

# Launch X



## OnBoarding

Aquí inicia tu viaje



### Modulo0Katas.ipynb

File Edit Selection View Go Run Terminal Help • Modulo0Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

- ✓ EJERCICIOSPYTHON
  - Módulo 10 - Manejo de errores.md
  - Modulo0Katas.ipynb
  - Modulo1Katas.ipynb
  - Modulo2Katas.md
  - Módulo3Katas.ipynb
  - Módulo4Katas.ipynb
  - Módulo5Katas.ipynb
  - Módulo6Katas.ipynb
  - Módulo7Katas.ipynb
  - Módulo8Katas.ipynb
  - Módulo9Katas.ipynb

Modulo0Katas.ipynb

Modulo0Katas.ipynb > Ejercicio: crea y ejecuta tu notebook > %pip install ipywidgets

+ Code + Markdown | ▶ Run All | Clear Outputs of All Cells | Restart | Python 3.9.10 32-bit

## Ejercicio: crea y ejecuta tu notebook

Primero instalamos la biblioteca:

```
%pip install ipywidgets
```

[1] ✓ 10.4s Python

... Output exceeds the size limit. Open the full output data in a text editor

Requirement already satisfied: ipywidgets in c:\users\alejandro\appdata\local\programs\python\python39-32\lib\site-packages (7.6.5)

Requirement already satisfied: ipython>=4.0.0 in c:\users\alejandro\appdata\roaming\python\python39\site-packages (from ipywidgets) (8.0.1)

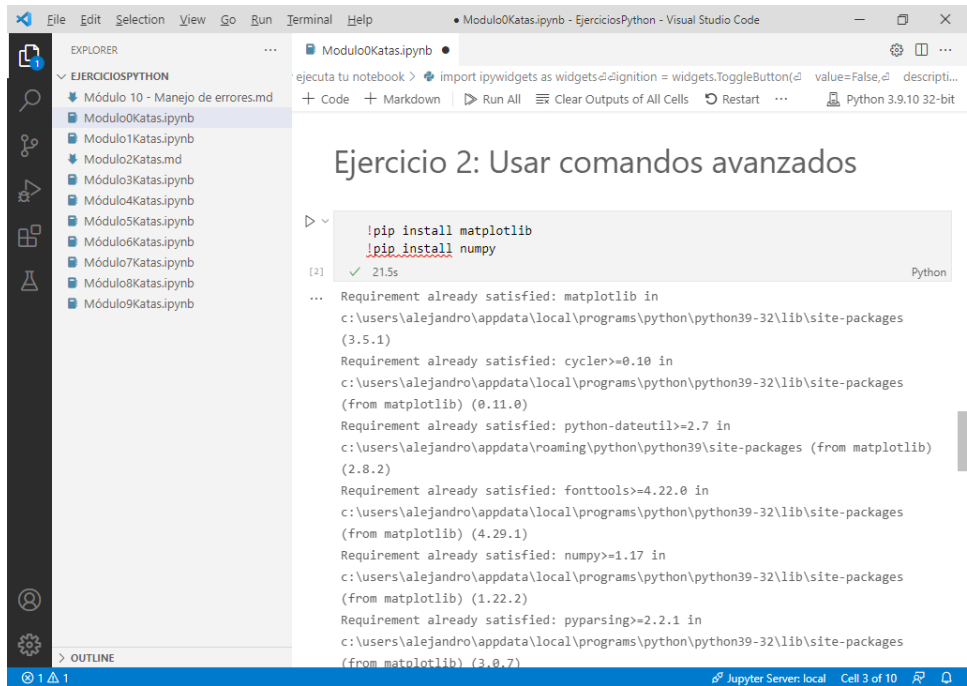
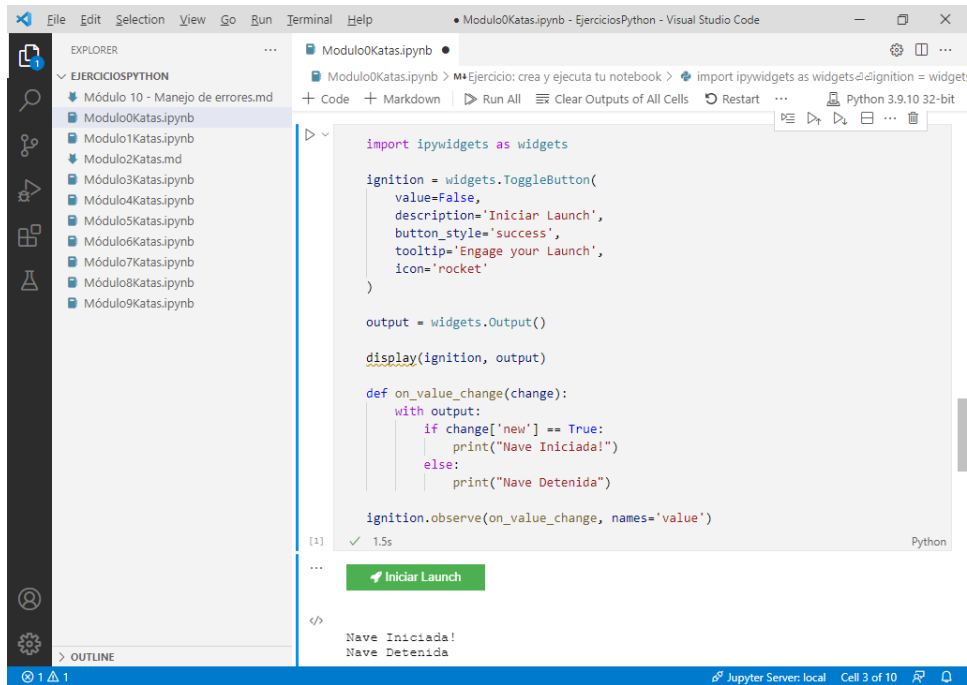
Requirement already satisfied: ipykernel>=4.5.1 in c:\users\alejandro\appdata\roaming\python\python39\site-packages (from ipywidgets) (6.9.0)

Requirement already satisfied: jupyterlab-widgets>=1.0.0 in c:\users\alejandro\appdata\local\programs\python\python39-32\lib\site-packages (from ipywidgets) (1.0.2)

Requirement already satisfied: ipython-genutils<=0.2.0 in c:\users\alejandro\appdata\local\programs\python\python39-32\lib\site-packages (from ipywidgets) (0.2.0)

Requirement already satisfied: widgetsnbextension<=3.5.0 in

1 1 Jupyter Server: local Cell 2 of 10



FileEditSelectionViewGoRunTerminalHelp

Modulo0Katas.ipynb - EjerciciosPython - Visual Studio Code

—□×

EXPLORER

...

EJERCICIOSPYTHON

▼ Módulo 10 - Manejo de errores.md

Modulo0Katas.ipynb

Modulo1Katas.ipynb

▼ Módulo2Katas.md

Modulo3Katas.ipynb

Modulo4Katas.ipynb

Modulo5Katas.ipynb

Modulo6Katas.ipynb

Modulo7Katas.ipynb

Modulo8Katas.ipynb

Modulo9Katas.ipynb

Modulo0Katas.ipynb

•

...

ejecuta tu notebook > import ipywidgets as widgets; widget.ToggleButton(value=False, descri...

+ Code + Markdown ▶ Run All ≡ Clear Outputs of All Cells ↺ Restart ... 🐍 Python 3.9.10 32-bit

## Niveles de Oxígeno

Muestra diez minutos de niveles de oxígeno en tu nave.

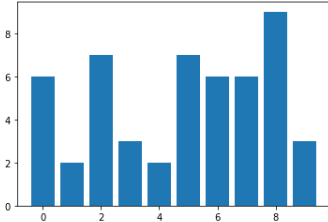
▶ ▼

```
import numpy as np
import matplotlib.pyplot as plt
data = np.random.default_rng(12345)
oxy_nums = data.integers(low=0, high=10, size=10)

plt.bar(range(len(oxy_nums)), oxy_nums)
plt.show()
```

[\*] ✓ 17.1s Python

...



Index	Oxygen Level
0	6
1	2
2	7
3	3
4	2
5	7
6	6
7	6
8	9
9	3

> OUTLINE

1 1

Jupyter Server: local Cell 3 of 10

File Edit Selection View Go Run Terminal Help • Modulo0Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

EJERCICIOSPYTHON

- Módulo 10 - Manejo de errores.md
- Modulo0Katas.ipynb
- Modulo1Katas.ipynb
- Modulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Módulo8Katas.ipynb
- Módulo9Katas.ipynb

Modulo0Katas.ipynb

ejecuta tu notebook > import ipywidgets as widgets; ignition = widgets.ToggleButton(value=False, description=...)

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

## Velocidad de la nave

Muestra los segundos necesarios para pasar de 0 a 11200 metros por segundo, dada la aceleración de la nave en metros por segundo.

```
endVelocity = 11200
startVelocity = 0
acceleration = 9.8

time = (endVelocity - startVelocity) / acceleration
print("Tiempo para alcanzar la velocidad deseada = ", time)
```

[2] ✓ 0.2s Python

... Tiempo para alcanzar la velocidad deseada = 1142.8571428571427

Curso Propedéutico de Python para Launch X - Innovación Virtual.

Material desarrollado con base en los contenidos de MSLearn y la metáfora de LaunchX, traducción e implementación por: Fernanda Ochoa - Learning Producer de LaunchX.

Redes:

- GitHub: [FernandaOchoa](#)
- Twitter: [@imonsh](#)
- Instagram: [fherz8a](#)

> OUTLINE

1 Jupyter Server: local Cell 3 of 10

## Modulo1Katas.ipynb

File Edit Selection View Go Run Terminal Help • Modulo1Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

EJERCICIOSPYTHON

- Módulo 10 - Manejo de errores.md
- Modulo0Katas.ipynb
- Modulo1Katas.ipynb
- Modulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Módulo8Katas.ipynb
- Módulo9Katas.ipynb

Modulo1Katas.ipynb

Modulo1Katas.ipynb > Tu primer programa > from datetime import date; print("Today's date is: "+ str(date.today()))

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

Oficial superior> "¿Cadete?"

Usted > "¿Sí, señora?"

Oficial superior> "¿Puedes construir un programa que me muestre la fecha? La computadora del barco no venía con mucho en términos de programas de utilidad, por lo que necesitamos crear los que necesitamos. ¿Puedo confiar en ti con esto?"

Usted > "Considéralo hecho".

Oficial superior> "Excelente".

## Tu primer programa

Para crear este programa, deberás utilizar los conceptos que aprendiste en el último módulo. Usarás Jupyter Notebook en este ejercicio, que es una combinación de texto y código con la que puede interactuar. Finaliza el código siguiente para que el resultado muestre la fecha de hoy.

```
from datetime import date

print("Today's date is: "+ str(date.today()))
```

[1] ✓ 0.3s Python

... Today's date is: 2022-02-16

> OUTLINE

0 Jupyter Server: local Cell 2 of 9

File Edit Selection View Go Run Terminal Help • Modulo1Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

▼ EJERCICIOSPYTHON

- ▼ Módulo 10 - Manejo de errores.md
- Modulo0Katas.ipynb
- Modulo1Katas.ipynb
- ▼ Modulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Modulo8Katas.ipynb
- Módulo9Katas.ipynb

Modulo1Katas.ipynb •

Modulo1Katas.ipynb > M4Soluciones

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

## Construir un convertidor de unidades

El oficial superior> "Necesito una cosa más: un programa de conversión entre parsecs y años luz. Tener un programa de este tipo podría ser realmente útil en el puente para trazar nuestro curso".

Tú> "¡Lo haré!"

```
parsec = 11

lightyears = 3.26156 * parsec

print(str(parsec) + " parsec, is " + str(lightyears) + " lightyears")
```

[2] ✓ 0.1s Python

... 11 parsec, is 35.87715999999996 lightyears

[!TIP] 1 parsec es 3.26156 años luz. Utiliza el operador de multiplicación.

## Soluciones

Aquí está la solución para el primer programa:

Python 3.9.10 32-bit Jupyter Server: local Cell 6 of 9

## Modulo2Katas.md

File Edit Selection View Go Run Terminal Help Modulo2Katas.md - EjerciciosPython - Visual Studio Code

EXPLORER

▼ EJERCICIOSPYTHON

- ▼ Módulo 10 - Manejo de errores.md
- Modulo0Katas.ipynb
- Modulo1Katas.ipynb
- ▼ Modulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Modulo8Katas.ipynb
- Módulo9Katas.ipynb

Modulo1Katas.ipynb • Modulo2Katas.md X

Modulo2Katas.md > Ejercicio - Crear un paquete > ## Crear un entorno virtual

### # Ejercicio - Crear un paquete

En este ejercicio, aprenderás a utilizar entornos virtuales como una forma para no afectar a los paquetes instalados globalmente u otros programas que se ejecutan en tu máquina.

\*Para este ejercicio es necesario que lo ejecutes desde la terminal, línea de comandos, cmd, consola, cli, etc. de tu computadora, sé que es desafiante, pero no te preocupes ¡¡Sé que puedes lograrlo!!\*

### ## Crear un entorno virtual

Crear un entorno virtual mediante ``venv``

\* Ejecutar en su terminal: ``python3 -m venv env`` o bien ``python -m venv env``

```
python3 -m venv env
```

python -m venv env

Ahora tienes un directorio (folder) ``env`` creado en tu terminal.

\* Ejecuta el comando para activar el entorno virtual: ``source env/bin/activate``

```
source env/bin/activate
```

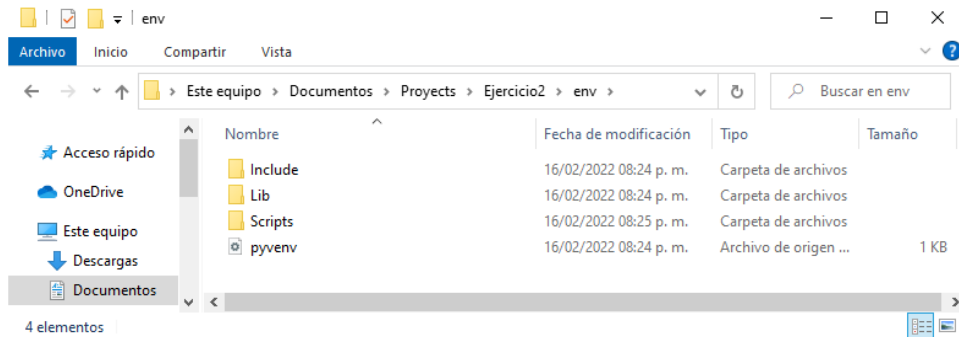
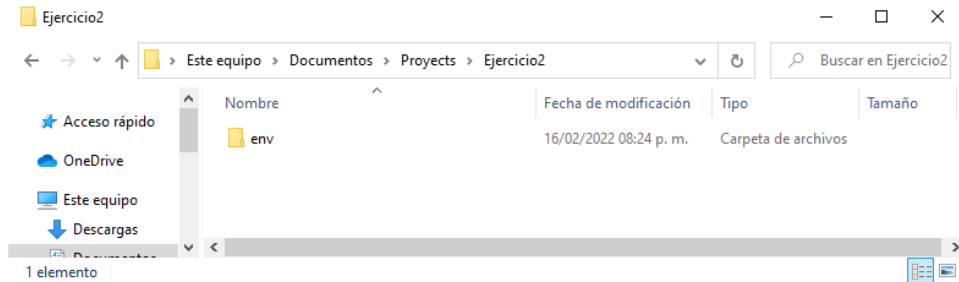
# Windows

```
env\bin\activate
```

Python 3.9.10 32-bit Ln 13, Col 8 Spaces: 4 UTF-8 LF Markdown

```
C:\Windows\system32\cmd.exe

C:\Users\Alejandro\Documents\Proyectos>cd C:\Users\Alejandro\Documents\Proyectos\Ejercicio2
C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>python -m venv env
C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>
```



```
C:\Windows\system32\cmd.exe

C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>C:\Users\Alejandro\Documents\Proyectos\Ejercicio2\env\Scripts\activate.bat
(env) C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>
```

```
C:\Windows\system32\cmd.exe

(env) C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>pip freeze
(env) C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>pip install python-dateutil
Collecting python-dateutil
  Using cached python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Collecting six>=1.5
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, python-dateutil
Successfully installed python-dateutil-2.8.2 six-1.16.0
WARNING: You are using pip version 21.2.4; however, version 22.0.3 is available.
You should consider upgrading via the 'C:\Users\Alejandro\Documents\Proyectos\Ejercicio2\env\Scripts\python.exe -m pip install
--upgrade pip' command.

(env) C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>pip freeze
python-dateutil==2.8.2
six==1.16.0

(env) C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>deactivate
C:\Users\Alejandro\Documents\Proyectos\Ejercicio2>
```

## Módulo3Katas.ipynb

File Edit Selection View Go Run Terminal Help • Módulo3Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

✓ EJERCICIOSPYTHON

- Módulo 10 - Manejo de errores.md
- Módulo0Katas.ipynb
- Módulo1Katas.ipynb
- Módulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Módulo8Katas.ipynb
- Módulo9Katas.ipynb

Módulo3Katas.ipynb

aciones if, else, y elif > # Añadir el código necesario para crear una variable que guarde la velocidad del asteroide.ías...

+ Code + Markdown Run All Clear Outputs of All Cells Restart ... Python 3.9.10 32-bit

### Ejercicio - Escribir declaraciones `if`, `else`, y `elif`

Las instrucciones `if` te permiten ejecutar condicionalmente código Python. Se usan comúnmente en Python para "tomar decisiones" sobre lo que debería suceder a continuación mientras se ejecuta un programa.

Para crear una instrucción `if` en Python, define una expresión de prueba que pueda tener un valor `True` o `False`, seguido de un bloque de código con sangría que se ejecutará si se cumple la condición.

```
if expresion_prueba:
    # intrucción(es) a ejecutar
```

Para escribir un programa con una lógica condicional más compleja, puedes agregar instrucciones `else` y `elif` al bloque de código. También puedes anidar instrucciones condicionales.

0 0 0 Jupyter Server: local Cell 2 of 14

File Edit Selection View Go Run Terminal Help • Módulo3Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

✓ EJERCICIOSPYTHON

- Módulo 10 - Manejo de errores.md
- Módulo0Katas.ipynb
- Módulo1Katas.ipynb
- Módulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Módulo8Katas.ipynb
- Módulo9Katas.ipynb

Módulo3Katas.ipynb

Módulo3Katas.ipynb > Ejercicio - Escribir declaraciones if, else, y elif

+ Code + Markdown Run All Clear Outputs of All Cells Restart ... Python 3.9.10 32-bit

Para este ejercicio, escribirás una lógica condicional que imprima una advertencia si un asteroide se acerca a la Tierra demasiado rápido. La velocidad del asteroide varía dependiendo de lo cerca que esté del sol, y cualquier velocidad superior a 25 kilómetros por segundo (km/s) merece una advertencia.

Un asteroide se acerca, y viaja a una velocidad de 49 km/s.

```
# Añadir el código necesario para crear una variable que guarde la velocidad del
asteroide = 49
# Escribe una expresión de prueba para calcular si necesita una advertencia.
if asteroide > 25:
    # Agregue las instrucciones que se ejecutarán si la expresión de prueba es true o
    print('¡Advertencia! ¡Un asteroide se acerca a la Tierra demasiado rápido! ¡S
else:
    print('¡Estuvo cerca!')
```

[3] ✓ 0.1s Python

... ¡Advertencia! ¡Un asteroide se acerca a la Tierra demasiado rápido! ¡Salvese quien pueda!

0 0 0 Jupyter Server: local Cell 1 of 14

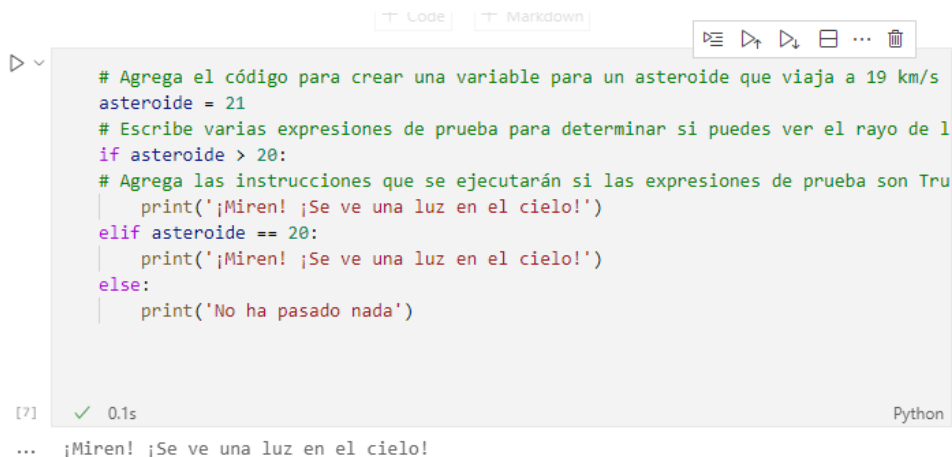
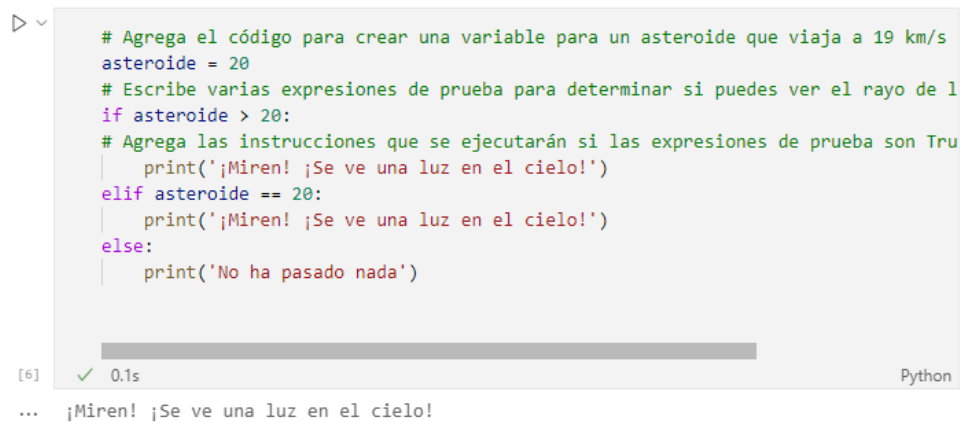
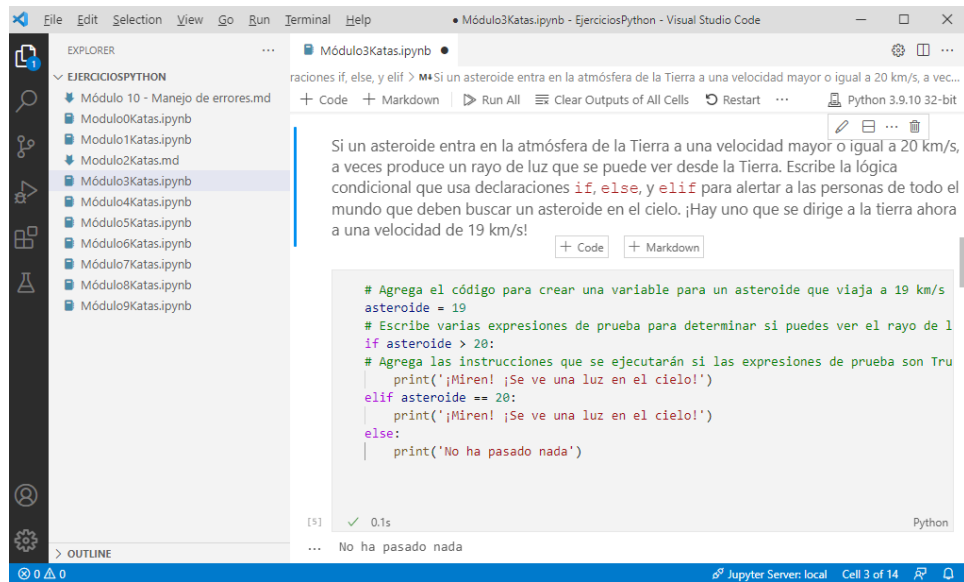
+ Code + Markdown

Python

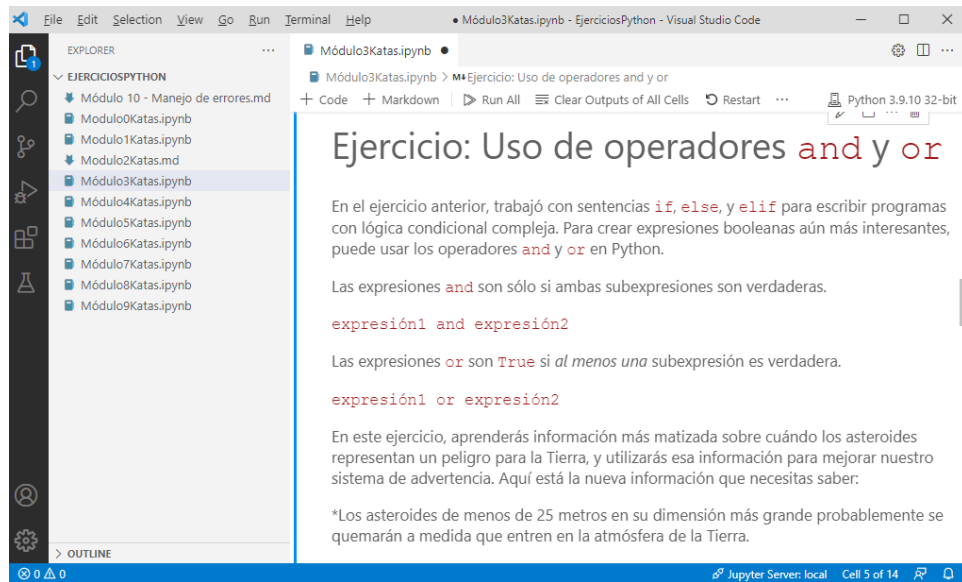
```
# Añadir el código necesario para crear una variable que guarde la velocidad del
asteroide = 25
# Escribe una expresión de prueba para calcular si necesita una advertencia.
if asteroide > 25:
    # Agregue las instrucciones que se ejecutarán si la expresión de prueba es true o
    print('¡Advertencia! ¡Un asteroide se acerca a la Tierra demasiado rápido! ¡S
else:
    print('¡Estuvo cerca!')
```

[2] ✓ 0.3s Python

... ¡Estuvo cerca!







Visual Studio Code interface showing a Jupyter Notebook titled "Ejercicio: Uso de operadores and y or". The Explorer sidebar on the left lists files under "EJERCICIOSPYTHON", with "Módulo3Katas.ipynb" selected. The notebook editor displays the following content:

## Ejercicio: Uso de operadores and y or

En el ejercicio anterior, trabajó con sentencias `if`, `else`, y `elif` para escribir programas con lógica condicional compleja. Para crear expresiones booleanas aún más interesantes, puede usar los operadores `and` y `or` en Python.

Las expresiones `and` son sólo si ambas subexpresiones son verdaderas.

```
expresión1 and expresión2
```

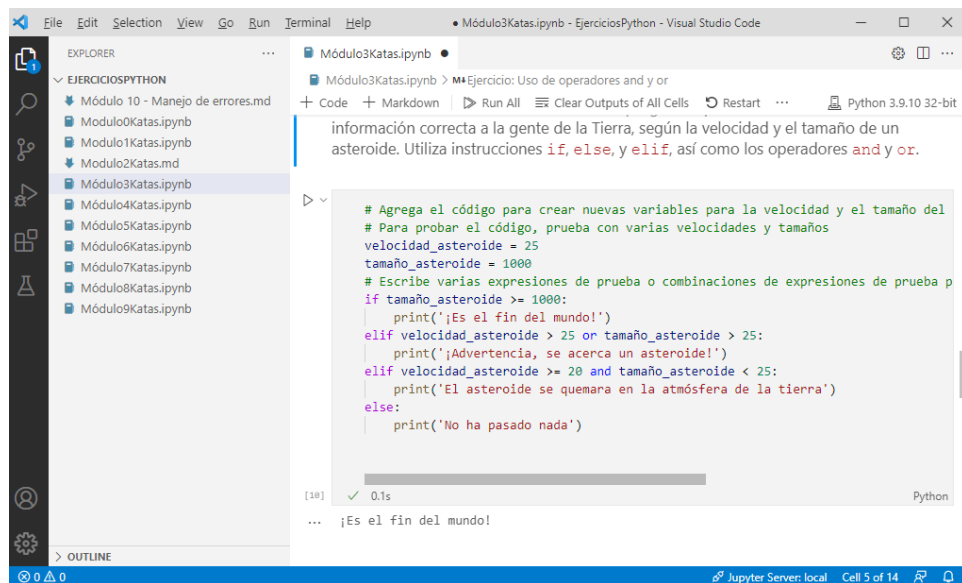
Las expresiones `or` son `True` si *al menos una* subexpresión es verdadera.

```
expresión1 or expresión2
```

En este ejercicio, aprenderás información más matizada sobre cuándo los asteroides representan un peligro para la Tierra, y utilizarás esa información para mejorar nuestro sistema de advertencia. Aquí está la nueva información que necesitas saber:

\*Los asteroides de menos de 25 metros en su dimensión más grande probablemente se quemarán a medida que entren en la atmósfera de la Tierra.

At the bottom of the window, the status bar indicates "Jupyter Server: local" and "Cell 5 of 14".



Visual Studio Code interface showing the same Jupyter Notebook, but now with a code cell selected and its output displayed. The Explorer sidebar remains the same. The code cell contains the following Python code:

```
# Agrega el código para crear nuevas variables para la velocidad y el tamaño del
# Para probar el código, prueba con varias velocidades y tamaños
velocidad_asteroide = 25
tamaño_asteroide = 1000
# Escribe varias expresiones de prueba o combinaciones de expresiones de prueba p
if tamaño_asteroide >= 1000:
    print('¡Es el fin del mundo!')
elif velocidad_asteroide > 25 or tamaño_asteroide > 25:
    print('¡Advertencia, se acerca un asteroide!')
elif velocidad_asteroide >= 20 and tamaño_asteroide < 25:
    print('El asteroide se quemará en la atmósfera de la tierra')
else:
    print('No ha pasado nada')
```

The output of the cell is:

```
[10] ✓ 0.1s Python
... ¡Es el fin del mundo!
```

The status bar at the bottom shows "Jupyter Server: local" and "Cell 5 of 14".

```
# Agrega el código para crear nuevas variables para la velocidad y el tamaño del
# Para probar el código, prueba con varias velocidades y tamaños
velocidad_asteroide = 25
tamaño_asteroide = 40
# Escribe varias expresiones de prueba o combinaciones de expresiones de prueba p
if tamaño_asteroide >= 1000:
    print('¡Es el fin del mundo!')
elif velocidad_asteroide > 25 or tamaño_asteroide > 25:
    print('¡Advertencia, se acerca un asteroide!')
elif velocidad_asteroide >= 20 and tamaño_asteroide < 25:
    print('El asteroide se quemara en la atmósfera de la tierra')
else:
    print('No ha pasado nada')
```

[11] ✓ 0.1s Python

... ¡Advertencia, se acerca un asteroide!

```
# Agrega el código para crear nuevas variables para la velocidad y el tamaño del
# Para probar el código, prueba con varias velocidades y tamaños
velocidad_asteroide = 25
tamaño_asteroide = 24
# Escribe varias expresiones de prueba o combinaciones de expresiones de prueba p
if tamaño_asteroide >= 1000:
    print('¡Es el fin del mundo!')
elif velocidad_asteroide > 25 or tamaño_asteroide > 25:
    print('¡Advertencia, se acerca un asteroide!')
elif velocidad_asteroide >= 20 and tamaño_asteroide < 25:
    print('El asteroide se quemara en la atmósfera de la tierra')
else:
    print('No ha pasado nada')
```

[13] ✓ 0.1s Python

... El asteroide se quemara en la atmósfera de la tierra

```
# Agrega el código para crear nuevas variables para la velocidad y el tamaño del
# Para probar el código, prueba con varias velocidades y tamaños
velocidad_asteroide = 19
tamaño_asteroide = 25
# Escribe varias expresiones de prueba o combinaciones de expresiones de prueba p
if tamaño_asteroide >= 1000:
    print('¡Es el fin del mundo!')
elif velocidad_asteroide > 25 or tamaño_asteroide > 25:
    print('¡Advertencia, se acerca un asteroide!')
elif velocidad_asteroide >= 20 and tamaño_asteroide < 25:
    print('El asteroide se quemara en la atmósfera de la tierra')
else:
    print('No ha pasado nada')
```

[14] ✓ 0.1s Python

... No ha pasado nada

## Módulo4Katas.ipynb

File Edit Selection View Go Run Terminal Help Módulo4Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

- EJERCICIOSPYTHON
  - Módulo 10 - Manejo de errores.md
  - Módulo0Katas.ipynb
  - Módulo1Katas.ipynb
  - Módulo2Katas.md
  - Módulo3Katas.ipynb
  - Módulo4Katas.ipynb**
  - Módulo5Katas.ipynb
  - Módulo6Katas.ipynb
  - Módulo7Katas.ipynb
  - Módulo8Katas.ipynb
  - Módulo9Katas.ipynb

Módulo4Katas.ipynb X

Módulo4Katas.ipynb > Ejercicio 1: Transformar cadenas

+ Code + Markdown Run All Clear Outputs of All Cells Outline Python 3.9.10 32-bit

### Ejercicio 1: Transformar cadenas

Hay varias operaciones que puedes realizar en las cadenas cuando las manipulamos. En este ejercicio, usarás métodos de cadena para modificar el texto con hechos sobre la Luna y luego extraerás información para crear un breve resumen.

**Nota** Dedica unos minutos a tratar de encontrar una solución. Luego desplázate hacia abajo hasta la parte inferior para ver si has logrado compilar el programa de acuerdo con las especificaciones.

El texto con el que trabajarás es el siguiente:

```
text = """Interesting facts about the Moon. The Moon is Earth's only satellite. T
On average, the Moon moves 4cm away from the Earth every year. This yearly drift
```

[ 1 ] Python

Primero, divide el texto en cada oración para trabajar con su contenido:

Jupyter Server: local Cell 1 of 42

File Edit Selection View Go Run Terminal Help Módulo4Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

- EJERCICIOSPYTHON
  - Módulo 10 - Manejo de errores.md
  - Módulo0Katas.ipynb
  - Módulo1Katas.ipynb
  - Módulo2Katas.md
  - Módulo3Katas.ipynb
  - Módulo4Katas.ipynb**
  - Módulo5Katas.ipynb
  - Módulo6Katas.ipynb
  - Módulo7Katas.ipynb
  - Módulo8Katas.ipynb
  - Módulo9Katas.ipynb

Módulo4Katas.ipynb •

Módulo4Katas.ipynb > Ejercicio 1: Transformar cadenas > # Añade el código necesario

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

```
text = """Interesting facts about the Moon. The Moon is Earth's only satellite. T
On average, the Moon moves 4cm away from the Earth every year. This yearly drift
```

[ 1 ] ✓ 0.3s Python

Primero, divide el texto en cada oración para trabajar con su contenido:

```
# Añade el código necesario
text.split('.')
```

[ 6 ] ✓ 0.1s Python

```
... ['Interesting facts about the Moon',
    ' The Moon is Earth's only satellite',
    ' There are several interesting facts about the Moon and how it affects life here
on Earth',
    ' \nOn average, the Moon moves 4cm away from the Earth every year',
    ' This yearly drift is not significant enough to cause immediate effects on
Earth',
    ' The highest daylight temperature of the Moon is 127 C',
    '']
```

Jupyter Server: local Cell 5 of 42

Transformar cadenas > # Ciclo for para recorrer la cadena: # Ciclo for para recorrer la cadena: for sentence in text\_parts...

Ahora, define algunas palabras clave para búsqueda que te ayudarán a determinar si una oración contiene un hecho.

```
# Define las palabras pista: average, temperature y distance suenan bien
text_parts = text.split(' ')
key_words = ["average", "temperature", "distance"]
```

[7] ✓ 0.1s Python

Cre un bucle para imprimir solo datos sobre la Luna que estén relacionados con las palabras clave definidas anteriormente:

```
# Ciclo for para recorrer la cadena
# Ciclo for para recorrer la cadena
for sentence in text_parts:
    for key_word in key_words:
        if key_word in sentence:
            print(sentence)
            break
```

[8] ✓ 0.1s Python

Jupyter Server: local Cell 9 of 42

Transformar cadenas > text = ""Interesting facts about the Moon. The Moon is Earth's only satellite. There are several ...

```
# Define las palabras pista: average, temperature y distance suenan bien
text_parts = text.split(' ')
key_words = ["average", "temperature", "distance"]
```

[7] ✓ 0.1s Python

Cre un bucle para imprimir solo datos sobre la Luna que estén relacionados con las palabras clave definidas anteriormente:

```
# Ciclo for para recorrer la cadena
# Ciclo for para recorrer la cadena
for sentence in text_parts:
    for key_word in key_words:
        if key_word in sentence:
            print(sentence)
            break
```

[8] ✓ 0.1s Python

...

On average, the Moon moves 4cm away from the Earth every year  
The highest daylight temperature of the Moon is 127 C.

Jupyter Server: local Cell 3 of 42

```
# Ciclo para cambiar C a Celsius
for sentence in text_parts:
    for key_word in key_words:
        if key_word in sentence:
            print(sentence.replace(' C ', ' Celsius'))
            break
```

[9] ✓ 0.1s Python

...

On average, the Moon moves 4cm away from the Earth every year  
The highest daylight temperature of the Moon is 127 Celsius.

File Edit Selection View Go Run Terminal Help • Módulo4Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

▼ EJERCICIOSPYTHON

- Módulo 10 - Manejo de errores.md
- Módulo0Katas.ipynb
- Módulo1Katas.ipynb
- Módulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Módulo8Katas.ipynb
- Módulo9Katas.ipynb

Módulo4Katas.ipynb

Transformar cadenas > # Ciclo para cambiar C a Celsius# for sentence in text\_parts:# for key\_word in key\_words: ...

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

## Ejercicio 2: Formateando Cadenas

Saber cómo dar formato a las cadenas es esencial cuando se presenta información de un programa. Hay algunas maneras diferentes de lograr esto en Python. En este ejercicio, se utilizan variables que contienen datos clave sobre la gravedad en varias lunas y luego se utilizan para dar formato e imprimir la información.

El formato tiene que acomodar información sobre otras lunas, por lo que debe ser genérico.

En lugar de reemplazar las variables en una cadena larga como parte de un párrafo, utiliza la información para presentarla en un formato tabular. El resultado debería verse así:

```
Gravity Facts about Ganymede
-----
Planet Name: Mars
Gravity on Ganymede: 1.4300000000000002 m/s2
```

# Datos con los que vas a trabajar

Jupyter Server: local Cell 11 of 42

File Edit Selection View Go Run Terminal Help • Módulo4Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

▼ EJERCICIOSPYTHON

- Módulo 10 - Manejo de errores.md
- Módulo0Katas.ipynb
- Módulo1Katas.ipynb
- Módulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Módulo8Katas.ipynb
- Módulo9Katas.ipynb

Módulo4Katas.ipynb

io 2: Formateando Cadenas > M+Primero, crea un título para el texto. Debido a que este texto trata sobre la gravedad en la Tierra y la Luna, úsalo para crear un título significativo. Utiliza las variables en lugar de escribir.

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

```
# Datos con los que vas a trabajar
name = "Moon"
gravity = 0.00162 # in kms
planet = "Earth"
```

[10] ✓ 0.1s Python

Primero, crea un título para el texto. Debido a que este texto trata sobre la gravedad en la Tierra y la Luna, úsalo para crear un título significativo. Utiliza las variables en lugar de escribir.

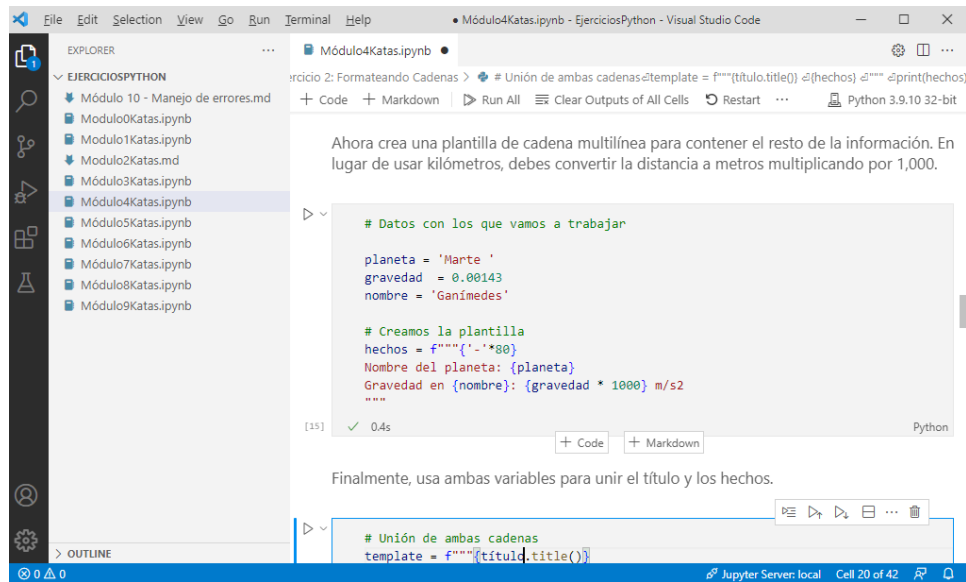
```
# Creamos el título
titulo = 'Gravity in ' + 'the ' + planet + ' and the ' + name
titulo.title()
```

[11] ✓ 0.1s Python

... 'Gravity In The Earth And The Moon'

Ahora crea una plantilla de cadena multilínea para contener el resto de la información. En lugar de usar kilómetros, debes convertir la distancia a metros multiplicando por 1,000.

Jupyter Server: local Cell 15 of 42



File Edit Selection View Go Run Terminal Help • Módulo4Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

- ▼ EJERCICIOSPYTHON
  - Módulo 10 - Manejo de errores.md
  - Módulo0Katas.ipynb
  - Módulo1Katas.ipynb
  - Módulo2Katas.md
  - Módulo3Katas.ipynb
  - Módulo4Katas.ipynb
  - Módulo5Katas.ipynb
  - Módulo6Katas.ipynb
  - Módulo7Katas.ipynb
  - Módulo8Katas.ipynb
  - Módulo9Katas.ipynb

Ejercicio 2: Formateando Cadenas > # Unión de ambas cadenas

```
template = f"{{titulo.title()}} {{hechos}} {{print(hechos)}}
```

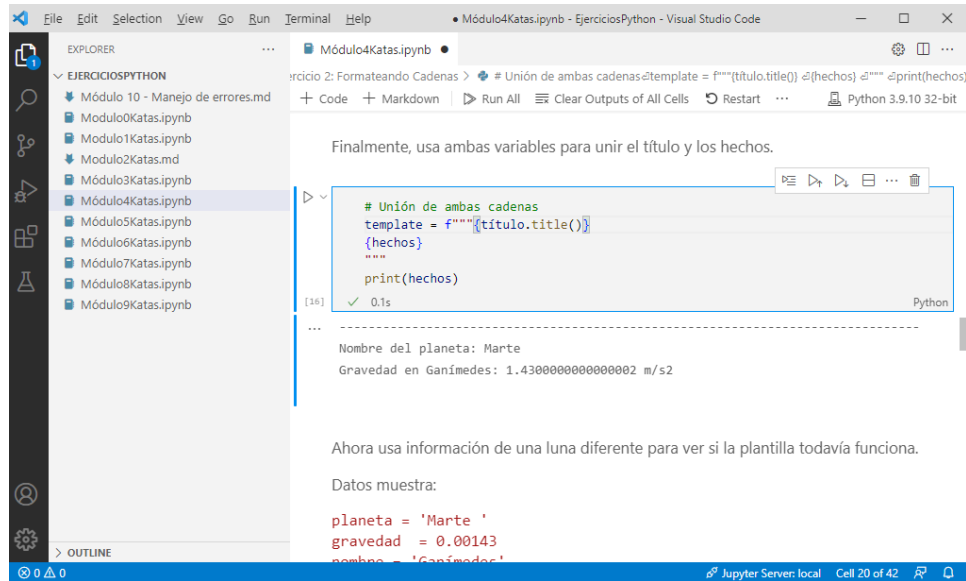
Ahora crea una plantilla de cadena multilinea para contener el resto de la información. En lugar de usar kilómetros, debes convertir la distancia a metros multiplicando por 1,000.

```
# Datos con los que vamos a trabajar
planeta = 'Marte '
gravedad = 0.00143
nombre = 'Ganímedes'

# Creamos la plantilla
hechos = f"{'-'*80}"
Nombre del planeta: {planeta}
Gravedad en {nombre}: {gravedad * 1000} m/s2
```

Finalmente, usa ambas variables para unir el título y los hechos.

```
# Unión de ambas cadenas
template = f"{{titulo.title()}}
```



File Edit Selection View Go Run Terminal Help • Módulo4Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

- ▼ EJERCICIOSPYTHON
  - Módulo 10 - Manejo de errores.md
  - Módulo0Katas.ipynb
  - Módulo1Katas.ipynb
  - Módulo2Katas.md
  - Módulo3Katas.ipynb
  - Módulo4Katas.ipynb
  - Módulo5Katas.ipynb
  - Módulo6Katas.ipynb
  - Módulo7Katas.ipynb
  - Módulo8Katas.ipynb
  - Módulo9Katas.ipynb

Ejercicio 2: Formateando Cadenas > # Unión de ambas cadenas

```
template = f"{{titulo.title()}} {{hechos}} {{print(hechos)}}
```

Finalmente, usa ambas variables para unir el título y los hechos.

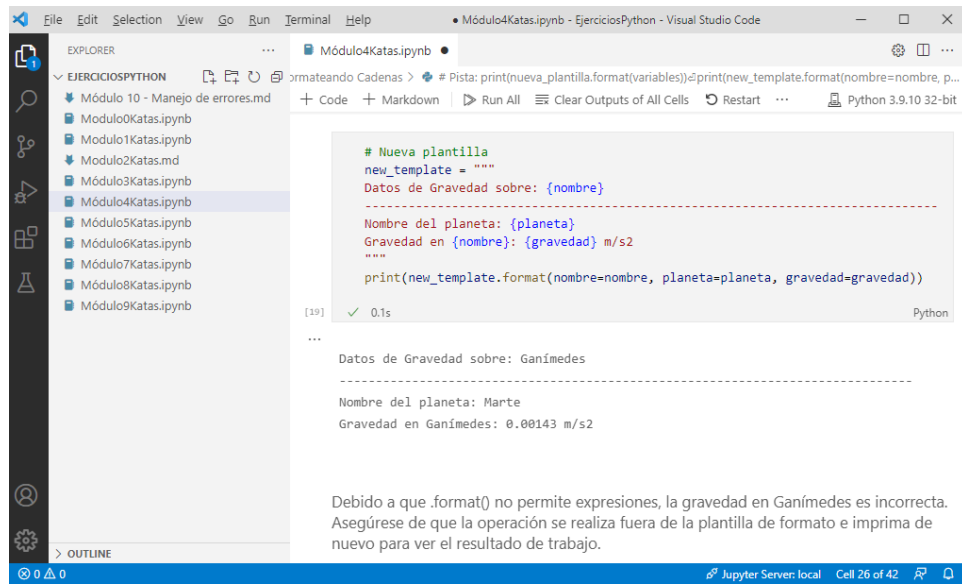
```
# Unión de ambas cadenas
template = f"{{titulo.title()}}
{hechos}
"
print(hechos)
```

Nombre del planeta: Marte  
Gravedad en Ganímedes: 1.4300000000000002 m/s2

Ahora usa información de una luna diferente para ver si la plantilla todavía funciona.

Datos muestra:

```
planeta = 'Marte '
gravedad = 0.00143
nombre = 'Ganímedes'
```



EXPLORER

• Módulo4Katas.ipynb

mateando Cadenas > # Pista: print(nueva\_plantilla.format(variables)); print(new\_template.format(nombre=nombre, p...

```
# Nueva plantilla
new_template = """
Datos de Gravedad sobre: {nombre}

-----

Nombre del planeta: {planeta}
Gravedad en {nombre}: {gravedad} m/s2
"""

print(new_template.format(nombre=nombre, planeta=planeta, gravedad=gravedad))
```

[19] ✓ 0.1s Python

...

Datos de Gravedad sobre: Ganímedes

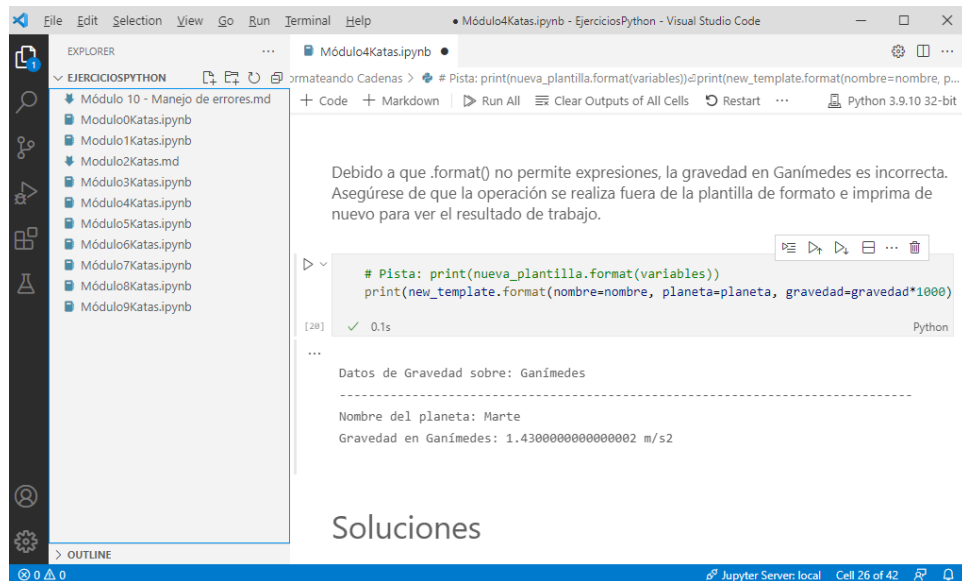
-----

Nombre del planeta: Marte

Gravedad en Ganímedes: 0.00143 m/s2

Debido a que .format() no permite expresiones, la gravedad en Ganímedes es incorrecta. Asegúrese de que la operación se realiza fuera de la plantilla de formato e imprima de nuevo para ver el resultado de trabajo.

Jupyter Server: local Cell 26 of 42



EXPLORER

• Módulo4Katas.ipynb

mateando Cadenas > # Pista: print(nueva\_plantilla.format(variables)); print(new\_template.format(nombre=nombre, p...

Debido a que .format() no permite expresiones, la gravedad en Ganímedes es incorrecta. Asegúrese de que la operación se realiza fuera de la plantilla de formato e imprima de nuevo para ver el resultado de trabajo.

```
# Pista: print(nueva_plantilla.format(variables))
print(new_template.format(nombre=nombre, planeta=planeta, gravedad=gravedad*1000))
```

[20] ✓ 0.1s Python

...

Datos de Gravedad sobre: Ganímedes

-----

Nombre del planeta: Marte

Gravedad en Ganímedes: 1.4300000000000002 m/s2

## Soluciones

Jupyter Server: local Cell 26 of 42

## Módulo5Katas.ipynb

File Edit Selection View Go Run Terminal Help Módulo5Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

- ✓ EJERCICIOSPYTHON
  - Módulo 10 - Manejo de errores.md
  - Módulo0Katas.ipynb
  - Módulo1Katas.ipynb
  - Módulo2Katas.md
  - Módulo3Katas.ipynb
  - Módulo4Katas.ipynb
  - Módulo5Katas.ipynb
  - Módulo6Katas.ipynb
  - Módulo7Katas.ipynb
  - Módulo8Katas.ipynb
  - Módulo9Katas.ipynb

Módulo5Katas.ipynb

Ejercicio 1 - Utilizar operadores aritméticos

+ Code + Markdown Run All Clear Outputs of All Cells Outline Python 3.9.10 32-bit

### Ejercicio1 - Utilizar operadores aritméticos

#### Operadores aritméticos en Python

Python proporciona operadores aritméticos comunes para que puedas realizar operaciones matemáticas en tu código. Estos incluyen las cuatro operaciones principales de suma, resta, multiplicación y división.

Exploreemos cómo podemos crear un programa que pueda calcular la distancia entre dos planetas. Comenzaremos usando dos distancias de planetas: Tierra (149.597.870 km) y Júpiter (778.547.200 km).

**TIP** Dedicar unos minutos a tratar de encontrar una solución. Luego desplázate hacia abajo hasta la parte inferior para ver si has logrado compilar el programa de acuerdo con las especificaciones.

0 0 Jupyter Server: local Cell 1 of 23

File Edit Selection View Go Run Terminal Help Módulo5Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

- ✓ EJERCICIOSPYTHON
  - Módulo 10 - Manejo de errores.md
  - Módulo0Katas.ipynb
  - Módulo1Katas.ipynb
  - Módulo2Katas.md
  - Módulo3Katas.ipynb
  - Módulo4Katas.ipynb
  - Módulo5Katas.ipynb
  - Módulo6Katas.ipynb
  - Módulo7Katas.ipynb
  - Módulo8Katas.ipynb
  - Módulo9Katas.ipynb

Módulo5Katas.ipynb

Realizar la operación > # Calcular la distancia entre planetas

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

```
# Calcular la distancia entre planetas
DistanciaKM = Júpiter - Tierra
DistanciaMillas = DistanciaKM * 0.621
print(DistanciaKM)
print(DistanciaMillas)
```

[1] ✓ 0.2s Python

### Realizar la operación

Con los valores obtenidos, es el momento de añadir el código para realizar la operación. Restarás el primer planeta del segundo para determinar la distancia en kilómetros. A continuación, puedes convertir la distancia del kilómetro en millas multiplicándola por 0.621.

[6] ✓ 0.1s Python

```
628949330
390577533.93
```

0 0 Jupyter Server: local Cell 5 of 23



File Edit Selection View Go Run Terminal Help • Módulo5Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

▼ EJERCICIOSPYTHON

- Módulo 10 - Manejo de errores.md
- Módulo0Katas.ipynb
- Módulo1Katas.ipynb
- Módulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Módulo8Katas.ipynb
- Módulo9Katas.ipynb

Módulo5Katas.ipynb

Realizar la operación > # Calcular la distancia entre planetas: DistanciaKM = Júpiter - Tierra: DistanciaMillas = Distanc...

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

## Ejercicio 2: convierte cadenas en números y usa valores absolutos

### Crear una aplicación para trabajar con números y entrada de usuario

Con frecuencia, deberás convertir los valores de cadena en números para realizar correctamente diferentes operaciones o determinar el valor absoluto de un número.

Para crear nuestra aplicación, queremos leer la distancia del sol para dos planetas, y luego mostrar la distancia entre los planetas. Haremos esto usando `input` para leer los valores, `int` para convertir a entero y luego `abs` para convertir el resultado en su valor absoluto.

### Lee los valores

Usando `input`, agrega el código para leer la distancia del sol para cada planeta, considerando 2 planetas.

Jupyter Server: local Cell 5 of 23

File Edit Selection View Go Run Terminal Help • Módulo5Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

▼ EJERCICIOSPYTHON

- Módulo 10 - Manejo de errores.md
- Módulo0Katas.ipynb
- Módulo1Katas.ipynb
- Módulo2Katas.md
- Módulo3Katas.ipynb
- Módulo4Katas.ipynb
- Módulo5Katas.ipynb
- Módulo6Katas.ipynb
- Módulo7Katas.ipynb
- Módulo8Katas.ipynb
- Módulo9Katas.ipynb

Módulo5Katas.ipynb

Convertir a número > # Convierte las cadenas de ambos planetas a números enteros: PrimerPlaneta = int(PrimerPlan...

+ Code + Markdown Run All Clear Outputs of All Cells Restart Python 3.9.10 32-bit

```
# Almacenar las entradas del usuario
#Pista: variable = input("¿Cuál es tu nombre?")
PrimerPlaneta = input("Distancia primer planeta")
SegundoPlaneta = input("Distancia segundo planeta")
print(PrimerPlaneta)
print(SegundoPlaneta)
```

[8] ✓ 62s Python

... 1000  
1000

### Convertir a número

Debido a que `input` devuelve valores de cadena, necesitamos convertirlos en números. Para nuestro ejemplo, usaremos `int`

```
# Convierte las cadenas de ambos planetas a números enteros
PrimerPlaneta = int(PrimerPlaneta)
SegundoPlaneta = int(SegundoPlaneta)
```

[10] ✓ 0.3s Python

Jupyter Server: local Cell 11 of 23

Visual Studio Code interface with the Explorer sidebar on the left showing a folder named 'EJERCICIOSPYTHON' containing files 'Módulo0Katas.ipynb' through 'Módulo9Katas.ipynb'. The main editor displays 'Módulo5Katas.ipynb'. The notebook cell contains the following text:

Realizar el cálculo y convertir a valor absoluto

Con los valores almacenados como números, ahora puedes agregar el código para realizar el cálculo, restando el primer planeta del segundo. Debido a que el segundo planeta podría ser un número mayor, usarás `abs` para convertirlo a un valor absoluto. También agregarás el código para mostrar el resultado en millas multiplicando la distancia del kilómetro por 0.621

```
# Realizar el cálculo y determinar el valor absoluto
DistanciaKM = SegundoPlaneta - PrimerPlaneta
print(DistanciaKM)

# Convertir de KM a Millas
DistanciaMillas = DistanciaKM * 0.621
print(abs(DistanciaMillas))
```

The output of the cell shows:

```
... -15000
9315.0
```

The bottom status bar indicates 'Jupyter Server: local' and 'Cell 13 of 23'.

## Módulo6Katas.ipynb

Visual Studio Code interface with the Explorer sidebar on the left showing the same 'EJERCICIOSPYTHON' folder. The main editor displays 'Módulo6Katas.ipynb'. The notebook cell contains the following text:

ar nombres de planetas > M• En primer lugar, crea una variable denominada `planets`. Agrega los ocho planetas (sin Plutón) a la lista. A c...

### Ejercicio1: Crear y usar listas de Python

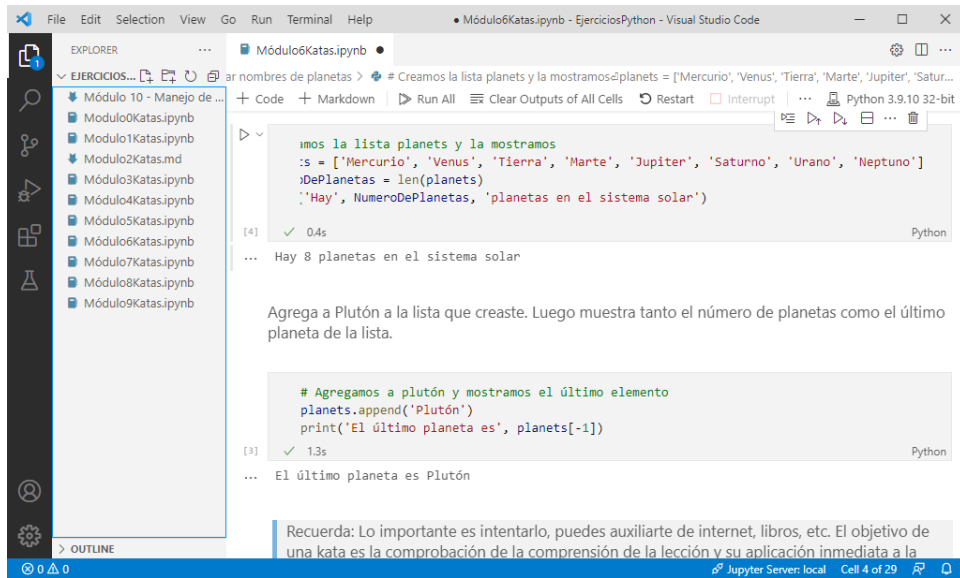
#### Ejercicio: Usar listas para almacenar nombres de planetas

Las listas permiten almacenar varios valores en una sola variable. Comenzarás un proyecto sobre información planetaria creando una lista de planetas.

**TIP** Dedicar unos minutos para tratar de encontrar una solución. Luego desplázate hacia abajo hasta la parte inferior para ver si has logrado compilar el programa de acuerdo con las especificaciones

En primer lugar, crea una variable denominada `planets`. Agrega los ocho planetas (sin Plutón) a la lista. A continuación, muestra el número de planetas.

The bottom status bar indicates 'Jupyter Server: local' and 'Cell 3 of 29'.



File Edit Selection View Go Run Terminal Help • Módulo6Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

ar nombres de planetas > • Creamos la lista planets y la mostramos: planets = ['Mercurio', 'Venus', 'Tierra', 'Marte', 'Jupiter', 'Saturno', 'Urano', 'Neptuno']

```
 planets = ['Mercurio', 'Venus', 'Tierra', 'Marte', 'Jupiter', 'Saturno', 'Urano', 'Neptuno']
DePlanetas = len(planets)
'Hay', NumeroDePlanetas, 'planetas en el sistema solar')
```

[4] ✓ 0.4s Python

... Hay 8 planetas en el sistema solar

Agrega a Plutón a la lista que creaste. Luego muestra tanto el número de planetas como el último planeta de la lista.

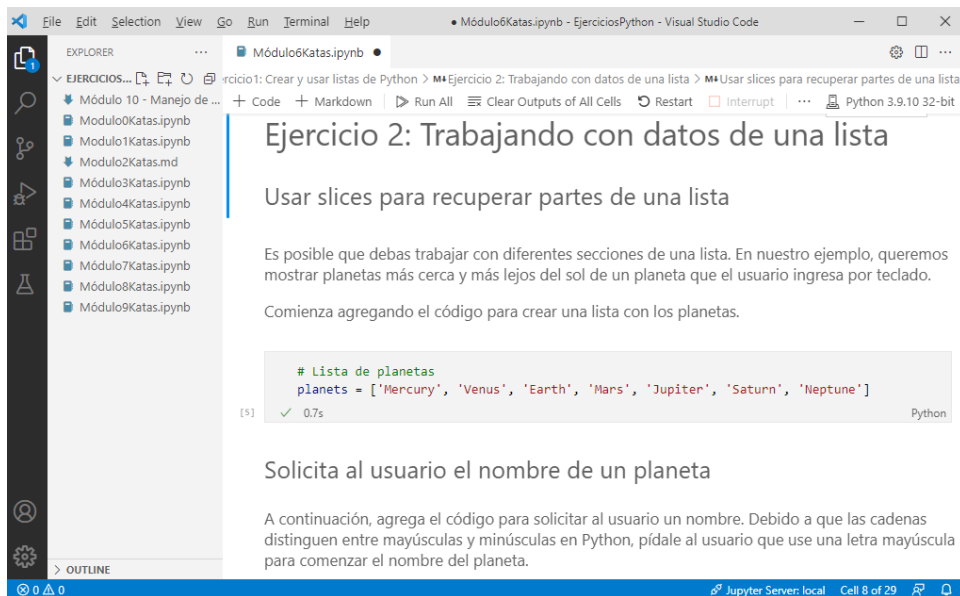
```
 # Agregamos a plutón y mostramos el último elemento
planets.append('Plutón')
print('El último planeta es', planets[-1])
```

[3] ✓ 1.3s Python

... El último planeta es Plutón

Recuerda: Lo importante es intentarlo, puedes auxiliarte de internet, libros, etc. El objetivo de una kata es la comprobación de la comprensión de la lección y su aplicación inmediata a la programación.

Jupyter Server: local Cell 4 of 29



File Edit Selection View Go Run Terminal Help • Módulo6Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

Ejercicio 1: Crear y usar listas de Python > M4 Ejercicio 2: Trabajando con datos de una lista > M4 Usar slices para recuperar partes de una lista

```
 + Code + Markdown + Run All + Clear Outputs of All Cells + Restart + Interrupt + ... Python 3.9.10 32-bit
```

## Ejercicio 2: Trabajando con datos de una lista

### Usar slices para recuperar partes de una lista

Es posible que debas trabajar con diferentes secciones de una lista. En nuestro ejemplo, queremos mostrar planetas más cerca y más lejos del sol de un planeta que el usuario ingresa por teclado.

Comienza agregando el código para crear una lista con los planetas.

```
 # Lista de planetas
planets = ['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Neptune']
```

[5] ✓ 0.7s Python

### Solicita al usuario el nombre de un planeta

A continuación, agrega el código para solicitar al usuario un nombre. Debido a que las cadenas distinguen entre mayúsculas y minúsculas en Python, pídale al usuario que use una letra mayúscula para comenzar el nombre del planeta.

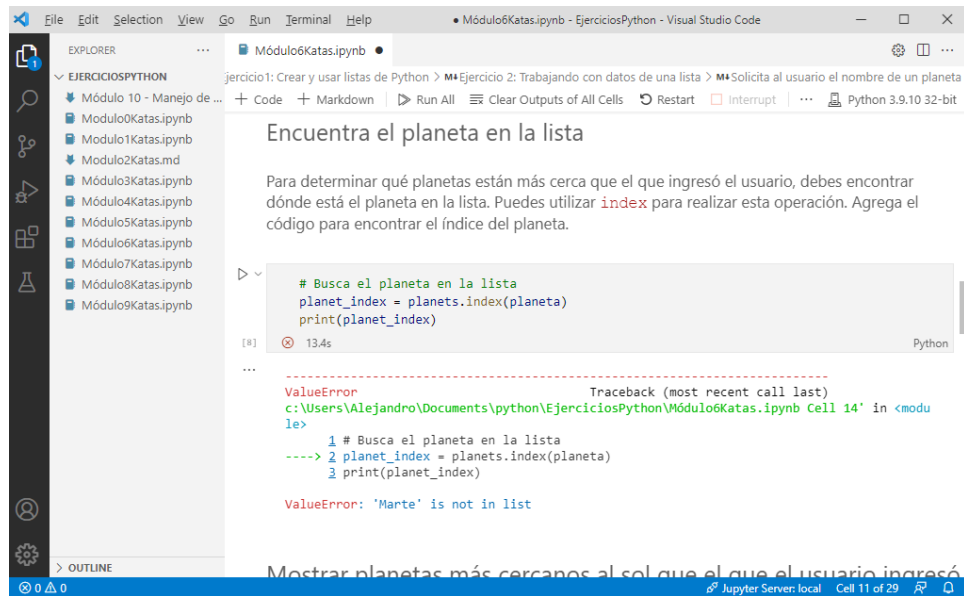
Jupyter Server: local Cell 8 of 29



```
 # Solicitamos el nombre de un planeta *Pista: input()*
planeta = input('Ingresa el nombre de un planeta, debe iniciar con letra mayúscula')
print(planeta)
```

[7] ✓ 4.1s Python

... Marte



Encuentra el planeta en la lista

Para determinar qué planetas están más cerca que el que ingresó el usuario, debes encontrar dónde está el planeta en la lista. Puedes utilizar `index` para realizar esta operación. Agrega el código para encontrar el índice del planeta.

```
# Busca el planeta en la lista
planet_index = planets.index(planeta)
print(planet_index)
```

Traceback (most recent call last):  
c:\Users\Alejandro\Documents\python\EjerciciosPython\Módulo6Katas.ipynb Cell 14' in <modu  
le>  
1 # Busca el planeta en la lista  
----> 2 planet\_index = planets.index(planeta)  
3 print(planet\_index)  
  
ValueError: 'Marte' is not in list

Mostrar planetas más cercanos al sol que el que el usuario ingresó

El error anterior fue debido a que ingrese el planeta Marte en español y este no existe en la lista.

Ahora ingreso “Mars” y devuelve el índice 3.

```
# Solicitamos el nombre de un planeta *Pista: input()*
planeta = input('Ingresa el nombre de un planeta, debe iniciar con letra mayúscula')
print(planeta)
```

Mars

```
# Busca el planeta en la lista
planet_index = planets.index(planeta)
print(planet_index)
```

3

Mostrar planetas más cercanos al sol que el que el usuario ingresó

Con el índice determinado, ahora puedes agregar el código para mostrar los planetas más cercanos al sol.

```
# Muestra los planetas más cercanos al sol
print(planets[0:planet_index])
```

['Mercury', 'Venus', 'Earth']

## Mostrar planetas más alejados del sol que el que el usuario ingresó

Puedes usar el mismo índice para mostrar planetas más alejados del sol. Sin embargo, recuerda que el índice inicial se incluye cuando usas un slice. Como resultado, tendrás que agregar 1 al valor. Agrega el código para mostrar los planetas más alejados del sol.

```
# Muestra los planetas más alejados al sol
print(planets[planet_index + 1:])
```

[12] ✓ 0.1s Python

... ['Jupiter', 'Saturn', 'Neptune']

### Módulo7Katas.ipynb

#### Ejercicio 1: Creación de un bucle "while"

Ejercicio 1: Uso de ciclos `while` en Python

En Python, los ciclos `while` te permiten ejecutar código un número desconocido de veces. Los ciclos examinan una condición booleana y, siempre que la condición sea verdadera, se ejecutará el código dentro del ciclo. Esto es muy útil para situaciones como solicitar valores a un usuario.

En este ejercicio, estás creando una aplicación que solicita a un usuario que ingrese una lista de planetas. En un ejercicio posterior, agregarás código que muestre la lista. Por ahora, crearás solo el código que solicita al usuario la lista de planetas.

**TIP** Dedica unos minutos para tratar de encontrar una solución. Luego desplázate hacia abajo hasta la parte inferior para ver si has logrado compilar el programa de acuerdo con las especificaciones

Comienza agregando dos variables, una para la entrada del usuario, con el nombre `new_planet`, y otra variable para la lista de planetas, denominada `planets`.

Jupyter Server: local Cell 1 of 16

Visual Studio Code interface showing a Jupyter Notebook. The Explorer sidebar on the left lists files from 'Módulo 0' to 'Módulo 9'. The active notebook is 'Módulo7Katas.ipynb'. The current cell contains the following Python code:

```
# Declara dos variables
new_planet = ''
planets = []
```

The cell has been executed successfully, indicated by a green checkmark and a duration of 0.4s. Below the code, the text 'Crea un ciclo while' is displayed. The explanation text reads: 'Comenzando con las variables que acabas de crear, crearás un ciclo **while**. El ciclo **while** se ejecutará *mientras* el **new\_planet** **no** sea igual a la palabra **'done'**. Dentro del ciclo, comprobarás si la variable **new\_planet** contiene un valor, que debería ser el nombre de un planeta. Esta es una forma rápida de ver si el usuario ha introducido un valor. Si lo han hecho, tu código agregará (**append**) ese valor a la variable **planets**. Finalmente, usarás **input** para solicitar al usuario que ingrese un nuevo nombre de planeta o que escriba **done** si ha terminado de ingresar nombres de planeta. Almacenará el valor de **input** en la variable **new\_planet**.'

The next cell contains the following Python code:

```
# Escribe el ciclo while solicitado
while new_planet.lower() != 'done':
    if new_planet:
        planets.append(new_planet)
    new_planet = input('Enter a new planet ')
```

This cell has also been executed successfully, with a duration of 19.2s. The status bar at the bottom indicates 'Jupyter Server: local' and 'Cell 5 of 16'.

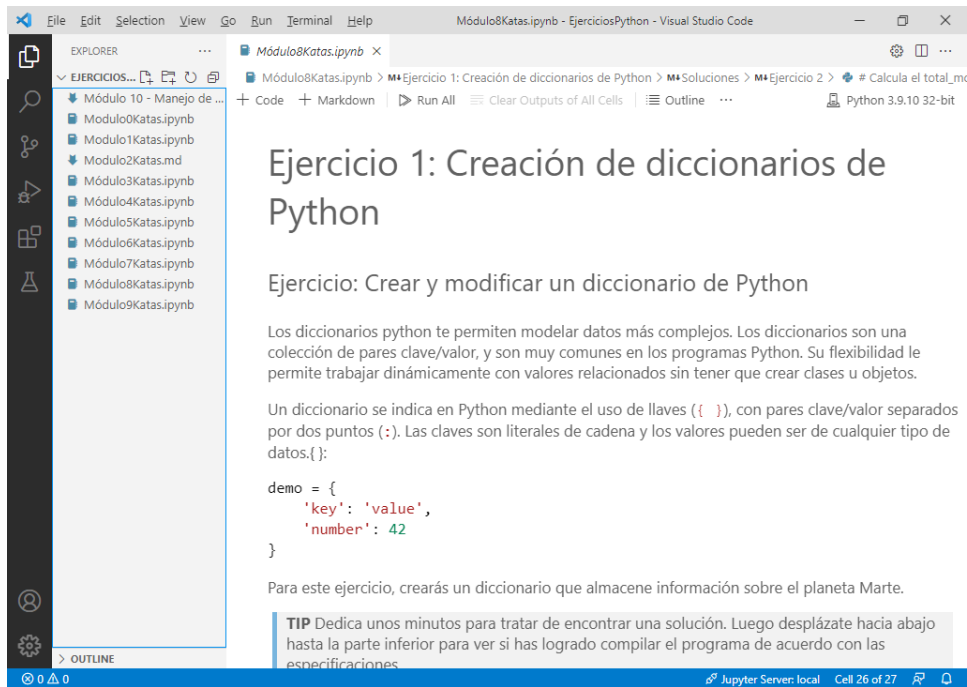
Visual Studio Code interface showing a Jupyter Notebook. The Explorer sidebar on the left lists files from 'Módulo 0' to 'Módulo 9'. The active notebook is 'Módulo7Katas.ipynb'. The current cell contains the following Python code:

```
# Escribe tu ciclo for para iterar en una lista de planetas
for planet in planets:
    print(planet)
```

The cell has been executed successfully, indicated by a green checkmark and a duration of 0.1s. Below the code, the text 'Ejercicio 2: Creación de un ciclo "for"' is displayed. The explanation text reads: 'Ejercicio: - Ciclo para una lista En el ejercicio anterior, creaste código para solicitar a los usuarios que introduzcan una lista de nombres de planetas. En este ejercicio, completarás la aplicación escribiendo código que muestre los nombres de esos planetas. Mostrar la lista de los planetas La variable **planets** almacena los nombres de planeta que ha introducido un usuario. Ahora usarás un ciclo para mostrar esas entradas. Crea un ciclo **for** para iterar sobre la lista **planets**. Puedes usar como nombre de la variable **planet** para cada planeta. Dentro del ciclo **for**, recuerda utilizar **print** para mostrar cada **planet**.'

The status bar at the bottom indicates 'Jupyter Server: local' and 'Cell 9 of 16'.

## Módulo8Katas.ipynb



Visual Studio Code interface showing the Explorer sidebar with a list of files including 'EJERCICIOS...', 'Módulo 10 - Manejo de...', 'Módulo0Katas.ipynb', 'Módulo1Katas.ipynb', 'Módulo2Katas.md', 'Módulo3Katas.ipynb', 'Módulo4Katas.ipynb', 'Módulo5Katas.ipynb', 'Módulo6Katas.ipynb', 'Módulo7Katas.ipynb', 'Módulo8Katas.ipynb', and 'Módulo9Katas.ipynb'. The main editor displays the title 'Ejercicio 1: Creación de diccionarios de Python' and the subtitle 'Ejercicio: Crear y modificar un diccionario de Python'. The text explains that Python dictionaries allow modeling complex data and are common in Python programs. It defines a dictionary as a collection of key-value pairs, with keys being string literals and values being any data type. A code example shows a dictionary 'demo' with keys 'key' and 'number'. A tip at the bottom suggests spending a few minutes to find a solution before scrolling down.

### Ejercicio 1: Creación de diccionarios de Python

#### Ejercicio: Crear y modificar un diccionario de Python

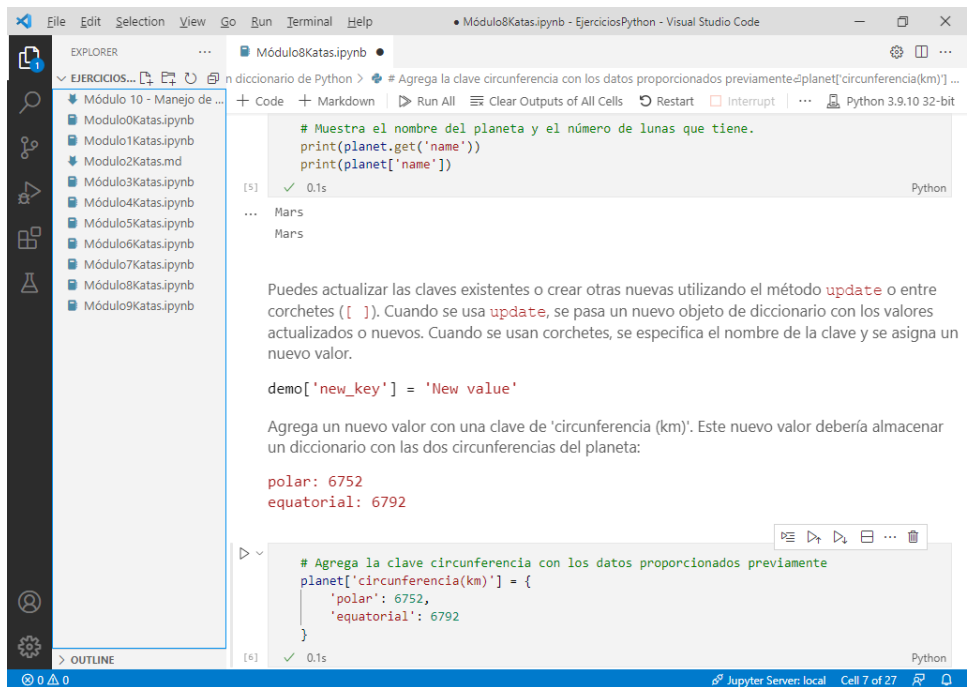
Los diccionarios python te permiten modelar datos más complejos. Los diccionarios son una colección de pares clave/valor, y son muy comunes en los programas Python. Su flexibilidad le permite trabajar dinámicamente con valores relacionados sin tener que crear clases u objetos.

Un diccionario se indica en Python mediante el uso de llaves { }, con pares clave/valor separados por dos puntos (:). Las claves son literales de cadena y los valores pueden ser de cualquier tipo de datos.{ }:

```
demo = {  
    'key': 'value',  
    'number': 42  
}
```

Para este ejercicio, crearás un diccionario que almacene información sobre el planeta Marte.

**TIP** Dedica unos minutos para tratar de encontrar una solución. Luego desplázate hacia abajo hasta la parte inferior para ver si has logrado compilar el programa de acuerdo con las especificaciones.



Visual Studio Code interface showing the second exercise. The Explorer sidebar is the same. The main editor displays the title 'Ejercicio 2: Modificar un diccionario de Python' and the subtitle 'Ejercicio: Agregar la clave circunferencia con los datos proporcionados previamente'. The text explains that dictionaries can be updated or new keys can be added using the 'update' method or square brackets. A code example shows a dictionary 'planet' with keys 'name', 'polar', and 'equatorial'. A tip at the bottom suggests spending a few minutes to find a solution before scrolling down.

### Ejercicio 2: Modificar un diccionario de Python

#### Ejercicio: Agregar la clave circunferencia con los datos proporcionados previamente

Puedes actualizar las claves existentes o crear otras nuevas utilizando el método `update` o entre corchetes [ ]. Cuando se usa `update`, se pasa un nuevo objeto de diccionario con los valores actualizados o nuevos. Cuando se usan corchetes, se especifica el nombre de la clave y se asigna un nuevo valor.

```
demo['new_key'] = 'New value'
```

Agrega un nuevo valor con una clave de 'circunferencia (km)'. Este nuevo valor debería almacenar un diccionario con las dos circunferencias del planeta:

```
polar: 6752  
equatorial: 6792
```

**TIP** Dedica unos minutos para tratar de encontrar una solución. Luego desplázate hacia abajo hasta la parte inferior para ver si has logrado compilar el programa de acuerdo con las especificaciones.

Imprime el nombre del planeta con su circunferencia polar.

```
# Imprime el nombre del planeta con su circunferencia polar.
planet.get('name')
print(f'{planet['name']} diametro polar: {planet['circunferencia(km)']['polar']}')

[17] 0.1s Python
```

Input In [17]  
print(f'{planet['name']} diametro polar: {planet['circunferencia(km)']['polar']}')  
^  
SyntaxError: f-string: unmatched '['

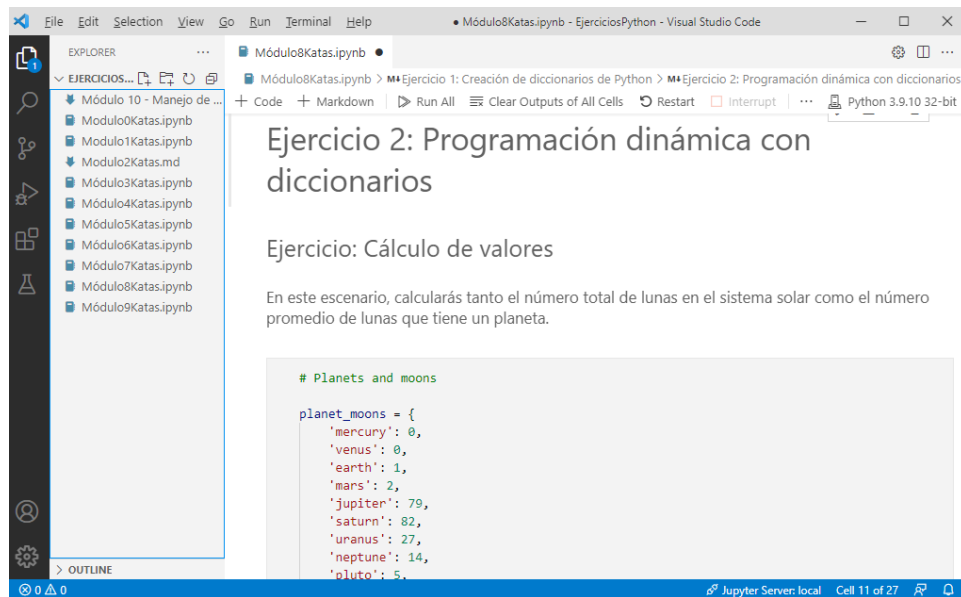
El error es debido a la comilla sencilla, la cambiamos por las comillas ("").

```
# Imprime el nombre del planeta con su circunferencia polar.

print(f'{planet["name"]} tiene una circunferencia polar de: {planet["circunferencia(km)"]["polar"]}')

[20] 0.1s Python
```

Mars tiene una circunferencia polar de: 6752



Los diccionarios de Python te permiten recuperar todos los valores y claves utilizando los métodos **values** y **keys**, respectivamente. Cada método devuelve una lista que contiene los datos, que luego se puede usar como una lista normal de Python. Puedes determinar el número de elementos mediante **len**, e iterar a través de él mediante un ciclo **for**.

Agrega el código a continuación para determinar el número de lunas. Comienza almacenando el valor **values** de **planet\_moons** en una variable denominada **moons**. A continuación, almacena el número de planetas en una variable denominada **planets**.

```
# Añade el código para determinar el número de lunas.
moons = planet_moons.values()
planets = len(planet_moons.keys())

[23] 0.1s Python
```



Agrega el código para contar el número de lunas. Puedes hacerlo creando un ciclo `for` para iterar a través de las lunas `moons` y agregándolos a una variable denominada `total_moons`. Finalmente calcule el promedio dividiendo `total_moons` por `planets` e imprimiendo los resultados.

```
# Agrega el código para contar el número de lunas.
total_moons = 0
for moon in moons:
    total_moons = total_moons + moon

# Calcula el promedio dividiendo el total_moons por el número de planetas
average = total_moons / planets

# Muestra el promedio
print(average)
```

[24] ✓ 0.1s Python

... 17.833333333333332

## Módulo9Katas.ipynb

File Edit Selection View Go Run Terminal Help • Módulo9Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

EJERCICIOSPYTHON

Módulo 10 - Manejo de...

Módulo0Katas.ipynb

Módulo1Katas.ipynb

Módulo2Katas.md

Módulo3Katas.ipynb

Módulo4Katas.ipynb

Módulo5Katas.ipynb

Módulo6Katas.ipynb

Módulo7Katas.ipynb

Módulo8Katas.ipynb

Módulo9Katas.ipynb

Módulo9Katas.ipynb

gumentos en funciones > # Actualiza la función: def generate\_report(main\_tank, external\_tank, hydrogen\_tank): return f"""Fuel ...

+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Interrupt ⌵ Python 3.9.10 32-bit

Ejercicio: Uso de funciones en Python

Ejercicio 1: Trabajar con argumentos en funciones

Los argumentos requeridos en las funciones se utilizan cuando las funciones necesitan que esos argumentos funcionen correctamente. En este ejercicio, construirás un informe de combustible que requiere información de varias ubicaciones de combustible en todo el cohete.

TIP Dedica unos minutos para tratar de encontrar una solución. Luego desplázate hacia abajo hasta la parte inferior para ver si has logrado compilar el programa de acuerdo con las especificaciones

Comienza por crear una función que necesite tres lecturas de combustible y devuelva un informe:

```
# Función para leer 3 tanques de combustible y muestre el promedio
def generate_report(main_tank, external_tank, hydrogen_tank):
    total_average = (main_tank + external_tank + hydrogen_tank) / 3
    return f"""Fuel Report:
    Total Average: {total_average}%
    Main tank: {main_tank}%
    External tank: {external_tank}%
    Hydrogen tank: {hydrogen_tank}%
    """
```

[1] ✓ 0.9s Python

File Edit Selection View Go Run Terminal Help • Módulo9Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

▼ EJERCICIOS...  
Módulo 10 - Manejo de...  
Módulo9Katas.ipynb  
Módulo1Katas.ipynb  
Módulo2Katas.md  
Módulo3Katas.ipynb  
Módulo4Katas.ipynb  
Módulo5Katas.ipynb  
Módulo6Katas.ipynb  
Módulo7Katas.ipynb  
Módulo8Katas.ipynb  
Módulo9Katas.ipynb

Módulo9Katas.ipynb

gumentos en funciones > # Actualiza la funcióndef generate\_report(main\_tank, external\_tank, hydrogen\_tank): return f"""Fuel ...  
+ Code + Markdown ▶ Run All Clear Outputs of All Cells Restart Interrupt ... Python 3.9.10 32-bit

Ahora que hemos definido la función de informes, vamos a comprobarlo. Para esta misión, los tanques no están llenos:

```
# Llamamos a la función que genera el reporte print(funcion(tanque1, tanque2, tanque3))  
print(generate_report(80, 70, 85))
```

[2] ✓ 0.2s Python

... Fuel Report:  
Total Average: 78.33333333333333%  
Main tank: 80%  
External tank: 70%  
Hydrogen tank: 85%

En lugar de simplemente crear el informe, la función también está calculando el promedio. Mejora la legibilidad extrayendo el cálculo promedio de la función en una nueva función para que el promedio se pueda hacer de forma independiente:

> OUTLINE

0 0 Jupyter Server: local Cell 10 of 29

```
# Función promedio  
def average(values):  
    total = sum(values)  
    number_of_items = len(values)  
    return total / number_of_items  
  
# Test the averaging function with a list of integers:  
average([80, 85, 81])
```

[3] ✓ 0.1s Python

... 82.0

Ahora actualiza la función de informes para llamando a la nueva función del promedio:

▶

```
# Actualiza la función  
def generate_report(main_tank, external_tank, hydrogen_tank):  
    return f"""Fuel Report:  
    Total Average: {average([main_tank, external_tank, hydrogen_tank])}%  
    Main tank: {main_tank}%  
    External tank: {external_tank}%  
    Hydrogen tank: {hydrogen_tank}%  
    """
```

```
# Call the updated function again with different values  
print(generate_report(88, 76, 70))
```

[4] ✓ 0.1s Python

... Fuel Report:  
Total Average: 78.0%  
Main tank: 88%  
External tank: 76%  
Hydrogen tank: 70%

File Edit Selection View Go Run Terminal Help • Módulo9Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

Python > Ejercicio 2: Trabajo con argumentos de palabra clave > Ejercicio : Trabajar con argumentos de palabras clave en funciones

Python 3.9.10 32-bit

### Ejercicio : Trabajar con argumentos de palabras clave en funciones

En este ejercicio, construirás un informe de cohete que requiere varias piezas de información, como el tiempo hasta el destino, el combustible a la izquierda y el nombre del destino. Comienza por crear una función que creará un informe preciso de la misión:

```
# Función con un informe preciso de la misión. Considera hora de prelanzamiento, tiempo de
def mission_report(pre_launch_time, flight_time, destination, external_tank, main_tank):
    return f"""
    Mission to {destination}
    Total travel time: {pre_launch_time + flight_time} minutes
    Total fuel left: {external_tank + main_tank} gallons
    """

print(mission_report(14, 51, "Moon", 200000, 300000))
```

[5] ✓ 0.1s Python

...

Mission to Moon  
Total travel time: 65 minutes  
Total fuel left: 500000 gallons

Jupyter Server: local Cell 13 of 29

File Edit Selection View Go Run Terminal Help • Módulo9Katas.ipynb - EjerciciosPython - Visual Studio Code

EXPLORER

Python > Ejercicio 2: Trabajo con argumentos de palabra clave > Ejercicio : Trabajar con argumentos de palabras clave en funciones

Python 3.9.10 32-bit

### Ejercicio : Trabajar con argumentos de palabras clave en funciones

La función es problemática porque no se puede adaptar para minutos adicionales o tanques adicionales de combustible. Hazlo más flexible permitiendo cualquier número de pasos basados en el tiempo y cualquier número de tanques. En lugar de usar `*args` y `**kwargs`, aprovecha el hecho de que puedes usar cualquier nombre que desees. Asegurate que la función sea más legible mediante el uso de nombres de variables que están asociados con las entradas, como `*minutes` y `**fuel_reservoirs`:

```
# Escribe tu nueva función de reporte considerando lo anterior
def mission_report(destination, *minutes, **fuel_reservoirs):
    return f"""
    Mission to {destination}
    Total travel time: {sum(minutes)} minutes
    Total fuel left: {sum(fuel_reservoirs.values())}
    """

print(mission_report("Moon", 10, 15, 51, main=300000, external=200000))
```

[6] ✓ 0.1s Python

...

Mission to Moon  
Total travel time: 76 minutes  
Total fuel left: 500000

Jupyter Server: local Cell 13 of 29

Debido a que el combustible que queda en los tanques es específico de cada tanque, actualiza la función para usar el nombre de cada tanque en el informe:

```
# Escribe tu nueva función
def mission_report(destination, *minutes, **fuel_reservoirs):
    main_report = f""
    Mission to {destination}
    Total travel time: {sum(minutes)} minutes
    Total fuel left: {sum(fuel_reservoirs.values())}
    ""
    for tank_name, gallons in fuel_reservoirs.items():
        main_report += f"{tank_name} tank --> {gallons} gallons left\n"
    return main_report

print(mission_report("Moon", 8, 11, 55, main=300000, external=200000))
```

[7] ✓ 0.2s Python

...

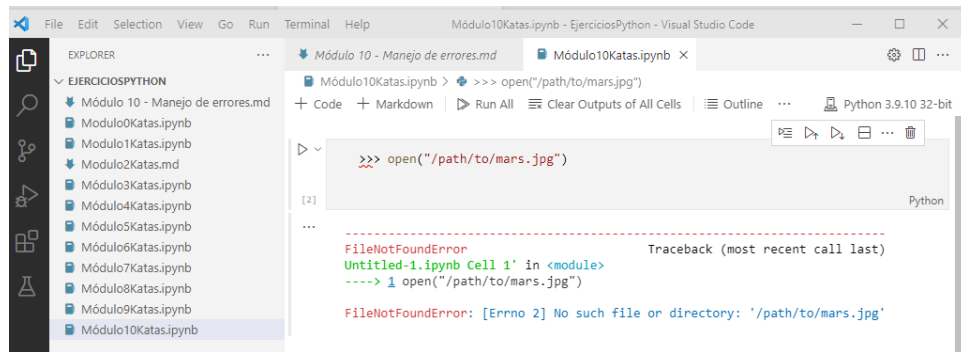
Mission to Moon  
Total travel time: 74 minutes  
Total fuel left: 500000  
main tank --> 300000 gallons left  
external tank --> 200000 gallons left

## 📁 Módulo 10 - Manejo de errores.md

Módulo 10 - Manejo de errores.md

Módulo 10 - Manejo de errores.md > # Introducción

```
1 # Introducción
2
3 Incluso el código mejor escrito contendrá errores. Los errores pueden
4 producirse debido a actualizaciones, archivos movidos u otros cambios
5 inesperados. Afortunadamente, Python ofrece una amplia compatibilidad
6 para el seguimiento y el control de errores.
7
8 ## Escenario: Creación de un programa de cohetes
9
10 Imagina que es un desarrollador que está creando un programa para un
11 cohete. El programa debe leer un archivo de configuración para
12 asegurarse de que se carga la configuración adecuada. La lectura del
13 archivo puede producir un error si falta el archivo o tiene otros
14 problemas. En este módulo, explorarás cómo crear el programa.
15
16 ## ¿Qué descubrirás?
17
18 Al término de este módulo, sabrás hacer lo siguiente:
19
20 * Leer y usar la salida de errores de las excepciones
21 * Controlar correctamente las excepciones
22 * Generar excepciones con mensajes de error útiles
23 * Usar excepciones para controlar el flujo de un programa
24
25 ## ¿Cuál es el objetivo principal?
26
27 En este módulo, descubrirás cómo usar la salida de excepción para la
28 depuración, cómo detectar y generar excepciones, y cómo afecta a la
29 lógica de un programa cuando se producen excepciones.
```



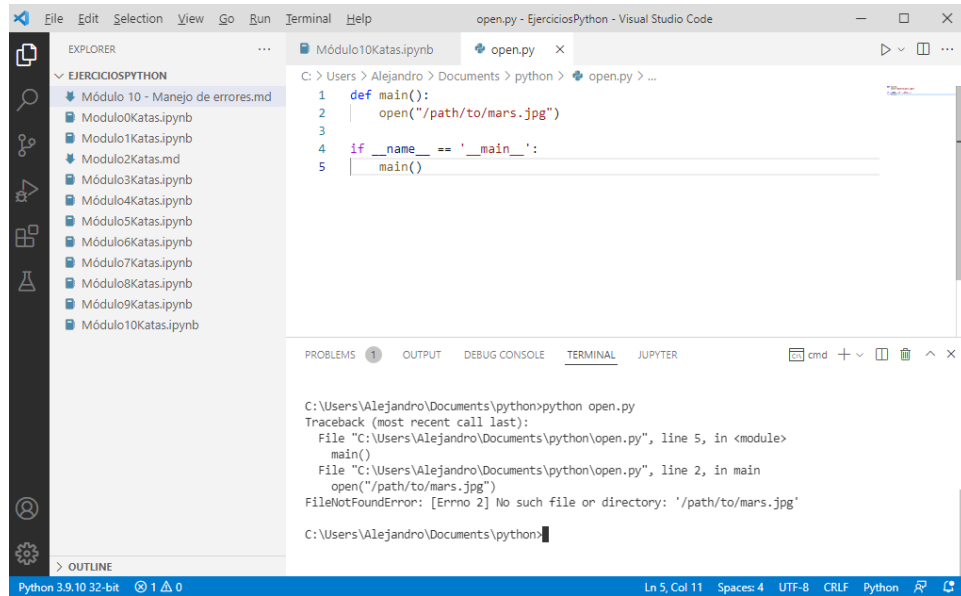
Visual Studio Code interface showing a Jupyter Notebook. The Explorer sidebar on the left lists files under 'EJERCICIOSPYTHON', including 'Módulo 10 - Manejo de errores.md' and various 'MóduloXKatas.ipynb' files. The active notebook cell contains the following code:

```
>>> open("/path/to/mars.jpg")
```

The output of the cell shows a `FileNotFoundError` traceback:

```
FileNotFoundError                                Traceback (most recent call last)
Untitled-1.ipynb Cell 1 in <module>
----> 1 open("/path/to/mars.jpg")

FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
```



Visual Studio Code interface showing a Python script named 'open.py'. The Explorer sidebar on the left is the same as in the first screenshot. The active editor shows the following code:

```
1 def main():
2     open("/path/to/mars.jpg")
3
4 if __name__ == '__main__':
5     main()
```

The TERMINAL panel at the bottom shows the command `C:\Users\Alejandro\Documents\python>python open.py` and the resulting `FileNotFoundError` traceback, which is identical to the one in the first screenshot.



A Jupyter Notebook cell showing a try-except block designed to handle a `FileNotFoundError`:

```
>>> try:
...     open('config.txt')
... except FileNotFoundError:
...     print("Couldn't find the config.txt file!")
```

The output of the cell is:

```
[1] ✓ 0.2s
... Couldn't find the config.txt file!
```

The screenshot shows the Visual Studio Code interface with the Explorer panel on the left displaying a project named 'EJERCICIOSPYTHON'. The file 'Módulo10Katas.ipynb' is selected. The main editor shows a Python file named 'config.py' with the following code:

```
1 def main():
2     try:
3         configuration = open('config.txt')
4     except FileNotFoundError:
5         print("Couldn't find the config.txt file!")
6
7
8 if __name__ == '__main__':
9     main()
```

The TERMINAL panel at the bottom shows the command 'python config.py' and the resulting error:

```
C:\Users\Alejandro\Documents\python>python config.py
Traceback (most recent call last):
  File "C:\Users\Alejandro\Documents\python\config.py", line 9, in <module>
    main()
  File "C:\Users\Alejandro\Documents\python\config.py", line 3, in main
    configuration = open('config.txt')
PermissionError: [Errno 13] Permission denied: 'config.txt'

C:\Users\Alejandro\Documents\python>
```

The status bar at the bottom indicates 'Python 3.9.10 32-bit' and 'Ln 3, Col 41'.

The screenshot shows the same Visual Studio Code interface, but the code in 'config.py' has been modified to catch a general 'Exception' instead of 'FileNotFoundError':

```
1 def main():
2     try:
3         configuration = open('config.txt')
4     except Exception:
5         print("Couldn't find the config.txt file!")
6
7
8 if __name__ == '__main__':
9     main()
```

The TERMINAL panel now shows the command 'python config.py' and the resulting error:

```
C:\Users\Alejandro\Documents\python>python config.py
Couldn't find the config.txt file!

C:\Users\Alejandro\Documents\python>
```

The status bar at the bottom indicates 'Python 3.9.10 32-bit' and 'Ln 7, Col 11'.

The screenshot shows the Visual Studio Code interface with the Explorer on the left displaying a folder named 'EJERCICIOSPYTHON'. The file 'Módulo10Katas.ipynb' is selected. The main editor shows a Python file named 'config.py' with the following code:

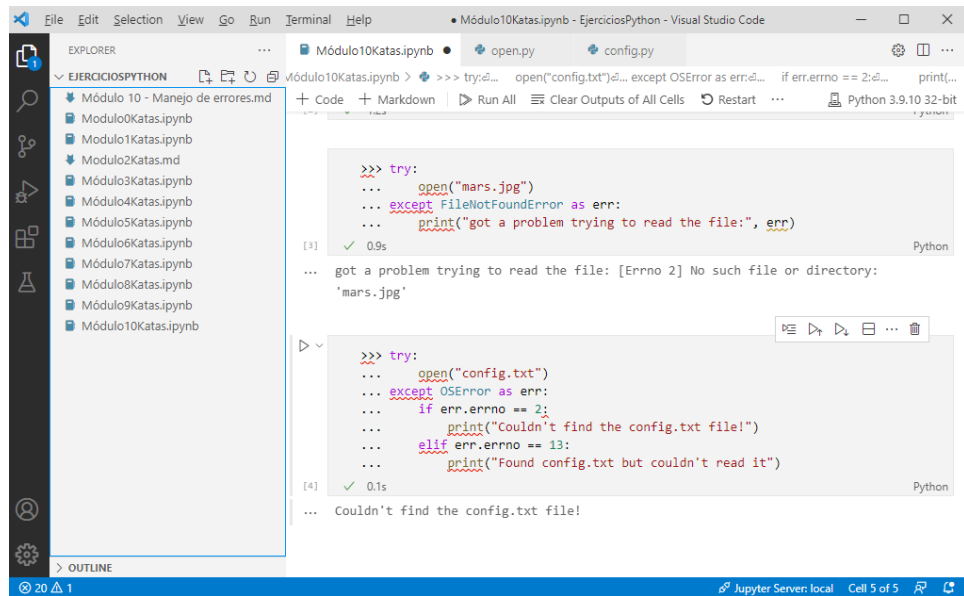
```
1 def main():
2     try:
3         configuration = open('config.txt')
4     except FileNotFoundError:
5         print("Couldn't find the config.txt file!")
6     except PermissionError:
7         print("Found config.txt but Permission was denied, couldn't read")
8
9     if __name__ == '__main__':
10        main()
```

The terminal at the bottom shows the command 'python config.py' being executed, resulting in the output: 'Found config.txt but it is a directory, couldn't read it'. The status bar at the bottom indicates 'Python 3.9.10 32-bit' and 'Ln 6, Col 28'.

The screenshot shows the Visual Studio Code interface with the Explorer on the left displaying a folder named 'EJERCICIOSPYTHON'. The file 'Módulo10Katas.ipynb' is selected. The main editor shows a Python file named 'config.py' with the following code:

```
1 def main():
2     try:
3         configuration = open('config.txt')
4     except FileNotFoundError:
5         print("Couldn't find the config.txt file!")
6     except PermissionError:
7         print("Found config.txt but Permission was denied, couldn't read")
8     except (BlockingIOError, TimeoutError):
9         print("Filesystem under heavy load, can't complete reading config")
10
11     if __name__ == '__main__':
12        main()
```

The terminal at the bottom shows the command 'python config.py' being executed, resulting in the output: 'Found config.txt but Permission was denied, couldn't read it'. The status bar at the bottom indicates 'Python 3.9.10 32-bit' and 'Ln 7, Col 58'.





```

-----
RuntimeError                                Traceback (most recent call last)
c:\Users\Alejandro\Documents\python\EjerciciosPython\Módulo10Katas.ipynb Cell 9'
in <module>
----> 1 water_left(5, 100, 2)

c:\Users\Alejandro\Documents\python\EjerciciosPython\Módulo10Katas.ipynb Cell 8'
in water_left(astronauts, water_left, days_left)
      4 total_water_left = water_left - total_usage
      5 if total_water_left < 0:
----> 6     raise RuntimeError(f"There is not enough water for {astronauts} astro
nauts after {days_left} days!")
      7 return f"Total water left after {days_left} days is: {total_water_left} l
iters"

```

RuntimeError: There is not enough water for 5 astronauts after 2 days!

```

ater_left(astronauts, water_left, days_left):
or argument in [astronauts, water_left, days_left]:
    try:
        # If argument is an int, the following operation will work
        argument / 10
    except TypeError:
        # TypeError will be raised only if it isn't the right type
        # Raise the same exception but with a better error message
        raise TypeError(f"All arguments must be of type int, but received: '{argume
aily_usage = astronauts * 11
otal_usage = daily_usage * days_left
otal_water_left = water_left - total_usage
f total_water_left < 0:
    raise RuntimeError(f"There is not enough water for {astronauts} astronauts afte
return f"Total water left after {days_left} days is: {total_water_left} liters"

[9] ✓ 0.1s Python

```

```

>>> water_left("3", "200", None)

[10] ✗ 0.6s Python

```

```

-----
TypeError                                Traceback (most recent call last)
c:\Users\Alejandro\Documents\python\EjerciciosPython\Módulo10Katas.ipynb Cell 10'
in water_left(astronauts, water_left, days_left)
      3 try:
      4     # If argument is an int, the following operation will work
----> 5     argument / 10
      6 except TypeError:
      7     # TypeError will be raised only if it isn't the right type
      8     # Raise the same exception but with a better error message

```

TypeError: unsupported operand type(s) for /: 'str' and 'int'

During handling of the above exception, another exception occurred:

```

TypeError                                Traceback (most recent call last)
c:\Users\Alejandro\Documents\python\EjerciciosPython\Módulo10Katas.ipynb Cell 11'
in <module>
----> 1 water_left("3", "200", None)

```

```
c:\Users\Alejandro\Documents\python\EjerciciosPython\Módulo10Katas.ipynb Cell 10'
in water_left(astronauts, water_left, days_left)
     5         argument / 10
     6     except TypeError:
     7         # TypeError will be raised only if it isn't the right type
     8         # Raise the same exception but with a better error message
----> 9         raise TypeError(f"All arguments must be of type int, but receive
d: '{argument}'")
    10 daily_usage = astronauts * 11
    11 total_usage = daily_usage * days_left
```

TypeError: All arguments must be of type int, but received: '3'