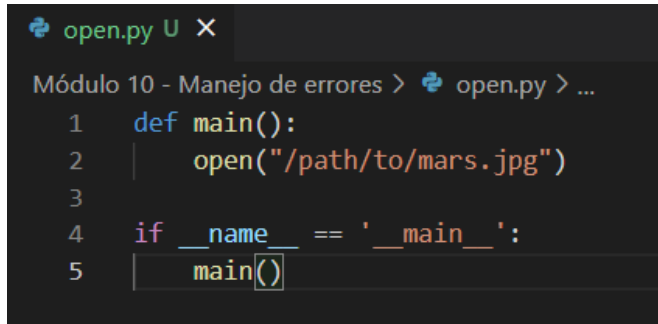


Kata Módulo 10 - Manejo de errores

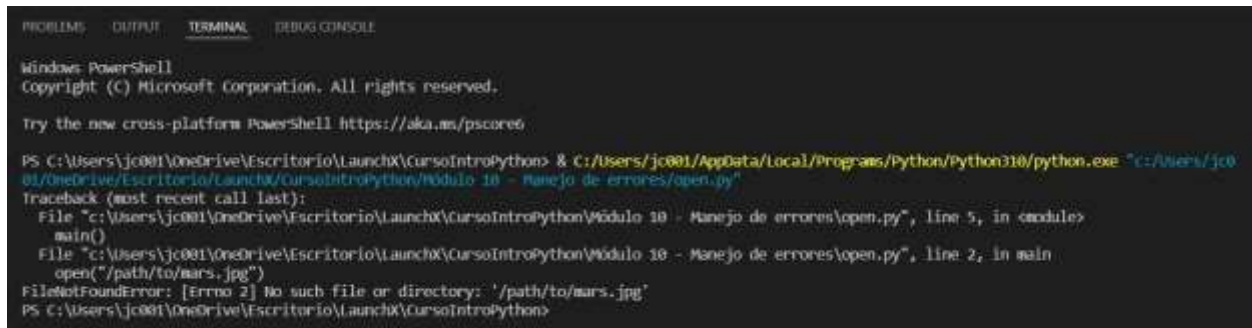
Tracebacks

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



```
open.py U X
Módulo 10 - Manejo de errores > open.py > ...
1 def main():
2     open("/path/to/mars.jpg")
3
4 if __name__ == '__main__':
5     main()
```

Se trata de una sola función main() que abre el archivo inexistente, como antes. Al final, esta función usa un asistente de Python que indica al intérprete que ejecute la función main() cuando se le llama en el terminal. Ejecútala con Python y podrás comprobar el siguiente mensaje de error:



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> & C:/Users/jc001/AppData/Local/Programs/Python/Python310/python.exe "C:/Users/jc001/OneDrive/Escritorio\LaunchX\CursoIntroPython\Módulo 10 - Manejo de errores/open.py"
Traceback (most recent call last):
  File "C:/Users/jc001/OneDrive/Escritorio\LaunchX\CursoIntroPython\Módulo 10 - Manejo de errores/open.py", line 5, in <module>
    main()
  File "C:/Users/jc001/OneDrive/Escritorio\LaunchX\CursoIntroPython\Módulo 10 - Manejo de errores/open.py", line 2, in main
    open("/path/to/mars.jpg")
FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython>
```

La salida de error tiene más sentido ahora. Las rutas de acceso apuntan a un único archivo denominado open.py. La salida menciona que el error se inicia en la línea 5, que incluye la llamada a main(). A continuación, la salida sigue el error a la línea 2 en la llamada de función open(). Y, por último, FileNotFoundError notifica de nuevo que el archivo o el directorio no existen.

Los tracebacks casi siempre incluyen la información siguiente:

- Todas las rutas de acceso de archivo implicadas, para cada llamada a cada función.
- Los números de línea asociados a cada ruta de acceso de archivo.
- Los nombres de las funciones, métodos o clases implicados en la generación de una excepción.
- El nombre de la excepción que se ha producido.

Controlando las excepciones

Try y Except de los bloques

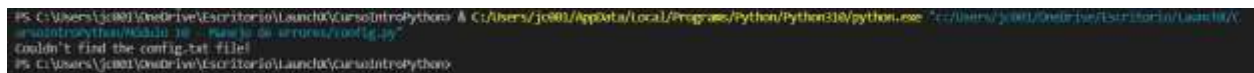
Al querer abrir un archivo que no existe, arroja un error en el código lo cual es lo siguiente:

Después de la palabra clave try, agregamos código que tenga la posibilidad de producir una excepción. A continuación, agregamos la palabra clave except junto con la posible excepción, seguida de cualquier código que deba ejecutarse cuando se produce esa condición. Puesto que config.txt no existe en el sistema, Python imprime que el archivo de configuración no está ahí. El bloque try y except, junto con un mensaje útil, evita un seguimiento y sigue informando al usuario sobre el problema.

Aunque es común un archivo que no existe, no es el único error que podemos encontrar. Los permisos de archivo no válidos pueden impedir la lectura de un archivo, incluso si este existe. Vamos a crear un archivo de Python denominado config.py. El archivo tiene código que busca y lee el archivo de configuración del sistema de navegación:



```
open.py U  config.py U X
Módulo 10 - Manejo de errores > config.py > ...
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()
```



```
PS C:\Users\jcm01\OneDrive\Escritorio\Launch\cursointropython> C:\Users\jcm01\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\jcm01\OneDrive\Escritorio\Launch\cursointropython\Modulo 10 - Manejo de errores\config.py"
Couldn't find the config.txt file!
PS C:\Users\jcm01\OneDrive\Escritorio\Launch\cursointropython>
```

El problema ahora es que el mensaje de error es incorrecto. El archivo existe, pero tiene permisos diferentes y Python no puede leerlo. Cuando se trata con errores de software, puede resultar frustrante tener errores que hagan lo siguiente:

- No indiquen cuál es el problema real.
- Proporcionen una salida que no coincida con el problema real.
- No sugieran lo que se puede hacer para corregir el problema.

Vamos a corregir este fragmento de código para abordar todas estas frustraciones. Revertiremos la detección de `FileNotFoundError` y luego agregamos otro bloque `except` para detectar `PermissionError`:

```
open.py U  config.py U X
Módulo 10 - Manejo de errores > config.py > main
1 def main():
2     try:
3         configuration = open('config.txt')
4     except FileNotFoundError:
5         print("Couldn't find the config.txt file!")
6     except IsADirectoryError:
7         print("Found config.txt but it is a directory, couldn't read it")

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> & C:/Users/jc001/AppData/Local/Programs/Python/Python310/python.exe
"C:/Users/jc001/OneDrive/Escritorio\LaunchX\CursoIntroPython/Módulo 10 - Manejo de errores/config.py"
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> █
```

Eliminamos el archivo config.txt para asegurarnos de que se alcanza el primer bloque `except` en su lugar:

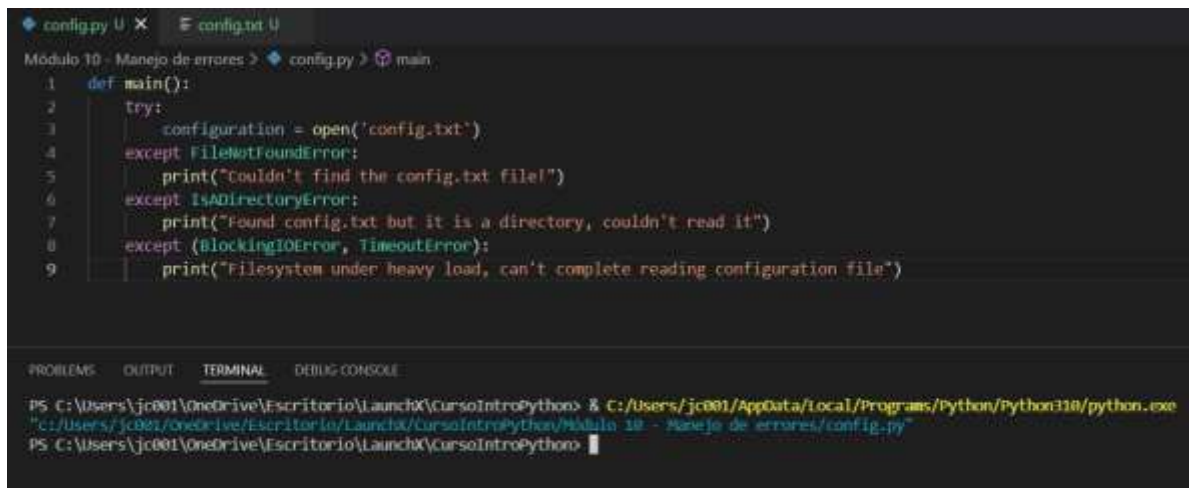
```
config.py U X  config.txt U
Módulo 10 - Manejo de errores > config.py > main
1 def main():
2     try:
3         configuration = open('config.txt')
4     except FileNotFoundError:
5         print("Couldn't find the config.txt file!")
6     except IsADirectoryError:
7         print("Found config.txt but it is a directory, couldn't read it")

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> rm -f config.txt
Remove-Item : Parameter cannot be processed because the parameter name 'f' is ambiguous. Possible matches include: -Filter
-Force.
At line:1 char:4
+ rm -f config.txt
+ ~~~
+ CategoryInfo          : InvalidArgument: (:) [Remove-Item], ParameterBindingException
+ FullyQualifiedErrorId : AmbiguousParameter,Microsoft.PowerShell.Commands.RemoveItemCommand

PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> python config.py
C:\Users\jc001\AppData\Local\Programs\Python\Python310\python.exe: can't open file 'C:\Users\jc001\OneDrive\Escritorio\L
aunchX\CursoIntroPython\config.py': [Errno 2] No such file or directory
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> █
```

Cuando los errores son de una naturaleza similar y no es necesario controlarlos individualmente, puedes agrupar las excepciones como una usando paréntesis en la línea except. Por ejemplo, si el sistema de navegación está bajo cargas pesadas y el sistema de archivos está demasiado ocupado, tiene sentido detectar BlockingIOError y TimeoutError juntos:



```
config.py x config.txt U
Módulo 10 - Manejo de errores > config.py > main
1 def main():
2     try:
3         configuration = open('config.txt')
4     except FileNotFoundError:
5         print("Couldn't find the config.txt file!")
6     except IsADirectoryError:
7         print("Found config.txt but it is a directory, couldn't read it")
8     except (BlockingIOError, TimeoutError):
9         print("Filesystem under heavy load, can't complete reading configuration file")

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> & C:/Users/jc001/AppData/Local/Programs/Python/Python310/python.exe
"C:/Users/jc001/OneDrive/Escritorio/LaunchX/CursoIntroPython/Módulo 10 - Manejo de errores/config.py"
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython>
```

Aquí ya existe el archivo config.txt y no arroja ningún caso de error.

Generación de excepciones

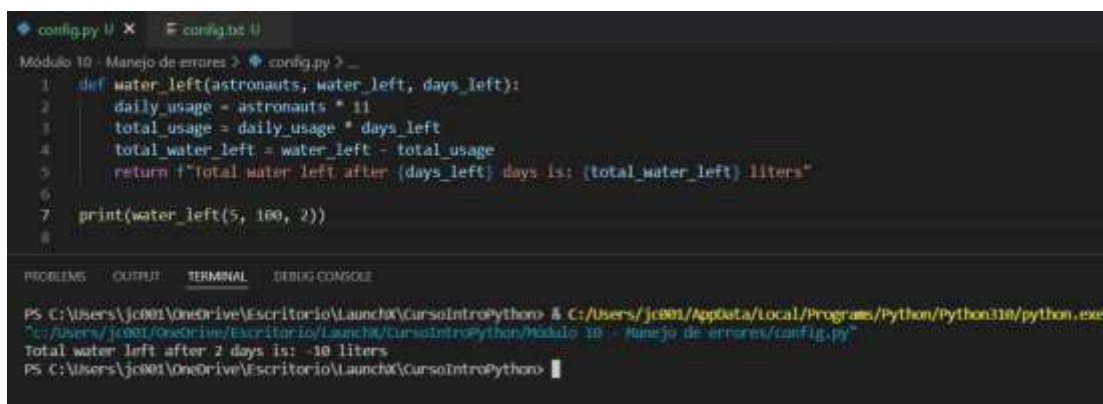
Ahora que tienes una buena comprensión de los tracebacks y el control de excepciones, vamos a revisar la generación de excepciones.

Es posible que ya conozcas una situación que podría provocar una condición de error al escribir código. En estas situaciones, resulta útil generar excepciones que permitan que otro código comprenda cuál es el problema.

La generación de excepciones también puede ayudar en la toma de decisiones para otro código. Como hemos visto antes, en función del error, el código puede tomar decisiones inteligentes para resolver, solucionar o ignorar un problema.

Los astronautas limitan su uso de agua a unos 11 litros al día. Vamos a crear una función que, con base al número de astronautas, pueda calcular la cantidad de agua quedará después de un día o más:

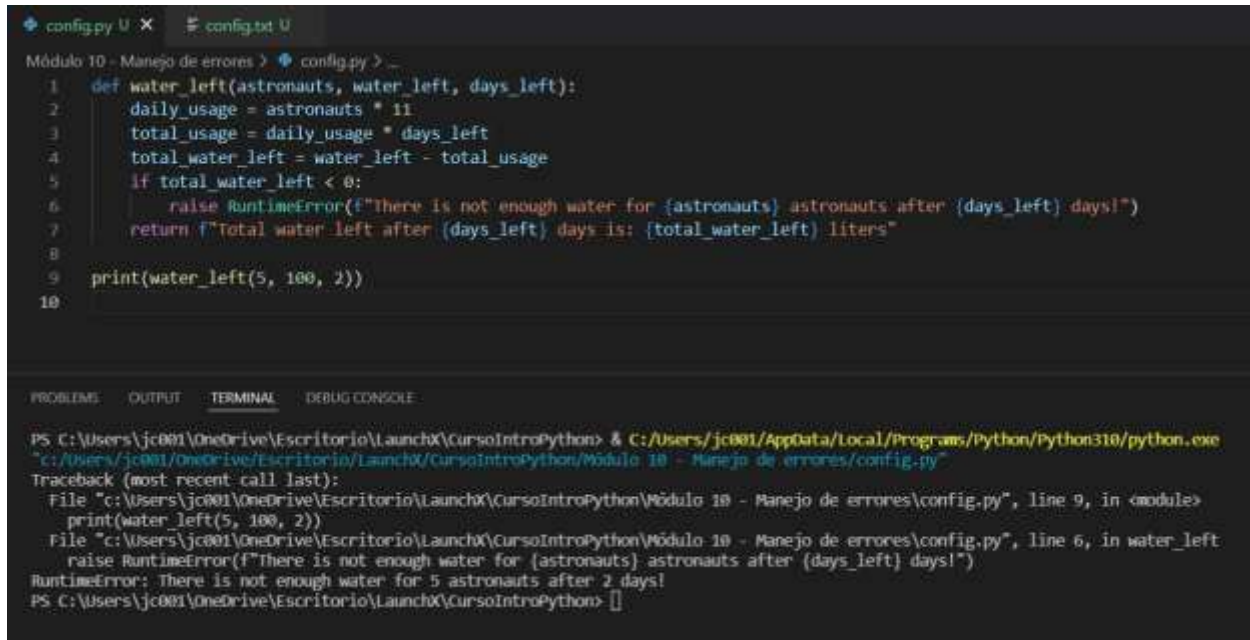
Probemos con cinco astronautas, 100 litros de agua sobrante y dos días:



```
config.py x config.txt U
Módulo 10 - Manejo de errores > config.py > _
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     return f"Total water left after {days_left} days is: {total_water_left} liters"
6
7 print(water_left(5, 100, 2))
8

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> & C:/Users/jc001/AppData/Local/Programs/Python/Python310/python.exe
"C:/Users/jc001/OneDrive/Escritorio/LaunchX/CursoIntroPython/Módulo 10 - Manejo de errores/config.py"
Total water left after 2 days is: -10 liters
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython>
```

Esto no es muy útil, ya que una carencia en los litros sería un error. Después, el sistema de navegación podría alertar a los astronautas que no habrá suficiente agua para todos en dos días. Si eres un ingeniero(a) que programa el sistema de navegación, podrías generar una excepción en la función `water_left()` para alertar de la condición de error:



```
config.py x config.txt U
Módulo 10 - Manejo de errores > config.py > ...
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!")
7     return f"Total water left after {days_left} days is: {total_water_left} liters"
8
9 print(water_left(5, 100, 2))
10

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> & C:/Users/jc001/AppData/Local/Programs/Python/Python310/python.exe
"c:/Users/jc001/OneDrive/Escritorio/LaunchX/CursoIntroPython/Módulo 10 - Manejo de errores/config.py"
Traceback (most recent call last):
  File "c:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython\Módulo 10 - Manejo de errores\config.py", line 9, in <module>
    print(water_left(5, 100, 2))
  File "c:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython\Módulo 10 - Manejo de errores\config.py", line 6, in water_left
    raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
RuntimeError: There is not enough water for 5 astronauts after 2 days!
PS C:\Users\jc001\OneDrive\Escritorio\LaunchX\CursoIntroPython> []
```