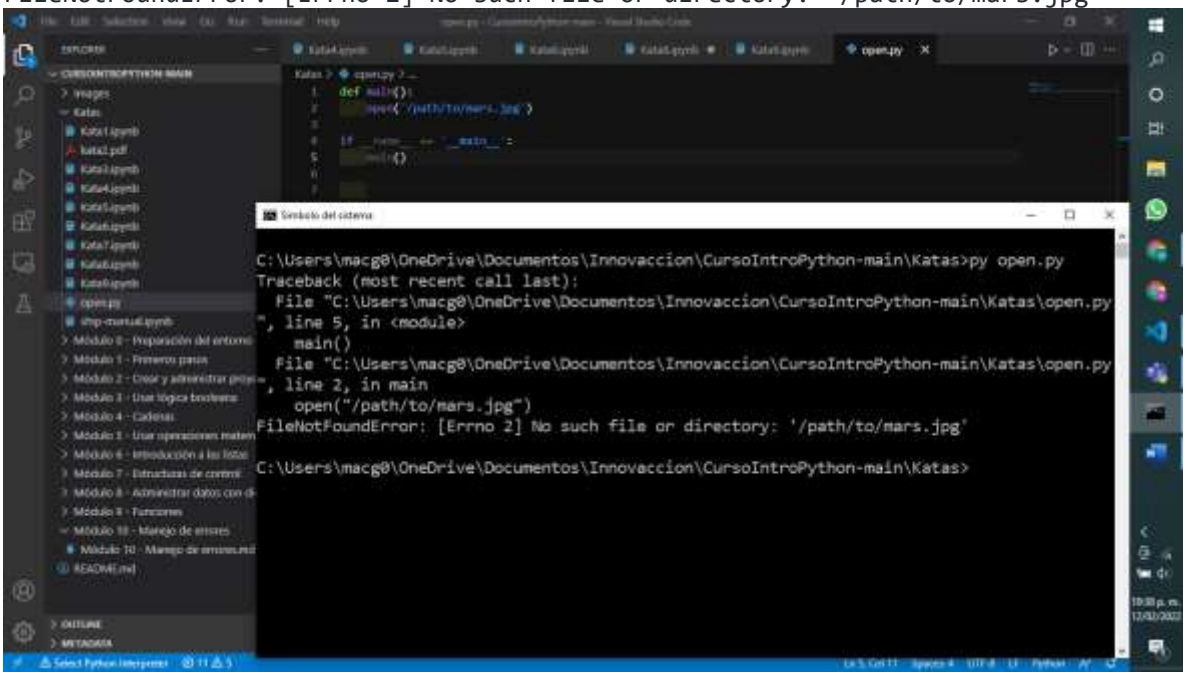
Intenta crear un archivo de Python y asígnale el nombre *open.py*, con el contenido siguiente:

```
def main():
    open("/path/to/mars.jpg")

if __name__ == '__main__':
    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:

```
$ python3 open.py
Traceback (most recent call last):
  File "/tmp/open.py", line 5, in <module>
    main()
  File "/tmp/open.py", line 2, in main
    open("/path/to/mars.jpg")
FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
```



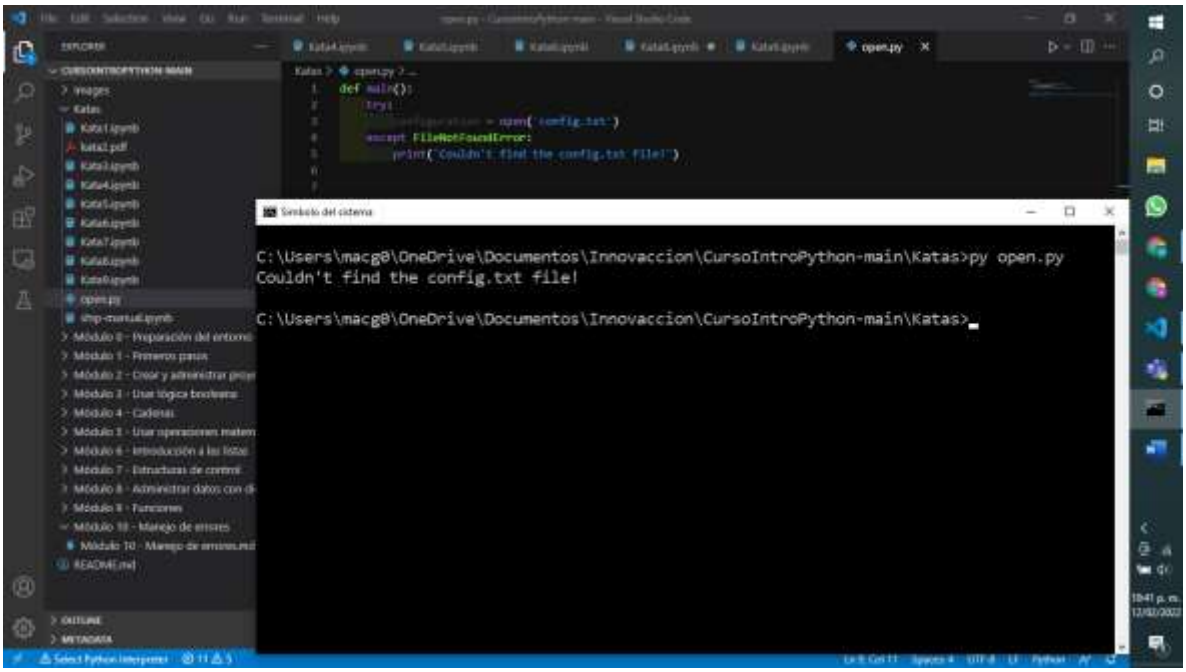
Try y Except de los bloques

Vamos a usar el ejemplo de navegador a fin de crear código que abra archivos de configuración para la misión de Marte. Los archivos de configuración pueden tener todo tipo de problemas, por lo que es fundamental notificarlos con precisión cuando se presenten. Sabemos que, si no existe un archivo o directorio, se genera `FileNotFoundError`. Si queremos controlar esa excepción, podemos hacerlo con un bloque try y except:

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

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.

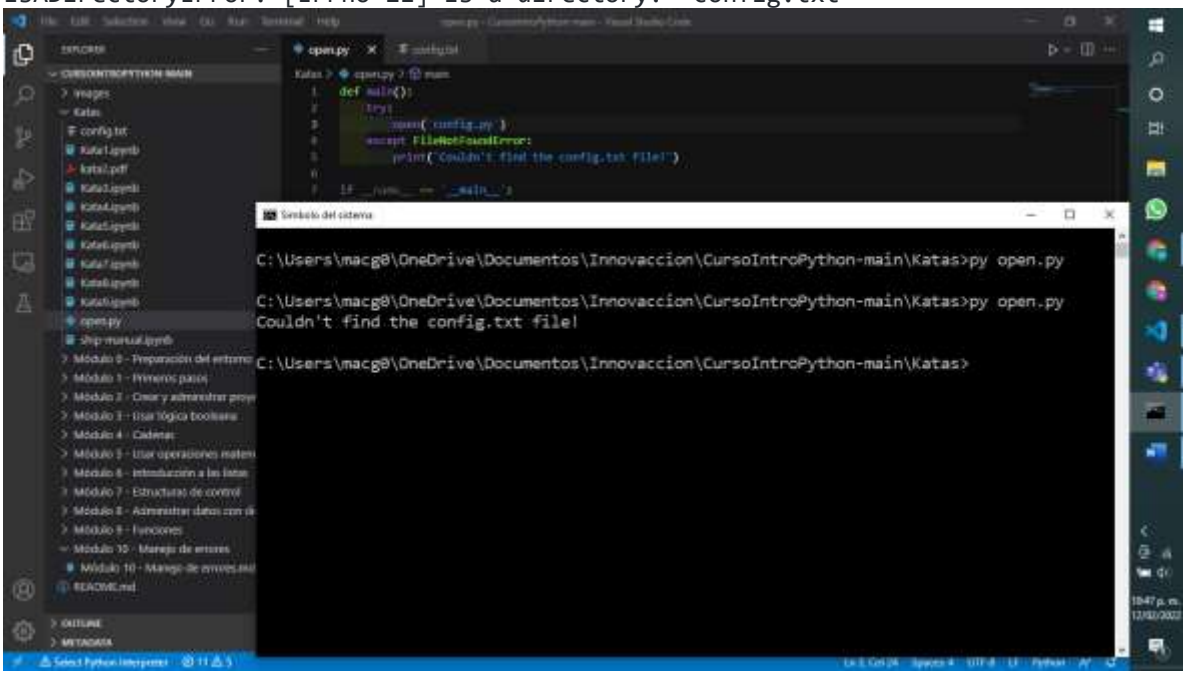
Aunque es común un archivo que no existe, no es el único error que podemos encontrar. Los permisos de archivo no válidos pueden impedir la lectura de un archivo, incluso si este existe. Vamos a crear un archivo de Python denominado config.py. El archivo tiene código que busca y lee el archivo de configuración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



A continuación, quita,ps el archivo *config.txt* y creamos un directorio denominado *config.txt*. Intentaremos llamar al archivo *config.py* para ver un error nuevo que debería ser similar al siguiente:

```
$ python config.py
Traceback (most recent call last):
  File "/tmp/config.py", line 9, in <module>
    main()
  File "/tmp/config.py", line 3, in main
```

```
    configuration = open('config.txt')
IsADirectoryError: [Errno 21] Is a directory: 'config.txt'
```



Una manera poco útil de controlar este error sería detectar todas las excepciones posibles para evitar un traceback. Para comprender por qué detectar todas las excepciones es problemático, probaremos actualizando la función `main()`:

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