Sprint 6

Capítulo 1/5

Conclusiones clave del primer módulo

**Conclusiones clave del primer módulo**

¡Enhorabuena! Completaste oficialmente los primeros cuatro sprints del programa, completaste tus primeros proyectos y te acercaste a tu objetivo.

Para averiguar qué lecciones tienes que repasar antes de completar este proyecto responde las siguientes preguntas:

Pregunta

Recibiste datos de usuario y descubriste lagunas en los campos "Edad" e "Ingresos". ¿Cómo vas a llenar las lagunas?

Rellenaré "Edad" con el valor medio e "Ingresos" con la mediana.

Primero veré la variabilidad de los datos y luego tomaré una decisión.

Rellenaré tanto "Edad" como "Ingresos" con la media.

Rellenaré tanto "Edad" como "Ingresos" con la mediana.

¡Bien hecho!

Pregunta

¿Qué problemas surgirán cuando intentes aplicar *mean()* a una columna en la que hay tanto strings como números?

Aparecerá un mensaje de error de conversión de tipos.

No habrá errores; el promedio se calculará a partir de los datos numéricos.

No habrá un mensaje de error pero no se calcularán los resultados.

No habrá errores; el promedio se calculará a partir de todos los datos.

¡Perfecto!

Pregunta

¿Qué afirmación acerca de los duplicados completos es correcta?

Son filas que son 100% idénticas.

Son datos con diferentes registros en una columna.

Son datos con diferentes formas de palabras en una columna.

Son datos que aparecen repetidamente en una columna.

¡Perfecto!

Pregunta

Elige una situación en la que la categorización de datos sea obligatoria.

Cuando necesitas agrupar datos por ciudad.

Cuando nos interesan los datos de calificación de los estudiantes.

Cuando necesitas ordenar datos por calificación.

Cuando nos interesan los datos sobre los grupos de edad.

¡Perfecto!

Pregunta

La correlación entre los dos valores es igual a 0.78. ¿Qué significa esto?

Existe una fuerte conexión directa entre los dos valores.

Existe una correlación positiva media entre los dos valores.

Existe una fuerte correlación inversa entre los dos valores.

Existe una correlación inversa débil entre los dos valores.

No se puede declarar nada con confianza sobre el nivel de conexión a partir de estos datos.

¡Tu comprensión del material es impresionante!

Pregunta

¿Qué métodos se pueden utilizar para obtener un slice de datos?

El método *query()* y el atributo *loc*.

El método *query()* y un slice con una condición lógica.

Un slice con una condición lógica y el atributo *loc*.

El método *cut()* y el método *query()*.

¡Perfecto!

Pregunta

¿Qué tipos de hipótesis encontramos en el test de hipótesis?

Primera y segunda.

Nula y alternativa.

Principal y secundaria.

Buena y mala.

Blanco y negro.

¡Buen trabajo!

Pregunta

¿Qué es un valor *p*?

La medida de la dispersión de valores de variables aleatorias en relación con el valor esperado.

El valor promedio de una variable aleatoria a medida que el número de mediciones tiende a infinito.

El nivel de significación que separa la media del valor con el que se compara un criterio estadístico.

La probabilidad de conseguir un resultado al menos tan extremo como el que se está considerando, asumiendo que la hipótesis nula es true (verdadera).

El nivel de significación.

¡Bien hecho!

Sprint 6

Capítulo 2/5 · Faltan 2 lecciones

Trabajar con la documentación

**Cómo trabajar con la documentación**

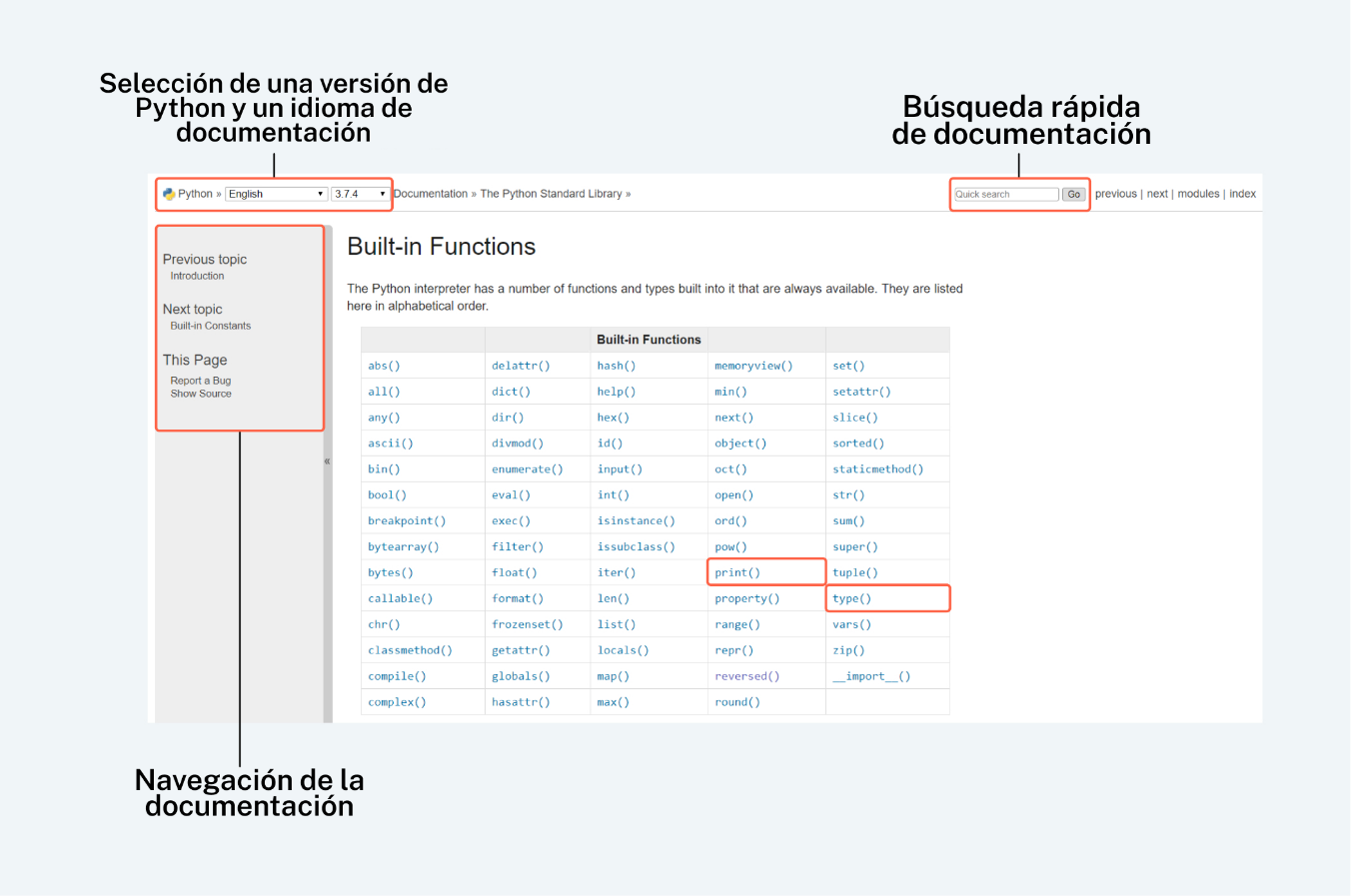
Cada lenguaje de programación tiene lo que se llama documentación (o, en la jerga de la industria, docs): una descripción de su sintaxis y funciones. Por lo general, los docs son de dominio público.

Si hay un error en tu código y no entiendes cuál es el problema, consulta la documentación.

Es fácil encontrar la documentación del lenguaje de programación necesario: simplemente busca, por ejemplo, "documentación de Python".

Si esa solicitud de búsqueda resulta demasiado genérica, puedes hacerla más descriptiva. Por ejemplo, incluye el nombre de la función: "Documentación de la función len() de Python". El primer resultado probablemente será un enlace a la documentación oficial del lenguaje: <https://docs.python.org/3/library/functions.html>.

Echemos un vistazo de cerca a esta página.



¿Cómo puedes concluir que lo que ves es documentación oficial? Es difícil dar orientaciones generales pero si estás mirando los docs para las librerías de Python y pandas, ten en cuenta que sus dominios y la forma en que está estructurada la información son similares.

Intenta usar sitios con documentación oficial. Así puedes asegurarte de que su información es correcta.

Intenta buscar información sobre métodos en docs por tu cuenta.

Enlaces a la documentación oficial:

— [Lenguaje de programación Python 3](https://docs.python.org/3/)

— [librería de pandas](https://pandas.pydata.org/pandas-docs/stable/)

— [librería Matplotlib](https://matplotlib.org/)

Pregunta

Busca información sobre el método *abs()* de *pandas* y selecciona la respuesta correcta:

Este método calcula el promedio y solo es adecuado para *Series*.

Este método encuentra el valor absoluto y solo funciona con un *DataFrame*.

Este método encuentra el valor absoluto y funciona tanto con *DataFrames* como con *Series*.

Este método encuentra la suma en la columna *DataFrame*.

¡Perfecto!

Pregunta

Encuentra información acerca de la función *sorted()* y selecciona la respuesta correcta:

La función *sorted()* ordena listas.

La función *sorted()* ordena diccionarios.

La función *sorted()* ordena cualquier estructura de datos.

¡Perfecto!

A veces es difícil entender la documentación. Si tienes problemas, existen sitios web en los que los desarrolladores y analistas hacen y responden preguntas. El recurso más popular es [Stack Overflow](https://stackoverflow.com/), que tiene discusiones en varios idiomas.

Las respuestas a menudo incluyen segmentos de código de trabajo. No importa qué tan grande sea la tentación de simplemente copiarlo y pegarlo en tu trabajo, primero tienes que ver exactamente cómo el código prestado resolverá tu tarea y asegurarte de que comprendes cómo funciona. Si el código sigue siendo un misterio para ti, inevitablemente volverás a él y tendrás que descifrarlo.

Además de Stack Overflow y sitios similares, también puedes consultar [GitHub](https://github.com/), que es una especie de almacén de códigos. Aquí, desarrolladores del mundo entero publican soluciones a sus problemas y, si tenemos suerte, también brindan descripciones detalladas explicando por qué su código está escrito de tal o cual forma. Al igual que con las soluciones publicadas en Stack Overflow, debes tener cuidado:

* Asegúrate que entiendes todos los métodos utilizados en el código.
* Comprueba si este código soluciona tu problema correctamente.

Descubrir lo que están haciendo otros desarrolladores amplía tus horizontes y mejora tus habilidades. Si la documentación u otros recursos no te ayudan a encontrar una solución, siempre puedes pedir ayuda a un colega o supervisor.

Sprint 6

Capítulo 2/5 · Última lección

Trabajar con la documentación

**Cómo leer los errores**

Los errores en Python se denominan mensajes de traza (traceback). Este nombre quiere decir que un programa escrito se verifica a cada paso ("se traza hacia atrás"). Si algo no encaja, Python genera un mensaje de error.

Veamos un ejemplo:



Los errores típicamente contienen flechas. La última flecha indica la ubicación real del error.

No tengas miedo de las tracebacks largas. De hecho, son algo bueno porque significan que el programa realizó muchas funciones antes de encontrar un error. Echa un vistazo a la última flecha: es la raíz del problema.

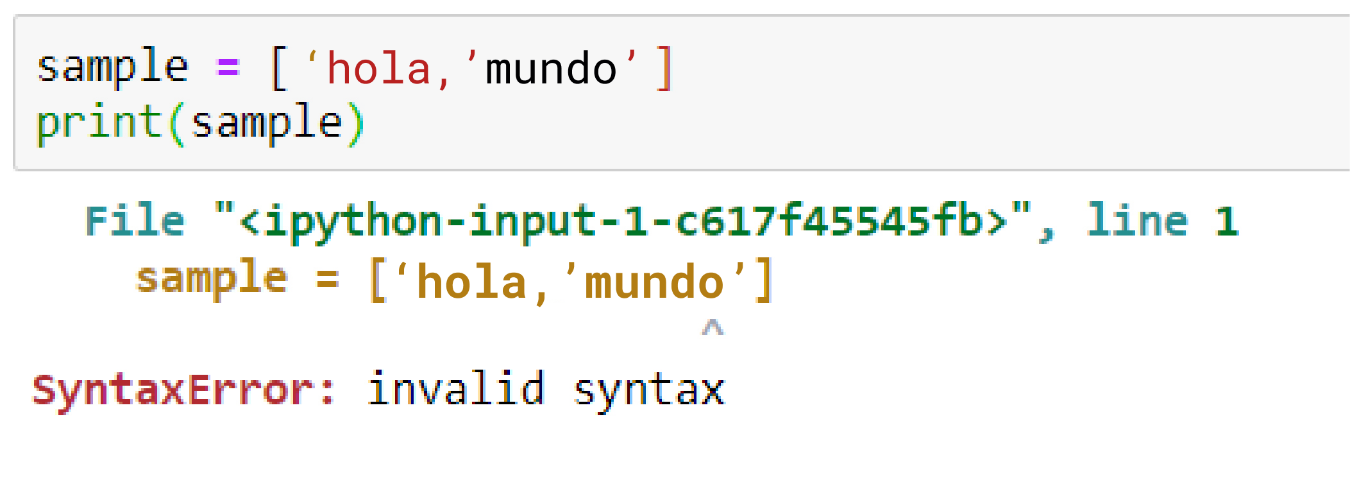
Python es una serpiente amigable y útil por lo que cuando describe un error siempre te dice su tipo. En el segundo ejemplo nos dice que tenemos un IndexError y luego brinda una explicación más detallada.

En algunos casos basta con averiguar dónde debes realizar un cambio pero a veces no hay más opción que analizar el problema en detalle. Los docs sobre los valores de error serán útiles aquí: <https://docs.python.org/3/library/exceptions.html> *(materiales en inglés)*.

Veamos algunos otros errores y aprendamos a solucionarlos:

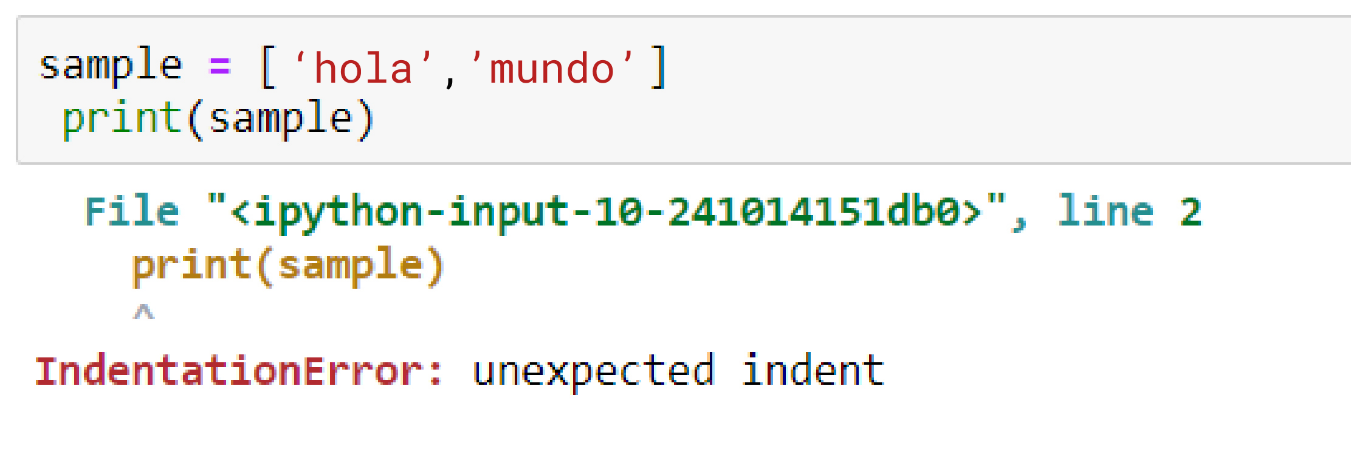
SyntaxError: sintaxis inválida

Este error significa que se te olvidó un paréntesis, dos puntos o una coma. *Python* ayuda a detectar el problema rápidamente indicando la ubicación del error y la línea donde aparece.



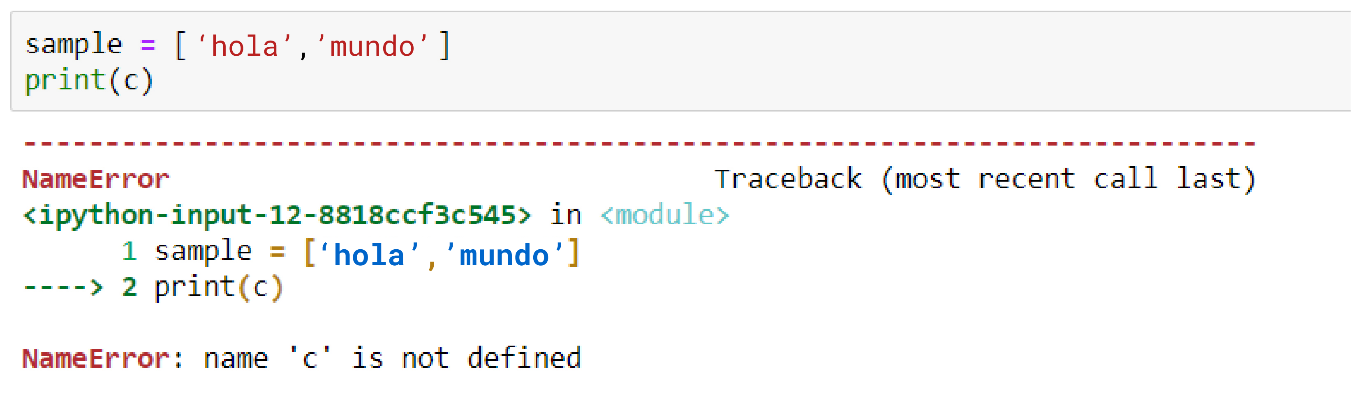
IndentationError: unexpected indent (Error de indentación: indentación inesperada)

Indentaciones... hazte amigo de este error porque lo verás a menudo. Las indentaciones juegan un papel crucial en Python y no hay nada que puedas hacer al respecto. Este error aparece cuando hay un número incorrecto de indentaciones.

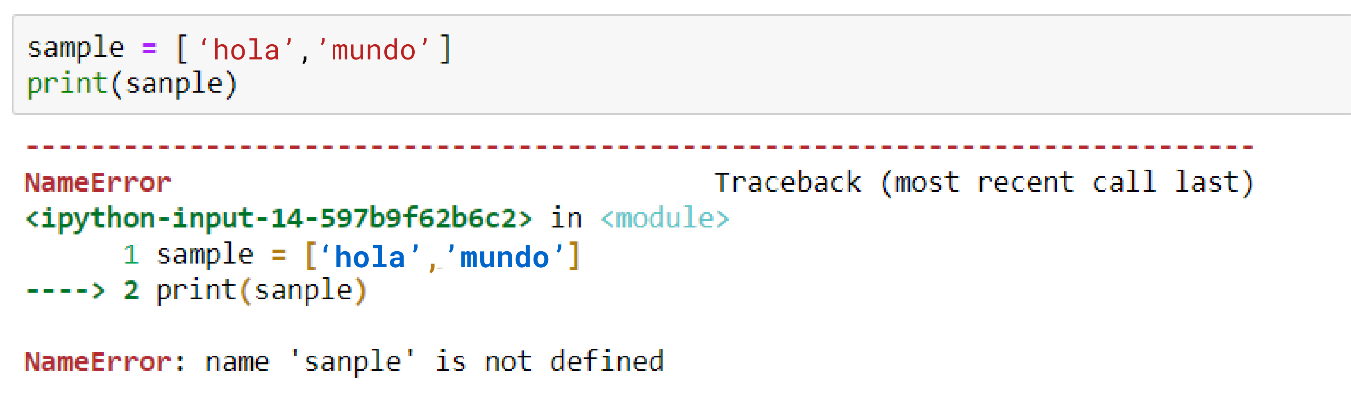


NameError: el nombre 'c' no está definido

Cuando intentas hacer referencia a una variable antes de haberle asignado un valor obtienes un *NameError*.

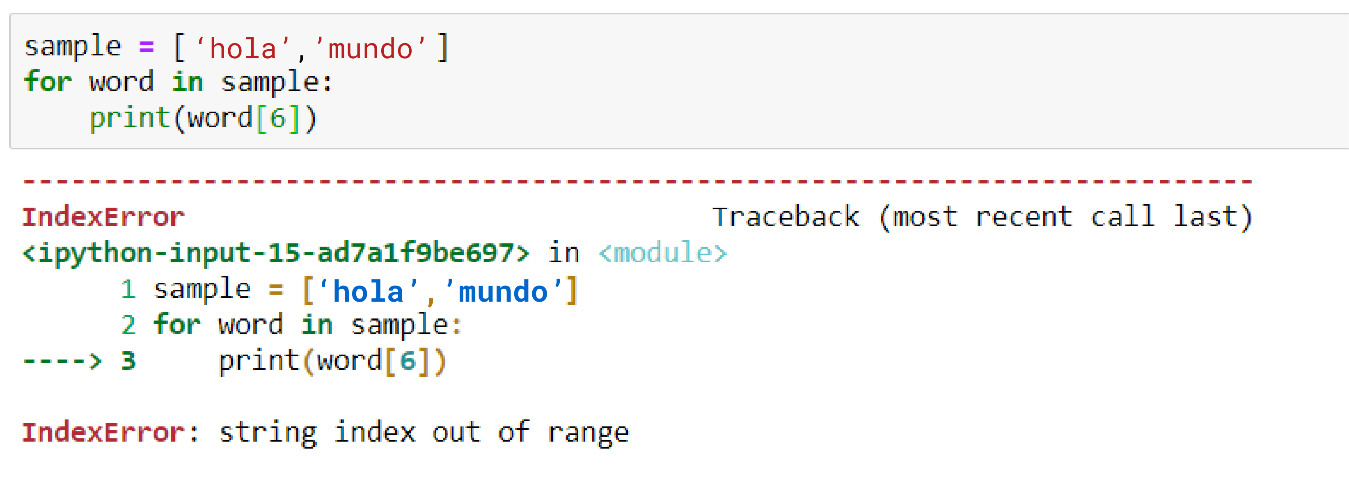


Esto sucede, por ejemplo, cuando hay una errata en el nombre de la variable:



IndexError: índice de lista/string fuera de rango

Si haces referencia a un elemento que no está en la lista o el diccionario en cuestión, el *IndexError* te lo hará saber. Comprueba de cuántos elementos consiste la lista o el diccionario y asegúrate de que estás intentando obtener el elemento correcto.



Resumen:

— Los mensajes de traceback pueden parecer bastante intimidantes pero nos brindan información útil sobre lo que salió mal. Por ejemplo, nos muestran dónde se ha producido el error y de qué tipo de error se trata.

— Los tipos de errores más comunes son *SyntaxError*, *IndentationError*, *NameError* e *IndexError*.

— *SyntaxError*: error en la sintaxis del programa.

— *IndentationError*: número incorrecto de indentaciones.

— *NameError*: un intento de hacer referencia a una variable que no ha sido definida o cuyo nombre ha sido mal escrito.

— *IndexError*: un intento de acceder a un elemento inexistente en un diccionario o una lista.

Sprint 6

Capítulo 3/5

Aprender Python más a fondo

**Introducción**

Enhorabuena por el viaje que has emprendido para dominar Python y sumergirte en el apasionante mundo de la ciencia de datos y el desarrollo de software. Tu dedicación y progreso han sido impresionantes, y está claro que has construido una base sólida en lo que respecta a los fundamentos de Python, las técnicas de manipulación de datos y las herramientas básicas de desarrollo de software.

Ahora que estás a punto de alcanzar un hito importante, queremos ofrecerte la oportunidad de profundizar en tus conocimientos de Python. Las siguientes lecciones optativas no solo mejorarán tu comprensión, sino que también te capacitarán para afrontar retos más complejos:

En esta etapa de tu viaje de aprendizaje, aprenderás lo siguiente:

* Recorrer diccionarios en bucle: descubre cómo iterar a través de diccionarios, una estructura de datos fundamental en Python.
* Estructuras de datos anidadas con diccionarios: explora el poder de anidar diccionarios dentro de listas, desbloqueando capacidades avanzadas de la organización de datos.
* Procesar listas de diccionarios: aprende a manipular y extraer, de forma eficiente, información de listas que contienen diccionarios.
* Parámetros y valores por defecto: sumérgete en el mundo de los parámetros de función y descubre cómo los valores por defecto pueden mejorar la flexibilidad de tu código.
* Argumentos posicionales y de palabra clave: aprende a distinguir entre argumentos posicionales y de palabra clave, lo que te permitirá tener más control y claridad a la hora de llamar a funciones.
* Valores de retorno: aprende a hacer tus funciones más versátiles devolviendo valores que luego podrás utilizar en tus programas.
* El objeto Series: familiarízate con el objeto Series, un componente clave de la librería Pandas a la hora de manipular datos.
* Estadística descriptiva: domina el arte de resumir e interpretar datos utilizando la estadística descriptiva, que es una habilidad crucial en el análisis de datos.

Este plan de estudios está diseñado para desafiar y enriquecer tus conocimientos de Python, proporcionándote las herramientas para abordar tareas más complejas de programación y manipulación de datos. Siéntete libre de embarcarte en este viaje, y recuerda que cada paso adelante es un testimonio de tu continuo crecimiento como experto o experta en programación Python.

Este capítulo debería tomarte entre 2 y 3.5 horas. ¡Así que empecemos!

Sprint 6

Capítulo 3/5

Aprender Python más a fondo

**Diccionarios: recorrer diccionarios en bucle**

Diagrama

Descripción generada automáticamente

Tras haber profundizado en los entresijos de los diccionarios de Python, estás a un paso de vivir una experiencia de aprendizaje mucho más enriquecedora. Ahora vamos a sumergirnos aún más en el tema de los diccionarios.

Es importante mencionar que existen varias formas de acceder a los elementos de un diccionario. Además de las formas que ya conocemos, otra forma es recorrer las claves, valores o elementos del diccionario. Esto puede ser muy útil cuando necesites realizar alguna operación en cada elemento. Por ejemplo, puede que quieras actualizar los valores de ciertas claves o crear un nuevo diccionario basado en un subconjunto de claves y valores.

En esta lección, aprenderás a usar los métodos keys(), values() e items() para recorrer elementos específicos del diccionario.

**Bucles sobre claves y valores**

Puedes usar los métodos keys() y values() para ver las claves y los valores del diccionario, respectivamente. Estos métodos son útiles cuando quieres realizar operaciones solo en las claves o los valores, o cuando quieres comprobar si una clave o un valor en particular está presente en el diccionario.

Para recorrer las claves en un diccionario, puedes usar el método keys(). Aquí te mostramos un ejemplo:

financial\_info = {

'American Express': 93.23,

'Boeing': 178.44,

'Coca-Cola': 45.15,

'Walt Disney': 119.34,

'Nike': 97.99,

'JPMorgan':96.27,

'Walmart': 130.68

}

for key in financial\_info.keys():

print(key)

Esto dará como resultado:

American Express

Boeing

Coca-Cola

Walt Disney

Nike

JPMorgan

Walmart

Para recorrer los valores en un diccionario, usemos el método values():

financial\_info = {

'American Express': 93.23,

'Boeing': 178.44,

'Coca-Cola': 45.15,

'Walt Disney': 119.34,

'Nike': 97.99,

'JPMorgan':96.27,

'Walmart': 130.68

}

for value in financial\_info.values():

print(value)

Esto dará como resultado:

93.23

178.44

45.15

119.34

97.99

96.27

130.68

Asegurémonos de que realmente comprendes estos nuevos métodos:

Pregunta

Selecciona todas las afirmaciones correctas sobre los diccionarios.

Elige tantas como quieras

El método keys() nos permite extraer todas las claves del diccionario. Podemos usar un bucle for para iterar sobre las claves de esta manera:

for key in my\_dictionary.keys():

print(key)

¡Sí, es correcto!

El método values() nos permite extraer todos los valores del diccionario. Podemos usar un bucle for para iterar sobre los valores de esta manera:

for value in my\_dictionary.values():

print(value)

Tanto el método keys() como values() se pueden aplicar a una lista.

¡Bien hecho!

Vamos a crear un programa que recorra el diccionario financial\_info y calcule el valor total de todas las acciones. Esto puede resultar útil para los inversores y las inversoras que deseen obtener una visión general rápida del rendimiento de su cartera.

Para hacer esto, necesitas utilizar un bucle for para iterar sobre el diccionario financial\_info, extraer el precio de las acciones (los precios de las acciones almacenados como valores) y actualizar la variable total\_value.

Cuando termines, muestra total\_value en la pantalla.

CódigoPYTHON

Selection deleted

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

financial\_info = {

'American Express': 93,

'Boeing': 178,

'Coca-Cola': 45,

'Walt Disney': 119,

'Nike': 97,

'JPMorgan': 96,

'Walmart': 130

}

total\_value = 0 *# el valor que actualizarás en cada iteración del bucle*

for value in financial\_info.values():

total\_value += value

*# escribe tu bucle for aquí. En cada iteración del ciclo, extrae el precio*

*# y actualiza la variable total\_value*

print(total\_value)

Pista

Enviar

Resultado

758

Acabamos de mostrarte cómo recorrer los valores en un diccionario y hacer algo con ellos. Usamos el método values() para calcular cuánto valen todas las acciones en el diccionario financial\_info. Puedes usar este truco para resolver todo tipo de problemas en los que tienes que hacer algo con cada elemento de un diccionario.

**Bucles sobre elementos**

Además de las claves y los valores, puedes usar el método items() para ver los pares (clave, valor) del diccionario. Este método puede ser útil cuando quieres iterar sobre los pares y realizar alguna operación en cada uno.

Así es como se hace:

financial\_info = {

'American Express': 93.23,

'Boeing': 178.44,

'Coca-Cola': 45.15,

'Walt Disney': 119.34,

'Nike': 97.99,

'JPMorgan':96.27,

'Walmart': 130.68

}

for key, value in financial\_info.items():

print(key, value)

Y este es el resultado:

American Express 93.23

Boeing 178.44

Coca-Cola 45.15

Walt Disney 119.34

Nike 97.99

JPMorgan 96.27

Walmart 130.68

Recorrer los diccionarios en Python es una habilidad útil cuando se trabaja con esta estructura de datos. Al usar los métodos keys(), values() y items(), puedes recorrer fácilmente los elementos específicos de un diccionario y realizar las operaciones necesarias en ellos.

Ahora, calculemos nuevamente el valor total de todas las acciones, pero esta vez con la ayuda del método items():

De nuevo, vamos a crear un programa que recorra el diccionario financial\_info y calcule el valor total de todas las acciones.

Esta vez necesitas usar el método items() en lugar de values().

Cuando termines, muestra total\_value en la pantalla.

CódigoPYTHON

Selection deleted

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

financial\_info = {

'American Express': 93,

'Boeing': 178,

'Coca-Cola': 45,

'Walt Disney': 119,

'Nike': 97,

'JPMorgan': 96,

'Walmart': 130

}

total\_value = 0 *# el valor que actualizarás en cada iteración del bucle*

for key, value in financial\_info.items():

*# escribe tu bucle for aquí. En cada iteración del ciclo, extrae el precio*

total\_value += value*# y actualiza la variable total\_value*

print(total\_value)

¡Genial! ¡Ahora dominas al menos 2 métodos!

Pista

Enviar

Resultado

758

Sprint 6

Capítulo 3/5

Aprender Python más a fondo

**Diccionarios: estructuras de datos anidadas con diccionarios**

En la lección anterior, aprendiste acerca de 3 nuevos métodos: values() que extrae valores de un diccionario, keys() que extrae claves y items() que extrae tanto las claves como los valores. También te mostramos cómo utilizar estos métodos en un bucle for. A continuación te mostramos un ejemplo:

financial\_info = {

'American Express': 93.23,

'Boeing': 178.44,

'Coca-Cola': 45.15,

'Walt Disney': 119.34,

'Nike': 97.99,

'JPMorgan':96.27,

'Walmart': 130.68

}

for value in financial\_info.values():

print(value)

93.23

178.44

45.15

119.34

97.99

96.27

130.68

En esta lección, vamos a explorar las estructuras de datos anidadas usando diccionarios. Aprenderemos cómo combinar diccionarios con listas y cómo crear e iterar sobre estas estructuras. También discutiremos cómo iterar sobre diccionarios y cómo procesar una lista de diccionarios.

A veces, los valores de un diccionario son estructuras anidadas. En el primer sprint, vimos un ejemplo de una lista de listas que representan una estructura anidada. Aquí está:

movies\_info = [

['The Shawshank Redemption', 'USA', 1994, 'drama', 142, 9.111],

['The Godfather', 'USA', 1972, 'drama, crime', 175, 8.730],

['The Dark Knight', 'USA', 2008, 'fantasy, action, thriller', 152, 8.499],

["Schindler's List", 'USA', 1993, 'drama', 195, 8.818],

['The Lord of the Rings: The Return of the King', 'New Zealand', 2003, 'fantasy, adventure, drama', 201, 8.625],

['Pulp Fiction', 'USA', 1994, 'thriller, comedy, crime', 154, 8.619],

['The Good, the Bad and the Ugly', 'Italy', 1966, 'western', 178, 8.521],

['Fight Club', 'USA', 1999, 'thriller, drama, crime', 139, 8.644],

['Harakiri', 'Japan', 1962, 'drama, action, history', 133, 8.106],

['Good Will Hunting', 'USA', 1997, 'drama, romance', 126, 8.077]

]

Los diccionarios también pueden contener estructuras anidadas. Al final de esta lección, comprenderás mejor las estructuras anidadas con diccionarios y cómo se pueden usar para procesar datos complejos.

**Diccionarios de listas**

Una estructura de datos útil en Python es un diccionario de listas, donde cada diccionario sirve como registro o entrada en la lista. Esto facilita la organización de datos de muchas maneras diferentes. Cada diccionario puede tener muchos pares clave-valor que describen diferentes atributos o propiedades del registro.

Veamos un ejemplo de un horario de autobuses:

Diagrama

Descripción generada automáticamente con confianza media

En el diccionario anterior, las claves son los números de las rutas de autobuses y los valores son listas con los horarios de salida. Puedes obtener el horario de una ruta determinada utilizando la clave.

Esta es una forma de iterar sobre un diccionario de listas. Primero, examina el código a continuación y luego ejecútalo para ver el resultado.

CódigoPYTHON

1

2

3

4

5

6

7

8

9

10

11

12

13

bus\_schedule = {

'72': ['8:00', '12:00', '17:30'],

'26': ['9:30', '15:00'],

'17': ['7:30', '12:30', '15:30']

}

*# iteramos sobre las claves y los valores del diccionario*

for route, times in bus\_schedule.items():

*# iteramos sobre los valores de la lista*

for time in times:

*# mostramos la ruta y su horario correspondiente*

print(f"Ruta {route} - Hora {time}")

Ejecutar

Resultado

Ruta 72 - Hora 8:00

Ruta 72 - Hora 12:00

Ruta 72 - Hora 17:30

Ruta 26 - Hora 9:30

Ruta 26 - Hora 15:00

Ruta 17 - Hora 7:30

Ruta 17 - Hora 12:30

Ruta 17 - Hora 15:30

Usamos el método items() para iterar sobre las claves y los valores del diccionario, y luego usamos un bucle for para iterar sobre los valores de la lista y mostrar la ruta y su horario correspondiente.

Ahora, a practicar:

Pregunta

¿Qué va a hacer el siguiente código?

bus\_schedule = {

'72': ['8:00', '12:00', '17:30'],

'26': ['9:30', '15:00'],

'17': ['7:30', '12:30', '15:30']

}

for route, times in bus\_schedule.items():

earliest\_time = times[0]

print(f'La primera hora para el autobús #{route} es {earliest\_time}')

El código recorre el diccionario bus\_schedule, extrae la primera hora de la lista e imprime el string que dice cuál es la primera hora para una ruta en particular.

¡Sí! Esto es exactamente lo que hará este código. Aquí está la salida que produce:

La primera hora para el autobús *#72 es 8:00*

La primera hora para el autobús *#26 es 9:30*

La primera hora para el autobús *#17 es 7:30*

El código recorre el diccionario bus\_schedule, extrae la última hora de la lista e imprime el string que dice cuál es la última hora para una ruta en particular.

¡Lo has entendido bien!

Los diccionarios de listas fueron la primera estructura de datos anidada que aprendiste. A continuación, convirtamos diccionarios de listas en listas de diccionarios y veamos cuál es la ventaja de esta estructura anidada.

**Listas de diccionarios y tablas**

La última estructura de datos que veremos en esta lección es la lista de diccionarios.

Anteriormente, utilizaste listas anidadas para reproducir tablas. A continuación te mostramos un ejemplo:

movies\_table = [

['The Shawshank Redemption', 'USA', 'drama', 1994, 142, 9.111],

['The Godfather', 'USA', 'drama, crime', 1972, 175, 8.730],

['The Dark Knight', 'USA', 'fantasy, action, thriller', 2008, 152, 8.499]

]

El problema con estas "tablas" es que se accede a las filas y columnas mediante índices. Por ejemplo, para obtener el año de estreno de la película de la tercera fila, tenemos que acceder al elemento en el índice 3 de la sublista 2. Ejecuta el siguiente código para ver cómo funciona:

CódigoPYTHON

1

2

3

4

5

6

7

movies\_table = [

['The Shawshank Redemption', 'USA', 'drama', 1994, 142, 9.111],

['The Godfather', 'USA', 'drama, crime', 1972, 175, 8.730],

['The Dark Knight', 'USA', 'fantasy, action, thriller', 2008, 152, 8.499]

]

print(movies\_table[2][3])

Ejecutar

Resultado

2008

Si convertimos esta lista de listas en formato de tabla, el resultado no será tan fácil de comprender:

Tabla

Descripción generada automáticamenteTabla con índices como nombres de columna

Una computadora entendería la tabla, pero un analista podría confundirse. Para obtener los datos necesarios, tendrías que memorizar el orden de las columnas, qué índice corresponde a la calificación, la duración, etc.

Para que esta sea una tabla utilizable, necesitaríamos los nombres de las columnas:

Tabla

Descripción generada automáticamenteTabla con nombres de columna

Las listas de diccionarios nos permiten reemplazar los números de las columnas con nombres (claves del diccionario). A continuación se muestra cómo se ven los datos de las películas como una lista de diccionarios. Examina el código que aparece a continuación y ejecútalo:

CódigoPYTHON

1

2

3

4

5

6

7

8

movies\_table = [

{'movie\_name':'The Shawshank Redemption', 'country':'USA', 'genre':'drama', 'year':1994, 'duration':142, 'rating':9.111},

{'movie\_name':'The Godfather', 'country':'USA', 'genre':'drama, crime', 'year':1972, 'duration':175, 'rating':8.730},

{'movie\_name':'The Dark Knight', 'country':'USA', 'genre':'fantasy, action, thriller', 'year':2008, 'duration':152, 'rating':8.499}

]

*# ahora accedemos a la columna por su nombre:*

print(movies\_table[2]['movie\_name'])

Ejecutar

Resultado

The Dark Knight

Las claves de diccionario facilitan el acceso a las columnas.

Conocer las diferentes estructuras de datos también facilita el trabajo de quienes se encargan del análisis. Además, nos sirve para seleccionar la estructura idónea para cualquier dataset. Cuando el nombre de las columnas aporta más información que su orden, es preferible aplicar una estructura con nombres.

Una lista de diccionarios reúne las ventajas de ambas estructuras:

* De los diccionarios, los nombres.
* De las listas, la capacidad de iterar sobre elementos ordenados.

¡Practiquemos un poco ahora!

Pregunta

Aquí está la tabla movies\_table que acabamos de ver en el ejemplo:

movies\_table = [

{'movie\_name':'The Shawshank Redemption', 'country':'USA', 'genre':'drama', 'year':1994, 'duration':142, 'rating':9.111},

{'movie\_name':'The Godfather', 'country':'USA', 'genre':'drama, crime', 'year':1972, 'duration':175, 'rating':8.730},

{'movie\_name':'The Dark Knight', 'country':'USA', 'genre':'fantasy, action, thriller', 'year':2008, 'duration':152, 'rating':8.499}

]

¿Qué código no dará como resultado un error cuando se ejecute? Selecciona la opción correcta.

print(movies\_table[-1]['movie\_name'])

Sí, este código no dará como resultado un error. Devolverá el nombre de la película de la última fila de la tabla.

print(movies\_table['movie\_name'][2])

print(movies\_table[0]['wordlwide\_gross'])

¡Perfecto!

¡Perfecto! A continuación, veamos qué podemos hacer con una lista de diccionarios.

Sprint 6

Capítulo 3/5

Aprender Python más a fondo

**Diccionarios: procesar listas de diccionarios**

Ahora conocemos dos estructuras de datos anidadas con diccionarios:

* Diccionarios de listas. A continuación te mostramos un ejemplo:
* bus\_schedule = {
* '72': ['8:00', '12:00', '17:30'],
* '26': ['9:30', '15:00'],
* '17': ['7:30', '12:30', '15:30']
* }

* Listas de diccionarios, que son excelentes para la representación de tablas.

Ahora veamos un ejemplo concreto en el que las listas de diccionarios resultan útiles.

Imagina que pides pizza y bebidas mediante una aplicación móvil. Los datos del pedido se envían al servidor como una lista de diccionarios donde:

* Cada diccionario es un elemento en el pedido.
* Cada clave del diccionario es un parámetro del pedido.

Mira la lista de diccionarios order a continuación para ver cómo se envían los datos del pedido:

order = [

{

'item': 'Margherita pizza',

'category': 'pizza',

'quantity': 2,

'price': 9

},

{

'item': 'Ham pizza',

'category': 'pizza',

'quantity': 1,

'price': 12

},

{

'item': 'Pepsi 1 l',

'category': 'beverages',

'quantity': 3,

'price': 2

}

]

Supongamos que queremos saber el pedido total. Para hacer esto, necesitamos sumar valores numéricos de los diccionarios para poder calcularlo. Sabemos lo siguiente:

* La lista nos permite recorrer todos los artículos del pedido utilizando un bucle.
* Los diccionarios nos dan acceso al precio y a la cantidad de cada elemento del pedido.

Nuestro objetivo es multiplicar la cantidad y el precio de cada artículo del pedido. El siguiente código hace exactamente esto. Primero, examínalo y luego ejecútalo para ver cuál es el resultado. Presta atención a los comentarios que dejamos en el código. Explican claramente lo que está pasando.

CódigoPYTHON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

order = [

{

'item': 'Margherita pizza',

'category': 'pizza',

'quantity': 2,

'price': 9

},

{

'item': 'Ham pizza',

'category': 'pizza',

'quantity': 1,

'price': 12

},

{

'item': 'Pepsi 1 l',

'category': 'beverages',

'quantity': 3,

'price': 2

}

]

*# variable del precio total del pedido*

total\_price = 0

*# iterar sobre cada diccionario de la lista*

for item in order:

*# añadir a la variable el precio del elemento multiplicado por la cantidad*

total\_price += item['price'] \* item['quantity']

print(total\_price)

Ejecutar

Resultado

36

Como resultado, obtuvimos 36 como nuestro total, pero ¿qué pasa si queremos calcular el total para una categoría en particular? Por ejemplo, ¿solo para pizzas?

Mientras trabajábamos con las sentencias condicionales y la lista de listas anidadas en el primer sprint, aprendimos sobre los filtros. Para aplicar un filtro, iteramos sobre una lista (por ejemplo, for movie in movies\_info:), comprobamos una condición (por ejemplo, if movie[4] > 180:), y luego, realizamos una acción si la condición devuelve True. Este es el código completo para un filtro aplicado a la lista de las listas anidadas:

movies\_info = [

['The Shawshank Redemption', 'USA', 1994, 'drama', 142, 9.111],

['The Godfather', 'USA', 1972, 'drama, crime', 175, 8.730],

['The Dark Knight', 'USA', 2008, 'fantasy, action, thriller', 152, 8.499],

["Schindler's List", 'USA', 1993, 'drama', 195, 8.818],

['The Lord of the Rings: The Return of the King', 'New Zealand', 2003, 'fantasy, adventure, drama', 201, 8.625],

['Pulp Fiction', 'USA', 1994, 'thriller, comedy, crime', 154, 8.619],

['The Good, the Bad and the Ugly', 'Italy', 1966, 'western', 178, 8.521],

['Fight Club', 'USA', 1999, 'thriller, drama, crime', 139, 8.644],

['Harakiri', 'Japan', 1962, 'drama, action, history', 133, 8.106],

['Good Will Hunting', 'USA', 1997, 'drama, romance', 126, 8.077]

]

movies\_filtered = [] *# lista vacía para almacenar el resultado*

for movie in movies\_info: *# recorre en bucle las filas de la tabla original*

if movie[4] > 180: *# si una película dura más de 180 minutos*

movies\_filtered.append(movie) *# agrega la fila a movies\_filtered*

for movie in movies\_filtered: *# muestra el contenido de movies\_filtered*

print(movie)

Ahora filtremos los elementos de un diccionario anidado por categoría. ¡A ver si encuentras todas las pizzas!

* El bucle iterará sobre las listas de diccionarios y obtendrá el valor de la clave 'category'.
* Si el elemento pertenece a la categoría "pizza", el bucle añadirá el diccionario completo a la nueva lista.

Examina el código que aparece a continuación y ejecútalo para ver qué produce.

CódigoPYTHON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

order = [

{

'item': 'Margherita pizza',

'category': 'pizza',

'quantity': 2,

'price': 9

},

{

'item': 'Ham pizza',

'category': 'pizza',

'quantity': 1,

'price': 12

},

{

'item': 'Pepsi 1 l',

'category': 'beverages',

'quantity': 3,

'price': 2

}

]

*# variable para almacenar el resultado*

filtered\_order = []

for item in order: *# iterar sobre cada diccionario de la lista*

if item['category'] == 'pizza': *# si la categoría es pizza...*

filtered\_order.append(item) *# agregamos el diccionario a la lista filtered\_order*

*# mostramos la lista filtrada de diccionarios*

print(filtered\_order)

Ejecutar

Resultado

[{'item': 'Margherita pizza', 'category': 'pizza', 'quantity': 2, 'price': 9}, {'item': 'Ham pizza', 'category': 'pizza', 'quantity': 1, 'price': 12}]

Ahora tenemos una lista filtrada de diccionarios. ¿No es genial?

¡Ahora es tu momento de brillar! Mira el siguiente ejercicio.

Volvamos al diccionario order. Halla el precio total para todas las pizzas en order y muéstralo. Usa el mismo método que en los ejemplos de esta lección. Si tienes dificultades para resolverlo, consulta la pista.

CódigoPYTHON

Selection deleted

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

order = [

{

'item': 'Margherita pizza',

'category': 'pizza',

'quantity': 2,

'comment': 'Add extra cheese please!',

'price': 9 *# precio por elemento*

},

{

'item': 'Ham pizza',

'category': 'pizza',

'quantity': 1,

'comment': '',

'price': 12

},

{

'item': 'Pepsi 1 l',

'category': 'beverages',

'quantity': 3,

'comment': '',

'price': 2

},

{

'item': 'Apple juice 0.5 l',

'category': 'beverages',

'quantity': 1,

'comment': '',

'price': 1

},

{

'item': 'Croissant with cheese',

'category': 'baked foods',

'quantity': 2,

'comment': '',

'price': 1

}

]

total\_price = 0 *# Inicializamos el total del precio*

for item in order: *# Iterar sobre cada diccionario de la lista*

if item['category'] == 'pizza': *# Si la categoría es pizza...*

total\_price += item['price'] \* item['quantity'] *# Sumar el precio total de esa pizza multiplicado por la cantidad*

print(total\_price) *# Mostrar el precio total*

Bucles y condiciones, listas y diccionarios... Has conseguido poner en práctica unos cuantos capítulos de este curso en solo tres líneas de código. Realmente desbloqueas el potencial de un lenguaje de programación cuando eres capaz de combinar diferentes herramientas.

Pista

Enviar

Resultado

30

Sprint 6

Capítulo 3/5

Aprender Python más a fondo

**Funciones: parámetros y valores predeterminados**

Muy bien, acabas de aprender cómo recorrer diccionarios en bucle, cómo crear estructuras anidadas utilizando diccionarios y cómo procesarlos. Es hora de profundizar en las funciones y aprender algunas técnicas más emocionantes y potentes que las utilizan.

En el segundo sprint, escribiste una función de filtro e imprimiste los resultados. Recapitulemos y veamos de nuevo las funciones que escribiste:

*# función que extrae el año y lo compara*

def filter\_by\_year(data, year):

filtered\_result = []

for row in data:

if row[2] > year:

filtered\_result.append(row)

return filtered\_result

*# función que imprime solamente el nombre de la película*

def print\_movie\_info(data):

for movie in data:

print(movie)

La función filter\_by\_year requiere dos parámetros: data y year. En el cuerpo de la función creamos una lista vacía llamada filtered\_result (resultado del filtrado). El bucle for recorre las listas anidadas de películas, y si el año de una película es superior al parámetro year, toda la lista anidada se añade a filtered\_result. Después, utilizamos la palabra clave return para devolver los resultados del filtrado.

Ahora es el momento de ver cómo puedes utilizar los valores por defecto de los parámetros para mejorar tus funciones. Recuerda que el parámetro es lo que pasamos entre los paréntesis de una función cuando la creamos. Ya hemos mencionado que en la función anterior filter\_by\_year hay dos parámetros: data y year.

Para empezar con los valores predeterminados de los parámetros, vamos a intentar ejecutar de nuevo la función omelet que escribimos en uno de los sprints anteriores. Como siempre, primero examina el código y luego ejecútalo.

CódigoPYTHON

1

2

3

4

5

def omelet(eggs\_number):

result = '¡El omelet está listo! Huevos utilizados: ' + str(eggs\_number)

return result

print(omelet())

Ejecutar

Resultado

Traceback (most recent call last):

File "main.py", line 5, in <module>

print(omelet())

TypeError: omelet() missing 1 required positional argument: 'eggs\_number'

¡Error! Verás, es necesario pasar un argumento para eggs\_number. Sin él, la función no puede convertir eggs\_number en una cadena al llamar a str().

Si no queremos que esto ocurra, tenemos que proporcionar un valor de parámetro predeterminado al definir la función. Al final de esta lección, serás capaz de hacerlo.

**Valores predeterminados de los parámetros**

Mira el video a continuación, ¡pero no olvides continuar leyendo para obtener una comprensión sólida de este tema!

Video

Imagina que estás en una cafetería y te pides un omelet, ¿crees que el camarero se confundirá si no especificas cuántos huevos quieres? Lo más probable es que te tomen el pedido con un número "estándar" de huevos. Esto es lo mismo que un parámetro por defecto.

Así es como podemos agregar un valor predeterminado a este parámetro para nuestro código:

CódigoPYTHON

1

2

3

4

5

6

*# añade un valor por defecto para eggs\_number al parámetro*

def omelet(eggs\_number=2):

result = '¡El omelet está listo! Huevos utilizados: ' + str(eggs\_number)

return result

print(omelet())

Ejecutar

Resultado

¡El omelet está listo! Huevos utilizados: 2

Al definir una función, puedes establecer un valor por defecto para cualquier parámetro utilizando =. En este ejemplo, hemos escrito eggs\_number=2.

Los parámetros con valores predeterminados también se denominan parámetros opcionales. Si no especificas su valor al llamar a la función se utilizarán los valores predeterminados.

**¡Importante!**

Al definir una función, los parámetros opcionales deben ir después de los obligatorios, aquellos que no tienen un valor por defecto asignado. En caso contrario, obtendrás un error cuando llames a la función.

Veamos cómo debería ser en el ejemplo siguiente, que tiene dos parámetros:

* cheese (queso), que es un parámetro obligatorio.
* eggs\_number (número de huevos), que es un parámetro opcional. Tiene un valor predeterminado de 2.

CódigoPYTHON

1

2

3

4

5

6

7

8

9

10

*# añade un valor por defecto para eggs\_number al parámetro*

def omelet(cheese, eggs\_number=2):

result = '¡El omelet está listo! Huevos utilizados: ' + str(eggs\_number)

if cheese == True:

result = result + ', con queso'

else:

result = result + ', sin queso'

return result

print(omelet(False))

Ejecutar

Resultado

¡El omelet está listo! Huevos utilizados: 2, sin queso

Siempre tendrás que decir si quieres queso o no, ya que es un parámetro obligatorio, pero si no indicas cuántos huevos quieres, ¡serán dos!

¡Llegó el momento de practicar!

Volvamos a nuestro ejemplo de las películas. Ahora, tienes como objetivo escribir una función, filter\_by\_genre(), con dos parámetros:

* genre=: el nombre del género, por defecto 'drama'.
* data=: datos sobre las películas, sin valor por defecto.

La función debe devolver una lista de listas que solamente contengan películas con el género pasado en el parámetro genre=.

CódigoPYTHON

Selection deleted

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

*# esta función imprime las tablas filtradas. No modificar*

def print\_movie\_info(data):

for movie in data:

print(movie)

def filter\_by\_genre(data, genre='drama'):

filtered\_movies = [] *# Lista para almacenar las películas que coinciden con el género*

for movie in data:

if genre in movie[3]: *# Comprobar si el género está en el valor de la sublista en el índice 3*

filtered\_movies.append(movie) *# Si coincide, agregar la película a la lista filtrada*

return filtered\_movies *# Devolver la lista filtrada*

*# define tu función filter\_by\_genre() aquí*

movies\_info = [

['The Shawshank Redemption', 'USA', 1994, 'drama', 142, 9.111],

['The Godfather', 'USA', 1972, 'drama, crime', 175, 8.730],

['The Dark Knight', 'USA', 2008, 'fantasy, action, thriller', 152, 8.499],

["Schindler's List", 'USA', 1993, 'drama', 195, 8.818],

['The Lord of the Rings: The Return of the King', 'New Zealand', 2003, 'fantasy, adventure, drama', 201, 8.625],

['Pulp Fiction', 'USA', 1994, 'thriller, comedy, crime', 154, 8.619],

['The Good, the Bad and the Ugly', 'Italy', 1966, 'western', 178, 8.521],

['Fight Club', 'USA', 1999, 'thriller, drama, crime', 139, 8.644],

['Harakiri', 'Japan', 1962, 'drama, action, history', 133, 8.106],

['Good Will Hunting', 'USA', 1997, 'drama, romance', 126, 8.077]

]

*# a continuación ejecutamos dos funciones para obtener resultados. No las modifiques*

movies\_filtered = filter\_by\_genre(movies\_info)

print\_movie\_info(movies\_filtered)

¡Excelente trabajo!

Pista

Enviar

Resultado

['The Shawshank Redemption', 'USA', 1994, 'drama', 142, 9.111]

['The Godfather', 'USA', 1972, 'drama, crime', 175, 8.73]

["Schindler's List", 'USA', 1993, 'drama', 195, 8.818]

['The Lord of the Rings: The Return of the King', 'New Zealand', 2003, 'fantasy, adventure, drama', 201, 8.625]

['Fight Club', 'USA', 1999, 'thriller, drama, crime', 139, 8.644]

['Harakiri', 'Japan', 1962, 'drama, action, history', 133, 8.106]

['Good Will Hunting', 'USA', 1997, 'drama, romance', 126, 8.077]

Sprint 6

Capítulo 3/5

Aprender Python más a fondo

**Funciones: valores de retorno**

Ahora conocemos los parámetros obligatorios y opcionales (o predeterminados) de una función. Recuerda que el parámetro es lo que pasamos entre los paréntesis de una función cuando la creamos. Los parámetros obligatorios deben pasarse para ejecutar una función, mientras que los opcionales, si no se pasan, serán sustituidos por valores por defecto.

**Devolución de resultados**

Habiendo terminado con los argumentos posicionales y clave, echemos un vistazo a cómo podemos mejorar nuestro uso de la palabra clave return.

Al final de esta lección, serás capaz de utilizar la palabra clave return para asignar la salida de una función con uno o más valores a una o más variables.

Mira el video a continuación, ¡pero no olvides continuar leyendo para obtener una comprensión sólida de este tema!

Video

**Finalización de la función**

La palabra clave return puede aparecer más de una vez dentro del cuerpo de una función, pero solo se ejecutará la primera llamada. Tan pronto como Python ve esta palabra clave, termina la ejecución de la función y devuelve el resultado.

Aquí estamos cascando huevos otra vez. Imagina que intentaras hacer un omelet sin huevos, ¿no sería una locura? Entonces necesitamos probarlo y devolver un mensaje diferente.

Para ello, puedes tener distintas palabras clave return. Vamos a echar un vistazo:

CódigoPYTHON

1

2

3

4

5

6

7

8

9

10

def omelet(eggs\_number=2):

if eggs\_number > 0:

return '¡El omelet está listo! Huevos utilizados: ' + str(eggs\_number)

else:

return '¿Cómo puedo hacer un omelet sin huevos?'

print('Nunca podré mostrar el resultado :(')

print(omelet(3))

print(omelet(0))

Ejecutar

Resultado

¡El omelet está listo! Huevos utilizados: 3

¿Cómo puedo hacer un omelet sin huevos?

Si la función print va después de la sentencia if, nunca se ejecutará porque el código finalizará dentro del bloque if o dentro del bloque else.

El uso de return en diferentes partes de una función es una práctica común con la que te encontrarás a lo largo de tu vida laboral. Por lo tanto, es importante asegurarse de que cualquier código escrito después de un bloque que tenga una sentencia return sea ejecutado.

**Múltiples resultados**

¿Qué hacemos si queremos que una función devuelva más de un valor? Escribamos una función que tome dos lados de un rectángulo y calcule tanto el área como el perímetro:

def area\_and\_perim(side\_1, side\_2):

area = side\_1 \* side\_2

perimeter = 2 \* (side\_1 + side\_2)

return area, perimeter

print(area\_and\_perim(2, 3))

(6, 10)

Aquí la palabra clave return va seguida por dos variables, por lo que la función devolverá ambos valores. En realidad, si quieres puedes devolver cualquier número de valores. Lo único que hay que recordar es que las distintas variables que siguen a la palabra clave return deben ir separadas por comas.

Técnicamente, una función siempre devuelve un valor. Cuando enumeramos varias variables después de return, la función las agrupa en una única estructura llamada tupla. A partir de ahí podemos descomprimirlas asignando múltiples variables al resultado de la función (en este caso, dos variables):

CódigoPYTHON

1

2

3

4

5

6

7

8

9

def find\_area\_and\_perim(side1, side2):

area = side1 \* side2

perimeter = 2 \* (side1 + side2)

return area, perimeter

*# descomprime el resultado de la función*

rec\_area, rec\_perimeter = (find\_area\_and\_perim(7, 3))

print(f'El área del rectángulo es {rec\_area}, el perímetro es {rec\_perimeter}')

Ejecutar

Resultado

El área del rectángulo es 21, el perímetro es 20

Aquí asignamos la salida de la función a las variables globales rec\_area y rec\_perimeter. Ten en cuenta que el orden es importante. La variable global rec\_area va primera, así que toma el primer valor que viene después de return, que es el valor de la variable local area.

Fíjate en que hemos evitado nombrar de la misma manera las variables locales y globales del área y del perímetro. No se producirá ningún error si les das el mismo nombre (pruébalo por tu cuenta en el cuadro de código), pero esta práctica no es recomendable, ya que dificulta tanto la lectura del código como su depuración en caso de error.

¡Ha llegado la hora de la práctica!

Pregunta

Las funciones de Python pueden devolver:

Solo un valor.

No más de dos valores.

Cualquier número de valores.

¡Correcto!

¡Lo has entendido bien!

¡Y uno más, por favor!

Pregunta

Si quieres que una función devuelva varios resultados, tienes que enumerarlos después de la palabra clave return y separarlos con:

Puntos.

Punto y coma.

Comas.

¡Exactamente! Por ejemplo: return 'the', 'final', 'lesson'.

¡Perfecto!

Sprint 6

Capítulo 3/5 · Faltan 3 lecciones

Aprender Python más a fondo

**Pandas: el objeto Series**

En el segundo sprint hemos aprendido a utilizar los métodos count(), sum() y mean() para calcular datos de forma sencilla y directa. También hemos aprendido a crear y manipular DataFrames en Pandas para obtener resultados sorprendentes al manipular y analizar datos.

En esta lección conoceremos una nueva estructura de datos: el objeto Series. Un Series es básicamente una columna de un DataFrame. Por ejemplo, cuando filtramos un DataFrame y extraemos una columna específica, esa columna se convierte en un Series.

Es importante conocer la diferencia entre un DataFrame y un Series porque, por un lado, hay métodos y atributos que son exclusivos para uno u otro y, por otro, algunos de los métodos que se pueden aplicar a ambos objetos se comportan de forma diferente según se trate de un DataFrame o de un Series.

Al final de esta lección, podrás comparar y contrastar objetos DataFrames con objetos Series, así como utilizar la indexación mediante Series. La indexación es una habilidad crucial para acceder a los elementos de un objeto Series, que con frecuencia se necesita para procesar los datos de alguna manera.

Recapitulemos lo que sabemos hasta ahora. Cuando recuperas varias columnas o filas de una tabla, obtienes una nueva tabla que es un DataFrame; puedes comprobar su tipo mediante la función type. Vamos a echar un vistazo:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

part\_df = df[['genre', 'Artist']]

print(type(part\_df))

<class 'pandas.core.frame.DataFrame'>

La tabla original tiene el mismo tipo. Échale un vistazo:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

print(type(df))

<class 'pandas.core.frame.DataFrame'>

Como puedes ver, efectivamente tienen el mismo tipo.

Supongamos ahora que solo deseas recuperar una columna. Puedes hacerlo fácilmente pasando el nombre de la columna entre corchetes: esto devolverá un objeto Series. Observa:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

part\_df = df['Artist']

print(type(part\_df))

<class 'pandas.core.series.Series'>

Hemos extraído la columna Artist y ahora la tenemos como objeto Series.

Un Series es un bloque que compone una tabla; veamos cómo se estructuran.

**DataFrames y Series**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Cada columna en un DataFrame es un objeto Series. Como ya vimos, es posible recuperar Series separados de un DataFrame.

Para acceder a los datos en un DataFrame se utilizan dos coordenadas (nombre de columna e índice), mientras que para acceder a los datos de un Series solo se utilizan los valores del índice. Por lo tanto, nos referimos a un Series como a una estructura unidimensional.

También, los objetos Series tienen un atributo name:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

part\_df = df['Artist']

print(part\_df.name)

Artist

Cuando extraes una columna de un DataFrame a un objeto Series, el atributo name adopta el nombre de la columna, como puedes ver arriba. Asimismo, cuando se construye un DataFrame a partir de Series individuales, el nombre de un objeto Series se convierte en el nombre de la columna.

Además del nombre, un Series tiene longitud, es decir, el número total de celdas. Se puede acceder a la longitud mediante el atributo size:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

part\_df = df['Artist']

print(part\_df.size)

67963

Este Series tiene 67 963 elementos, o celdas.

Por supuesto, existen otros atributos para Series, pero name y size son los que más utilizamos.

**Indexación en un Series**

La indexación de un Series es similar a la indexación de una lista. Para extraer un valor de una celda en función de su índice, simplemente pasa el valor del índice entre corchetes:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

*# obtenemos un objeto Series a partir del DataFrame*

artist = df['Artist']

*# obtenemos una celda de un Series con una sola coordenada*

print(artist[0])

Marina Rei

Como puedes ver, hemos extraído el primer elemento del Series al pasar el índice 0. Esto significa que, al igual que ocurre con las listas, la indexación de los objetos Series también empieza en 0.

La indexación de Series es similar a la indexación de DataFrame en lo que respecta a la utilización del atributo loc y una notación abreviada. Consulta la siguiente tabla para ver el listado completo:

| Tipo | Notación completa | Notación abreviada |
| --- | --- | --- |
| Un elemento | total\_play.loc[7] | total\_play[7] |
| Múltiples elementos | total\_play.loc[[5, 7, 10]] | total\_play[[5, 7, 10]] |
| Múltiples elementos consecutivos (slice) | total\_play.loc[5:10] incluyendo 10 | total\_play[5:10] sin incluir 10 |
| Todos los elementos a partir del elemento dado | total\_play.loc[1:] | total\_play[1:] |
| Todos los elementos hasta el elemento dado | total\_play.loc[:3] incluyendo 3 | total\_play[:3] sin incluir 3 |

Ahora vamos a practicar la indexación de Series.

Extrae la columna 'track' del DataFrame original y almacénala en la variable tracks. Este será tu Series. A continuación, obtén los primeros 20 elementos de este Series y guárdalos en la variable top20. Muestra la variable.

CódigoPYTHON

Selection deleted

1

2

3

4

5

6

7

8

9

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

tracks = df['track']*# escribe tu código aquí*

top20 = tracks.loc[:19]*# escribe tu código aquí*

print(top20)

Filtrar Series es bastante sencillo, ¿verdad?

Pista

Enviar

Resultado

0 Musica

1 Love Planet

2 Love Planet

3 Loving Every Minute

4 Me And My Broken Heart

5 Home

6 NaN

7 Riviera

8 The Name of This Next Song Is Called

9 Dove

10 Morning Dew

11 Bye Bye

12 Cludo

13 Räuber und Gendarm

14 NaN

15 Пойти и не вернуться. Глава 1

16 Live Life

17 General Zod

18 No Love

19 Feel Good

Name: track, dtype: object

**Filtrado de un Series**

La indexación lógica funciona también para Series y es más simple que la de DataFrames. En un Series, no es necesario indicar la columna de la que proceden los datos, basta con indicar una sola condición lógica. A continuación, se muestra un ejemplo en el que comprobamos, utilizando una notación abreviada, que el valor de 'total play' sea inferior a 20:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

total\_play = df['total play']

lower\_20 = total\_play < 20

print(lower\_20)

0 False

1 False

2 False

3 True

4 False

...

67958 False

67959 False

67960 False

67961 True

67962 True

Name: total play, Length: 67963, dtype: bool

El resultado muestra el índice de cada canción del Series junto con el valor booleano que puede ser True, si la canción ha sido reproducida durante menos de 20 segundos, o False, en caso contrario.

Si solo necesitas analizar una columna de la tabla, puede ser una buena idea almacenar esta columna en una variable separada (como lo hicimos con la variable total\_play); de esta forma, no necesitarás indicar el nombre de la columna cada vez que desees analizarla.

Es interesante que ahora podemos utilizar este nuevo Series con valores booleanos para filtrar el DataFrame original. Así es como puedes hacerlo:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

total\_play = df['total play']

lower\_20 = total\_play < 20

df\_lower20\_only = df[lower\_20]

print(df\_lower20\_only)

user\_id total play Artist

3 EF15C7BA 8.966000 NaN \

5 4166D680 3.007000 Henry Hall & His Gleneagles Hotel Band

6 F4F5677 0.100000 NaN

8 A5E0D927 3.161000 Andrew Paul Woodworth

10 D3DD8D00 8.836000 Steve Campbell

... ... ... ...

67951 A6E13637 1.318000 Julien Mier

67953 A06381D8 2.502000 Flip Grater

67956 816FBC10 2.000000 89ers

67961 DB0038A8 11.529112 Less Chapell

67962 FE8684F6 0.100000 NaN

genre track

3 dance Loving Every Minute

5 jazz Home

6 classicmetal NaN

8 pop The Name of This Next Song Is Called

10 jazz Morning Dew

... ... ...

67951 dance Nearby

67953 folk My Old Shoes

67956 dance Go Go Go

67961 pop Home

67962 NaN NaN

[33868 rows x 5 columns]

Como resultado, la tabla original ahora se filtra de forma que solo contenga filas en las que los valores de la columna 'total play' sean inferiores a 20. Y aquí tenemos 33 868 canciones de las 67 963 que figuran en nuestro conjunto de datos. Eso significa que casi la mitad de nuestras canciones no han sido prácticamente escuchadas.

Otra opción sería filtrar el Series y obtener solo los valores que cumplan con tus criterios. Por ejemplo, cuando obtenemos un Series con valores booleanos, podemos utilizarlo para filtrar el Series original total\_play:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

total\_play = df['total play']

lower\_20 = total\_play < 20

series\_lower20\_only = total\_play[lower\_20]

print(series\_lower20\_only)

3 8.966000

5 3.007000

6 0.100000

8 3.161000

10 8.836000

...

67951 1.318000

67953 2.502000

67956 2.000000

67961 11.529112

67962 0.100000

Name: total play, Length: 33868, dtype: float64

Mediante este proceso de filtrado, recuperamos solamente los valores que cumplen con nuestros criterios; en otras palabras, obtenemos un Series en el que todos los valores son menores que 20.

Ahora vamos a practicar el filtrado de un Series.

Tu objetivo ahora es filtrar el DataFrame original extrayendo solo las canciones pop. ¿Cómo lo harías?

En primer lugar, extrae la columna 'genre' del DataFrame original y almacénala en la variable genre. De este modo se creará un Series. Ahora comprueba que los valores del Series sean iguales a 'pop'.

Esta comprobación de igualdad es un poco contraintuitiva, así que consideremos otro ejemplo en el que necesitamos comprobar que los valores del Series para el nombre de la canción sean iguales a 'Andrew Paul Woodworth'. Así es como se hace:

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

tracks = df['track']

track\_check = tracks == 'Andrew Paul Woodworth'

Esperamos que este ejemplo sea suficiente para ilustrar cómo comprobamos la igualdad. Ahora, volvamos a nuestra tarea anterior.

Entonces, el primer paso consiste en extraer la columna 'genre' del DataFrame original y almacenarla en la variable genre.

A continuación, teniendo en cuenta el ejemplo anterior, hay que comprobar que los valores del Series sean iguales a 'pop'. El resultado de esta comprobación se almacena en la variable pop\_genre\_check.

Finalmente, utiliza la variable pop\_genre\_check para filtrar el DataFrame original df para incluir solo las canciones pop. Almacena este DataFrame filtrado en la variable pop\_df y muéstrala.

CódigoPYTHON

Selection deleted

1

2

3

4

5

6

7

8

9

10

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_chpt\_11.csv')

genre = df['genre']*# escribe tu código aquí*

pop\_genre\_check = genre == 'pop' *# escribe tu código aquí*

pop\_df = df[pop\_genre\_check]*# escribe tu código aquí*

print(pop\_df)

Pista

Enviar

Resultado

user\_id total play ... genre track

0 BF6EA5AF 92.851388 ... pop Musica

4 82F52E69 193.776327 ... pop Me And My Broken Heart

8 A5E0D927 3.161000 ... pop The Name of This Next Song Is Called

11 596A4517 0.000000 ... pop Bye Bye

13 79D2876C 2.000000 ... pop Räuber und Gendarm

17 AEAEF451 15.000000 ... pop General Zod

33 65608E7C 0.100000 ... pop Don't Dream It's Over

41 2F4F0630 0.100000 ... pop Cool Breeze

48 EE908C06 84.519176 ... pop Eye Of The Tiger

55 D52D00EA 8.443000 ... pop Люби меня люби

59 A40236D7 43.000000 ... pop Bitches Tune

78 20B8C4FB 211.565714 ... pop Rakkautta ja rahaa

81 FA279DD2 1.523753 ... pop Up Where We Belong

85 F6939D17 187.000000 ... pop Cold (Reprise Maroon 5 Future)

86 AC0AF747 0.100000 ... pop Slow Day

96 7F38A6F8 50.040000 ... pop When It’s Done It’s Done

97 510D9BEE 213.004754 ... pop Anna Mun Jäädä Sun Luo

98 A95CC18E 31.696000 ... pop Gravity

101 11B83119 48.000000 ... pop All I Want

103 14B73B59 31.323301 ... pop Se te ne vai

111 9C533CB1 248.450612 ... pop Lullaby to an Anxious Child

112 748BE5A3 0.100000 ... pop Sessiz Gemi

114 E3B367E9 2.000000 ... pop 110% Intro

122 66177A34 0.100000 ... pop Butterflies

125 1451927 162.075000 ... pop Toda la Noche

137 FB7E4FD8 164.931000 ... pop Famous

139 8C1DC60F 0.000000 ... pop If You Should Ever Go (Castles Turn to Sand)

144 D5EDEFF9 2.436113 ... pop Første gangen om igjen

148 24C2E9D6 213.786083 ... pop D.i.s.c.o.

164 F14E5BD3 46.220238 ... pop Рисунками

... ... ... ... ... ...

67702 CAB52D09 250.723265 ... pop I Want What She's Got

67712 9E6700D3 184.000000 ... pop Näin on käynyt sorry - Ciao Bambino Sorry

67728 322B4139 85.000000 ... pop Bird In My Hand

67743 2135677B 2.538000 ... pop Unknown

67755 728D3E60 1.661000 ... pop Strings For Yasmin

67756 9B782CAB 184.436000 ... pop Time

67760 5D9AAD37 4.809582 ... pop Easy

67792 37667BDA 104.000000 ... pop Новый год

67799 7520B6B0 249.000000 ... pop Всё что я загадаю

67800 6B7725ED 89.463755 ... pop Seven

67802 14FD167E 216.000000 ... pop Falling Without Feeling

67805 49E8BFDB 141.000000 ... pop Inima Mea

67806 3DB27335 180.610612 ... pop In The Dark

67841 8E37B76A 41.500924 ... pop Hard Times

67848 32CD3D55 12.644381 ... pop Den jag väntat på

67857 17FFC0A9 0.100000 ... pop Erkusov

67859 6ADA7378 3.705316 ... pop This Bleeding Blue Country

67863 7AF8FAC3 295.732208 ... pop Chant Like A Mantra

67866 7101BDDA 27.020000 ... pop 04:55

67870 C79D0D0 179.000000 ... pop Sans Amis

67878 CE66E39 0.919000 ... pop Yazmaları Oyalı

67884 EF163B91 18.199002 ... pop I Can't Feel My Face (Tribute to The Weeknd)

67892 CC1A9140 87.778000 ... pop Stimmen im Wind

67902 293E3300 39.691469 ... pop Rock Me Gently

67911 DA333564 283.036735 ... pop Que Me Quedes Tú

67921 2D758485 18.554472 ... pop Вера Вероника

67937 8B6704A2 68.495839 ... pop Dear Love

67952 B0DF0750 44.200000 ... pop Du hast den schönsten Arsch der Welt

67958 2E27DF51 220.551837 ... pop Girls On Fire

67961 DB0038A8 11.529112 ... pop Home

[8663 rows x 5 columns]

Y ahora la librería pandas está bajo tu dominio, ¡con todas las herramientas que necesitas para domar a esta bestia!

Sprint 6

Capítulo 3/5 · Faltan 2 lecciones

Aprender Python más a fondo

**Pandas: estadística descriptiva**

A lo largo de tu reciente viaje de aprendizaje, has profundizado en los entresijos de los diccionarios, adquiriendo conocimientos sobre cómo iterar sobre ellos, cómo gestionar estructuras anidadas y cómo manejar listas de diccionarios. Además, has explorado los aspectos más potentes de las funciones, que son los parámetros, los valores por defecto y los distintos tipos de argumentos. Esta exploración también ha abarcado el objeto Series de Pandas, lo que ha mejorado tus habilidades de manipulación de datos. Ahora, te espera otro capítulo fascinante en el que te sumergirás en la estadística descriptiva. Este paso te ayudará a profundizar en el análisis de datos, permitiéndote sacar conclusiones significativas de la información que tienes a tu alcance.

Hasta ahora, hemos aprendido a limpiar datos, eliminando los duplicados y los valores ausentes, a agrupar los datos con groupby() y a ordenarlos. Es momento de agregar más herramientas a nuestro arsenal, que nos proporcionen lo que necesitamos para hacer el análisis.

Al analizar datos, siempre es importante comprender las características básicas de los datos, también conocidas como estadísticas descriptivas. Has estudiado algunos métodos similares en el contexto de las listas, pero pandas proporciona un conjunto de métodos que facilitan la obtención de características básicas de los datos en DataFrames.

Al final de esta lección, serás capaz de utilizar cuatro métricas simples para comprender las propiedades básicas de los datos: los valores máximo y mínimo, la mediana y la media. Saber cómo usarlas te dará un muy alto nivel de sentido de cómo es la estructura de los datos. Podrás obtener el rango de los datos, una posible indicación de valores atípicos o valores sin sentido, una idea de los valores típicos y un conocimiento de la magnitud y dirección de sesgo basado en la diferencia entre la media y la mediana. ¡Así que empecemos!

**Máximo**

Encontremos los valores máximos para la cantidad de segundos que se reprodujo cada canción (esta información ahora se almacena en la columna renombrada llamada 'total\_play'). Para encontrar el valor máximo, usa el método max():

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_processed.csv')

print(df['total\_play'].max())

8638.735999999999

¡La canción más larga se reprodujo durante más de dos horas! Sería interesante saber qué pista era.

Afortunadamente, podemos averiguarlo mediante la indexación lógica. Mostramos la fila del DataFrame para la cual 'total\_play' es igual al valor máximo:

print(df[df['total\_play'] == df['total\_play'].max()])

user\_id total\_play artist genre track

8895 FE76AC63 8638.736 no\_info no\_info no\_info

Desafortunadamente, no hay información sobre esta pista, ¡pero al menos sabemos que existe!

Intenta hallar algunos números más grandes en el ejercicio.

**Ejercicios**

**Ejercicio 1**

Llevaremos a cabo un análisis parecido solo para las canciones pop. En el siguiente código, filtramos el DataFrame original para extraer todas las filas correspondientes a canciones pop. Luego, como hicimos en la lección, eliminamos todas las filas donde la columna 'total\_play' tiene menos de 30 segundos porque asumimos que tales pistas se saltan.

Busquemos la reproducción más larga de una pista de pop. Calcúlala y muestra el resultado.

CódigoPYTHON

1

2

3

4

5

6

7

8

9

10

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_processed.csv')

pop\_tracks = df[df['genre'] == 'pop']

pop\_tracks = pop\_tracks[pop\_tracks['total\_play'] > 30]

max\_dur = pop\_tracks['total\_play'].max() *# escribe tu código aquí*

print(max\_dur)

Pista

Enviar

Resultado

1158.0280450000016

**Mínimo**

El valor mínimo, es decir, la pista reproducida durante el tiempo más corto, se puede encontrar con el método min().

Ahora, si se salta una canción, 'total\_play' será 0. Nos interesan las canciones que no se saltaron de inmediato, pero que tampoco se reprodujeron durante mucho tiempo. Busquemos la canción más corta que se escuchó durante al menos medio minuto.

Podemos comenzar realizando un segmento de las pistas escuchadas durante más de medio minuto y encontrando el valor mínimo:

df\_drop\_skip = df[df['total\_play'] > 30]

print(df\_drop\_skip['total\_play'].min())

30.000497

Ahora obtendremos los nombres de las pistas que se saltaron rápidamente. Aplicaremos la indexación lógica, como cuando buscábamos el valor máximo:

print(df\_drop\_skip[df\_drop\_skip['total\_play'] == df\_drop\_skip['total\_play'].min()])

user\_id total\_play artist genre track

56351 95B68844 30.000497 Bob Baldwin jazz Can You Feel It? (Interlude)

Preferimos que nuestras listas de reproducción sean mezcladas, no revueltas. Según nuestros datos, podemos concluir que la duración de las canciones está dentro del rango de 30.004 a 4133.616327 segundos, excluyendo las canciones saltadas. Sin embargo, no sabemos mucho sobre los valores que se encuentran entre estos extremos.

Para obtener información sobre los valores en el medio, podemos calcular los valores de la mediana y la media. Echémosles un vistazo más de cerca.

**Mediana**

En estadística, la mediana es una medida que indica el valor central de un conjunto de datos. Divide el conjunto en dos partes iguales: una parte tendrá valores menores que la mediana y la otra tendrá valores mayores que la mediana.

Gráfico, Gráfico de barras

Descripción generada automáticamente

Aquí hay algunos ejemplos simples para ilustrar cómo funciona la mediana:

Número impar de valores. Supongamos que tienes el siguiente conjunto de números: [6, 3, 5, 9, 8]. Para encontrar la mediana, primero ordena los números: [3, 5, 6, 8, 9]. Dado que este dataset contiene una cantidad impar de valores, la mediana es simplemente el valor medio, que es 6.

Número par de valores. Vamos a añadir un valor más a nuestro dataset: [3, 5, 6, 8, 9, 12]. Dado que este conjunto contiene un número par de valores, la mediana es el promedio de los dos valores centrales, que son 6 y 8. Así que la mediana de este conjunto de datos será (6+8)/2 = 7.

Para resumir: si hay un número impar de valores, la mediana será el valor que se encuentra exactamente en el centro de la muestra ordenada. Si hay un número par, la mediana se encuentra como la media de los dos valores vecinos en el medio de la muestra.

Para calcular la mediana en pandas, puedes usar el método median(). Al igual que min() y max(), se puede aplicar a toda la tabla, una columna o datos agrupados.

print(df['total\_play'].median())

21.337

Ahora podemos aplicar la mediana a todos los valores de duración de escucha en nuestra tabla, excepto a las pistas escuchadas durante menos de 30 segundos, que consideramos saltadas.

df\_drop\_skip = df[df['total\_play'] > 30]

print(df\_drop\_skip['total\_play'].median())

201.96400000000003

**Media**

La media ("promedio" o "media aritmética") representa el valor promedio de un conjunto de datos.

Ahora busquemos el promedio de todos los valores usando el método mean(). Para obtener este valor manualmente, sumaríamos todas las duraciones de las pistas y las dividiríamos entre el número de pistas.

print(df\_drop\_skip['total\_play'].mean())

206.13374546931612

¡Bastante cerca de la mediana! Si la media fuera mucho mayor o menor, eso podría indicar la presencia de valores atípicos. Los valores atípicos son valores demasiado grandes o pequeños en comparación con el resto de los datos. Hablaremos de los valores atípicos más adelante en el programa, así que no te preocupes por ellos por ahora.

Por ejemplo, si descubrimos que alguien escuchó una canción durante más de 3 horas, se considerará un caso atípico y sospechoso. Bueno, podría ser una canción en repetición (y probablemente la mejor canción de la historia), pero rara vez escuchamos una pista durante unas 3 horas.

**Si la media es distinta a la mediana**

Diagrama

Descripción generada automáticamente

Sigamos hablando de los valores atípicos y cómo afectan la diferencia entre la media y la mediana. Como mencionamos anteriormente, los registros que muestran que algunas personas escuchan una canción durante mucho tiempo se consideran atípicos. Sin embargo, ¿qué pasa si tenemos muchas filas con duraciones similares en nuestros datos? Si la proporción de tales filas es significativa, esto moverá la media y la hará más grande en comparación con la mediana.

Así que nuestro mensaje clave aquí es que si ves una diferencia significativa entre la media y la mediana, esta indica que hay muchos valores atípicos en la columna que estás analizando. ¡Recuerda, esto te será útil más adelante!

**Ejercicios**

**Ejercicio 2**

Ya habíamos encontrado el valor máximo en el ejercicio anterior. Tu objetivo es calcular los valores medios y medianos para la columna 'total\_play' e imprimirlos.

CódigoPYTHON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

import pandas as pd

df = pd.read\_csv('/datasets/music\_log\_processed.csv')

pop\_tracks = df[df['genre'] == 'pop']

pop\_tracks = pop\_tracks[pop\_tracks['total\_play'] > 30]

pop\_mean = pop\_tracks['total\_play'].mean()*# escribe tu código aquí*

pop\_median = pop\_tracks['total\_play'].median()*# escribe tu código aquí*

*#max\_dur = pop\_tracks['total\_play'].max()*

*#print(df['total\_play'].median())*

*#df\_drop\_skip = df[df['total\_play'] > 30]*

*#print(df\_drop\_skip['total\_play'].median())*

*#print(df\_drop\_skip['total\_play'].mean())*

print(pop\_mean)

print(pop\_median)

De manera que si nuestra media y mediana están entre 186 y 197 segundos y nuestro valor máximo fue 1158, ¡probablemente podemos considerarlo un valor atípico!

Pista

Enviar

Resultado

186.25846123970464

197.996

¡Y eso es todo, amigos! ¡Ya tienes todo lo que necesitas para ir a tu proyecto integrado!

Sprint 6

Capítulo 3/5

Aprender Python más a fondo

**Conclusión**

¡Enhorabuena por haber llegado al final de tu viaje a través de Python! Tu compromiso y entusiasmo te han impulsado a vivir una experiencia de aprendizaje extraordinaria, y nos complace mucho poder celebrar tus logros. Cuando reflexiones sobre tus progresos, llénate de orgullo por haber desarrollado las habilidades tan diversas a lo largo del último año.

Vamos a recapitular todos estos increíbles conocimientos que has adquirido:

* Recorrer diccionarios en bucle: dominaste el arte de iterar a través de diccionarios, que es una habilidad crucial para la manipulación eficiente de datos.
* Estructuras de datos anidadas con diccionarios: descubriste lo versátil que son los diccionarios anidados, lo cual te permite organizar y gestionar estructuras de datos complejas sin problemas.
* Procesar listas de diccionarios: adquiriste destreza en la manipulación y extracción de información valiosa de listas que contienen diccionarios.
* Parámetros y valores por defecto: obtuviste un profundo conocimiento de los parámetros de función, incluyendo el uso estratégico de valores por defecto para mejorar la flexibilidad.
* Valores de retorno: mejoraste la versatilidad de tus funciones mediante la incorporación de valores de retorno, lo que facilita su integración en programas de gran tamaño.
* El objeto Series: profundizaste en el objeto Series de la librería Pandas, una potente herramienta para la manipulación y el análisis eficiente de datos.
* Estadística descriptiva: dominaste el arte de resumir e interpretar datos utilizando la estadística descriptiva, que es una habilidad vital en el ámbito del análisis de datos.

Con este completo conjunto de habilidades, lo tienes todo preparado, mejor que nunca, para embarcarte en tu primer proyecto integrado. Gracias a tu gran dedicación, dispones de las herramientas necesarias para afrontar retos complejos, analizar datos de forma eficaz y generar soluciones que tengan un impacto significativo.

Nos entusiasma mucho la siguiente etapa de tu viaje, en la que podrás aplicar tus habilidades en el mundo real. Recuerda que cada reto es una oportunidad de crecimiento y que tu capacidad para desenvolverte en los entresijos de Python y de data science te abre las puertas del éxito continuo.

Felicidades por los logros que ya has conseguido y por todos los proyectos apasionantes que te esperan en el futuro. Tu viaje no termina aquí, sino que comienza un nuevo capítulo lleno de posibilidades ilimitadas. Y este capítulo comienza con tu primer proyecto integrado, en el que aplicarás todas las herramientas que has aprendido hasta ahora.

Te has preparado mejor que nunca, ¡así que ponte manos a la obra!

## Proyecto integrado

¡Enhorabuena! Has completado oficialmente la primera parte del curso en la plataforma interactiva. Llegó la hora de reunir todo lo que has aprendido hasta el momento en tu primer proyecto integrado, un estudio de caso analítico de la vida real.

Este proyecto englobará todas las habilidades que has adquirido hasta ahora:

Diagrama

Descripción generada automáticamente con confianza bajaCódigo QR

Descripción generada automáticamenteGráfico

Descripción generada automáticamenteLogotipo

Descripción generada automáticamente con confianza media

Cuando finalices el proyecto envía tu trabajo al revisor de proyecto para su evaluación. Te dará su opinión en 48 horas. Utiliza los comentarios para realizar cambios y luego envía la nueva versión al revisor del proyecto.

Puede que recibas aún más feedback en la nueva versión. Esto es totalmente normal. Es común pasar por varios ciclos de comentarios y revisiones.

Tu proyecto se considerará completado una vez que el revisor del proyecto lo apruebe.

## Descripción del proyecto

Trabajas para la tienda online Ice que vende videojuegos por todo el mundo. Las reseñas de usuarios y expertos, los géneros, las plataformas (por ejemplo, Xbox o PlayStation) y los datos históricos sobre las ventas de juegos están disponibles en fuentes abiertas. Tienes que **identificar patrones** que determinen si un juego tiene éxito o no. Esto te permitirá detectar proyectos prometedores y planificar campañas publicitarias.

Delante de ti hay datos que se remontan a 2016. Imaginemos que es diciembre de 2016 y estás planeando una campaña para 2017.

Lo importante es adquirir experiencia de trabajo con datos. Realmente no importa si estás pronosticando las ventas de 2017 en función de los datos de 2016 o las ventas de 2027 en función de los datos de 2026.

El dataset contiene una columna "rating" que almacena la clasificación ESRB de cada juego. El Entertainment Software Rating Board (la Junta de clasificación de software de entretenimiento) evalúa el contenido de un juego y asigna una clasificación de edad como Adolescente o Adulto.

### Instrucciones para completar el proyecto

1. Abre el archivo de datos.

1.1 Ruta de archivo: */datasets/games.csv* . [Descarga el dataset](https://practicum-content.s3.us-west-1.amazonaws.com/datasets/games.csv)

1.2 Estudia la información general

2. Prepara los datos

* 1. Reemplaza los nombres de las columnas **(ponlos en minúsculas).**
  2. Convierte los datos en los tipos necesarios.
  3. Describe las columnas en las que los tipos de datos han sido cambiados y explica por qué.
  4. Si es necesario, elige la manera de tratar los valores ausentes:
  5. Explica por qué rellenaste los valores ausentes como lo hiciste o por qué decidiste dejarlos en blanco.
  6. ¿Por qué crees que los valores están ausentes? Brinda explicaciones posibles.
  7. Presta atención a la abreviatura TBD: significa "to be determined" (a determinar). Especifica cómo piensas manejar estos casos.

2.8Calcula las ventas totales (la suma de las ventas en todas las regiones) para cada juego y coloca estos valores en una columna separada.

3 Analiza los datos

3.1 Mira cuántos juegos fueron lanzados en diferentes años.

3.1.1 ¿Son significativos los datos de cada período?

* 1. Observa cómo varían las ventas de una plataforma a otra.
     1. Elige las plataformas con las mayores ventas totales
     2. construye una distribución Estándar basada en los datos de cada año.
     3. Busca las plataformas que solían ser populares pero que ahora no tienen ventas.
     4. ¿Cuánto tardan generalmente las nuevas plataformas en aparecer y las antiguas en desaparecer?
  2. Determina para qué período debes tomar datos. Para hacerlo mira tus respuestas a las preguntas anteriores. Los datos deberían permitirte construir un modelo para 2017.
  3. Trabaja solo con los datos que consideras relevantes. Ignora los datos de años anteriores.(COMENTARIO 2017 ESTA MAS COMPLETA)
  4. ¿Qué plataformas son líderes en ventas?
     1. ¿Cuáles crecen y cuáles se reducen?
     2. Elige varias plataformas potencialmente rentables.
  5. Crea un diagrama de caja para las ventas globales de todos los juegos, desglosados por plataforma.
     1. ¿Son significativas las diferencias en las ventas?
     2. ¿Qué sucede con las ventas promedio en varias plataformas?
     3. Describe tus hallazgos.
  6. Mira cómo las reseñas de usuarios y profesionales afectan las ventas de una plataforma popular (tu elección).
     1. Crea un gráfico de dispersión
     2. calcula la correlación entre las reseñas y las ventas.
     3. Saca conclusiones.
  7. Teniendo en cuenta tus conclusiones compara las ventas de los mismos juegos en otras plataformas.
  8. Echa un vistazo a la distribución general de los juegos por género.
     1. ¿Qué se puede decir de los géneros más rentables?
     2. ¿Puedes generalizar acerca de los géneros con ventas altas y bajas?

4. Crea un perfil de usuario para cada región

* 1. Para cada región (NA, UE, JP) determina:
     1. Las cinco plataformas principales. Describe las variaciones en sus cuotas de mercado de una región a otra.
     2. Los cinco géneros principales. Explica la diferencia.
     3. Si las clasificaciones de ESRB afectan a las ventas en regiones individuales.

5. Prueba las siguientes hipótesis:

5.1 Calificaciones Promedio

5.1.1 Las calificaciones promedio de los usuarios para las plataformas Xbox One y PC son las mismas.

5.1.2 Las calificaciones promedio de los usuarios para los géneros de Acción y Deportes son diferentes.

5.2 Establece tu mismo el valor de umbral *alfa*  (cometario: también conocido como nivel de significación) . P VALUE

Explica:

5.2.1 Cómo formulaste las hipótesis nula y alternativa.

5.2.1 Qué criterio utilizaste para probar las hipótesis y por qué.

6. Escribe una conclusión general

Formato: Completa la tarea en Jupyter Notebook. Inserta el código de programación en las celdas *code* y las explicaciones de texto en las celdas *markdown*. Aplica formato y agrega encabezados.

### Descripción de datos

— *Name* (Nombre)

— *Platform* (Plataforma)

— *Year\_of\_Release* (Año de lanzamiento)

— *Genre* (Género)

— *NA\_sales* (ventas en Norteamérica en millones de dólares estadounidenses)

— *EU\_sales* (ventas en Europa en millones de dólares estadounidenses)

— *JP\_sales* (ventas en Japón en millones de dólares estadounidenses)

— *Other\_sales* (ventas en otros países en millones de dólares estadounidenses)

— *Critic\_Score* (máximo de 100)

— *User\_Score* (máximo de 10)

— *Rating* (ESRB)

Es posible que los datos de 2016 estén incompletos.

## ¿Cómo será evaluado mi proyecto?

Lee atentamente estos criterios de evaluación de proyectos antes de empezar a trabajar.

Esto es lo que buscan los revisores de proyecto cuando evalúan tu proyecto:

* ¿Cómo describirías los problemas identificados en los datos?
* ¿Cómo se prepara un dataset para el análisis?
* ¿Cómo creas gráficos de distribución y cómo los explicas?
* ¿Cómo calculas la desviación estándar y varianza?
* ¿Formulas las hipótesis alternativas y nulas?
* ¿Qué métodos aplicas a la hora de probarlos?
* ¿Explicas los resultados de tus pruebas de hipótesis?
* ¿Sigues la estructura del proyecto y mantienes tu código ordenado y comprensible?
* ¿A qué conclusiones llegas?
* ¿Has dejado comentarios claros y relevantes en cada paso?

Todo lo que necesitas para completar este proyecto se encuentra en las hojas informativas y los resúmenes de los capítulos anteriores.

¡Buena suerte!

### Las calificaciones promedio de los usuarios para las plataformas Xbox One y PC son las mismas.

* **Hipótesis Nula (𝐻0*H*0​):** Las calificaciones promedio de los usuarios para las plataformas Xbox One y PC son iguales.
* **Hipótesis Alternativa (𝐻1*H*1​):** Las calificaciones promedio de los usuarios para las plataformas Xbox One y PC son diferentes.
  + ​5.1.2 Las calificaciones promedio de los usuarios para los géneros de Acción y Deportes son diferentes.
* **Hipótesis Nula (𝐻0*H*0​):** Las calificaciones promedio de los usuarios para los géneros de Acción y Deportes son iguales.
* **Hipótesis Alternativa (𝐻1*H*1​):** Las calificaciones promedio de los usuarios para los géneros de Acción y Deportes son diferentes.

En ambos casos, la hipótesis nula asume que no hay diferencia significativa entre las calificaciones promedio, mientras que la hipótesis alternativa sugiere que sí hay una diferencia significativa. La diferencia radica en las poblaciones que estamos comparando en cada caso: plataformas de juego en el primer caso y géneros de juego en el segundo caso.