadas

Introducción a bibliotecas de Python