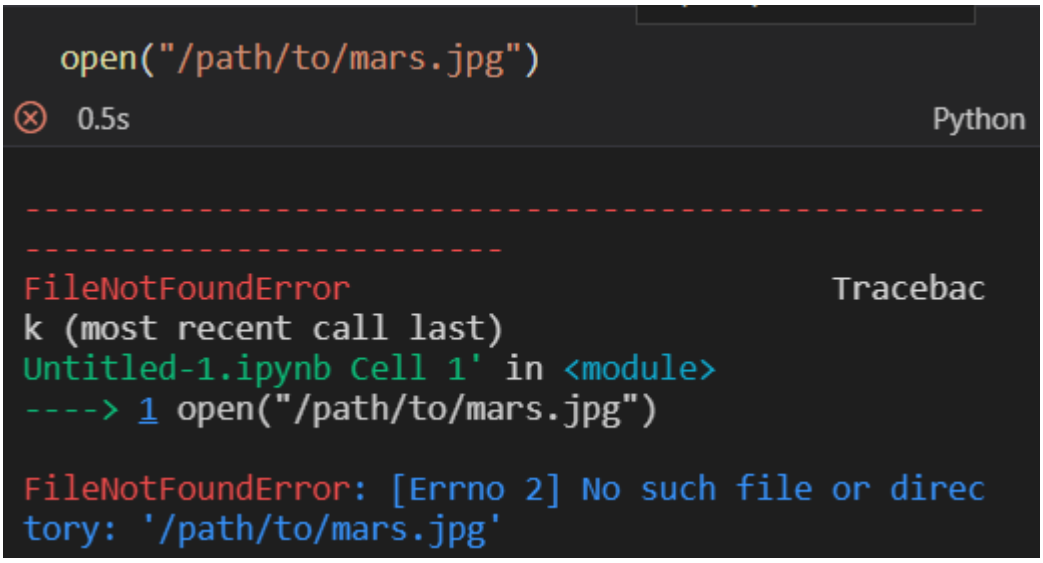
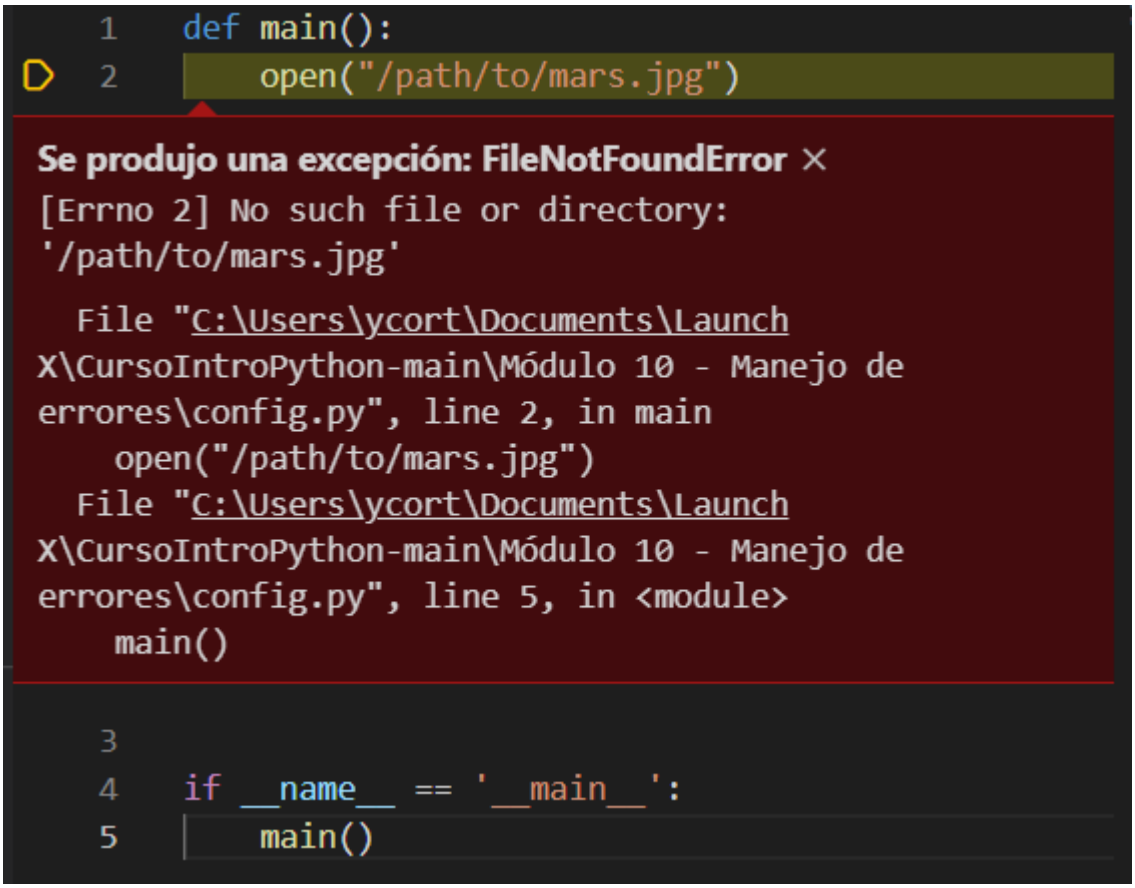
## Tracebacks

Si intentamos en un notebook, abrir un archivo inexistente sucede lo siguiente:




Intenta crear un archivo de Python y asígnale el nombre *open.py*, con el contenido siguiente:



## Controlando las excepciones

```
1   try:
2       open('config.txt')
```

**Se produjo una excepción: FileNotFoundError** ×

```
[Errno 2] No such file or directory:
'/path/to/mars.jpg'
  File "C:\Users\ycort\Documents\Launch
X\CursoIntroPython-main\Módulo 10 - Manejo de
errores\config.py", line 2, in main
    open("/path/to/mars.jpg")
  File "C:\Users\ycort\Documents\Launch
X\CursoIntroPython-main\Módulo 10 - Manejo de
errores\config.py", line 5, in <module>
    main()
```

```
3   except FileNotFoundError:
4       print("Couldn't find the config.txt file!")
```

```
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()
```

```
def main():
    try:
        configuration = open('config.txt')
    except Exception:
        print("Couldn't find the config.txt file!")
```

```python
def main():
    try:
        configuration = open('config.txt')
    except FileNotFoundError:
        print("Couldn't find the config.txt file!")
    except IsADirectoryError:
        print("Found config.txt but it is a directory, couldn't read it")
```

## Generación de excepciones

```python
def water_left(astronauts, water_left, days_left):
    daily_usage = astronauts * 11
    total_usage = daily_usage * days_left
    total_water_left = water_left - total_usage
    return f"Total water left after {days_left} days is: {total_water_left

water_left(5, 100, 2)
'Total water left after 2 days is: -10 liters'
```

```python
def water_left(astronauts, water_left, days_left):
    daily_usage = astronauts * 11
    total_usage = daily_usage * days_left
    total_water_left = water_left - total_usage
    if total_water_left < 0:
        raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
    return f"Total water left after {days_left} days is: {total_water_left} liters"
```

```
1  def water_left(astronauts, water_left, days_left):
2      daily_usage = astronauts * 11
3      total_usage = daily_usage * 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} astronauts after {days_left} days!")

Se produjo una excepción: RuntimeError ×
There is not enough water for 5 astronauts after 2 days!
  File "C:\Users\ycort\Documents\Launch X\CursoIntroPython-main\Módulo 10 - Manejo de errores\modulo10 - astronautas.py", line 6, in water_left
    raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
  File "C:\Users\ycort\Documents\Launch X\CursoIntroPython-main\Módulo 10 - Manejo de errores\modulo10 - astronautas.py", line 9, in <module>
    water_left(5, 100, 2)

7      return f"Total water left after {days_left} days is: {total_water_left} liters"
8
9  water_left(5, 100, 2)
```

```python
def water_left(astronauts, water_left, days_left):
    for argument in [astronauts, water_left, days_left]:
        try:
            # If argument is an int, the following operation will work
            argument / 10
        except TypeError:
            # TypError 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: '{argument}'")
    daily_usage = astronauts * 11
    total_usage = daily_usage * days_left
    total_water_left = water_left - total_usage
    if total_water_left < 0:
        raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
    return f"Total water left after {days_left} days is: {total_water_left} liters"
```

```
 7              # TypError 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 received: '{argument}'")

Se produjo una excepción: TypeError ×
All arguments must be of type int, but received: '3'
  File "C:\Users\ycort\Documents\Launch X\CursoIntroPython-main\Módulo 10 - Manejo de errores\modulo10 - astronautas.py", line 5, in water_left
    argument / 10

During handling of the above exception, another exception occurred:

  File "C:\Users\ycort\Documents\Launch X\CursoIntroPython-main\Módulo 10 - Manejo de errores\modulo10 - astronautas.py", line 9, in water_left
    raise TypeError(f"All arguments must be of type int, but received: '{argument}'")
  File "C:\Users\ycort\Documents\Launch X\CursoIntroPython-main\Módulo 10 - Manejo de errores\modulo10 - astronautas.py", line 17, in <module>
    water_left("3", "200", None)

10      daily_usage = astronauts * 11
11      total_usage = daily_usage * days_left
12      total_water_left = water_left - total_usage
13      if total_water_left < 0:
14          raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
15      return f"Total water left after {days_left} days is: {total_water_left} liters"
16
17  water_left("3", "200", None)
```