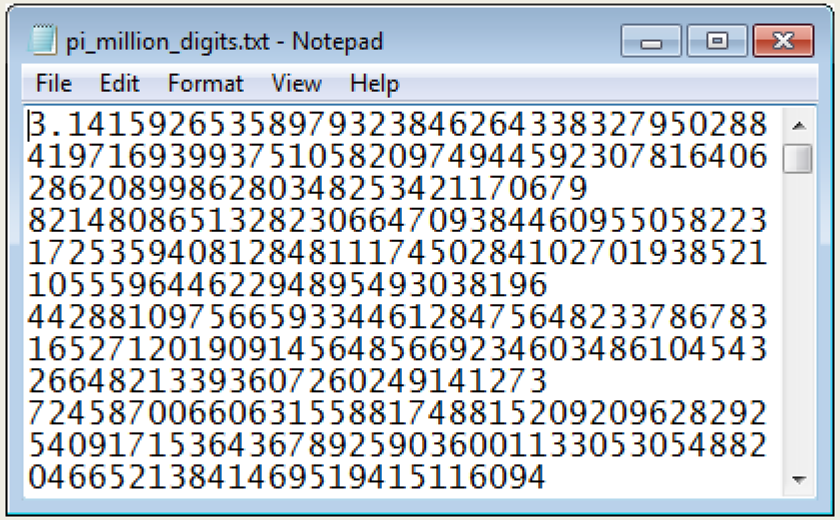
FILES AND EXCEPTS

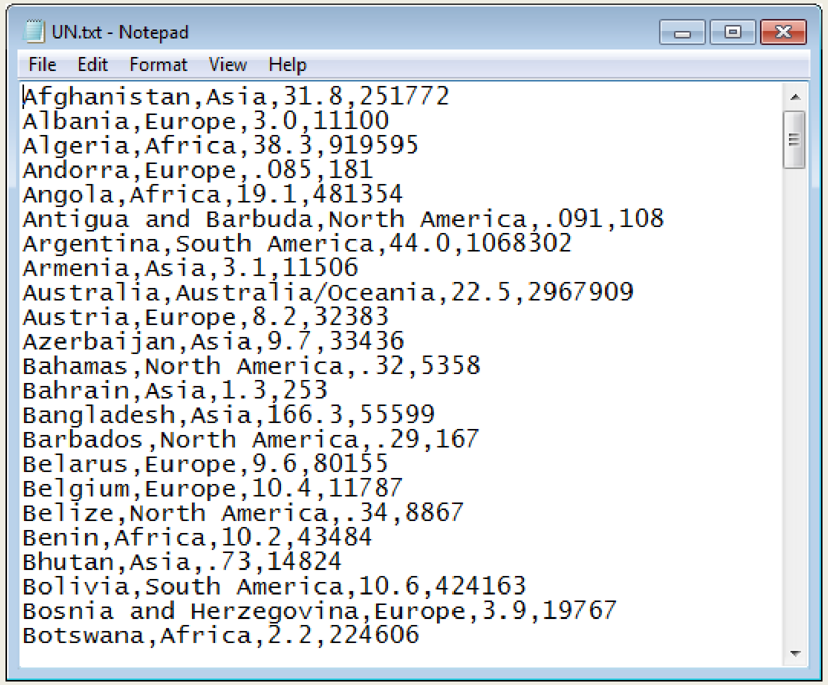
Processing Data

Files (1 of 3)

- Values used in a Python program reside in memory and are lost when the program terminates.
- If a program writes the values to a file on a storage device, any Python program can access the values at a later time.
- Files create long-term storage of data.



```
3.141592653589793238462643383279502884
4197169399375105820974944592307816406
28620899862803482534211706798
2148086513282306647093844609550582231
72535940812848111745028410270193852110
555964462294895493038196
4428810975665933446128475648233786783
16527120190914564856692346034861045432
66482133936072602491412737
24587006606315588174881520920962829254
091715364367892590360011330530548820
46652138414695194151160943
```



```
Afghanistan,Asia,31.8,251772
Albania,Europe,3.0,11100
Algeria,Africa,38.3,919595
Andorra,Europe,.085,181
Angola,Africa,19.1,481354
Antigua and Barbuda,North America,.091,108
Argentina,South America,44.0,1068302
Armenia,Asia,3.1,11506
Australia,Australia/Oceania,22.5,2967909
Austria,Europe,8.2,32383
Azerbaijan,Asia,9.7,33436
Bahamas,North America,.32,5358
Bahrain,Asia,1.3,253
Bangladesh,Asia,166.3,55599
Barbados,North America,.29,167
Belarus,Europe,9.6,80155
Belgium,Europe,10.4,11787
Belize,North America,.34,8867
Benin,Africa,10.2,43484
Bhutan,Asia,.73,14824
Bolivia,South America,10.6,424163
Bosnia and Herzegovina,Europe,3.9,19767
Botswana,Africa,2.2,224606
```

Files (2 of 3)

Text Files

- Text file is a simple file consisting of lines of text with no formatting.
- Text file can be created with any word processor. *Notepad*(on a PC) or *TextEdit* (on a Mac).
- Usually text files have the extension ***txt***.
- Python program can access the values

CSV Files

- CSV-formatted file has several items of data on each line with the items separated by commas.
- CSV stands for **Comma-Separated Values.**
- Each line of this text file is called a ***record*** and each record contains four ***fields***, a name field, a continent file, a population field, and an area field in the file, UN.txt.

Files (3 of 3)

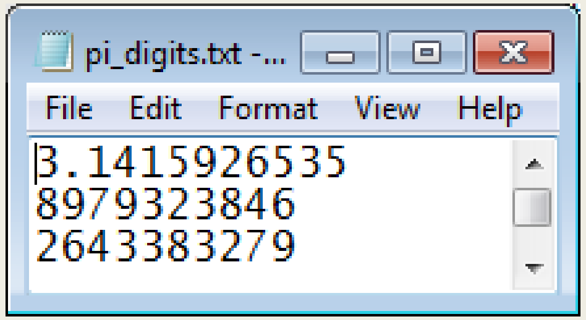
- File Methods
  https://www.w3schools.com/python/python_ref_file.asp

- File and Directory Access
  https://docs.python.org/3/library/filesys.html

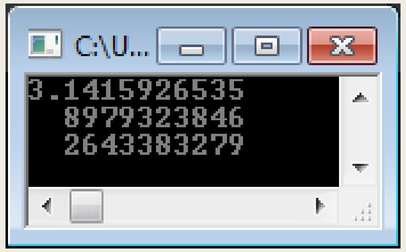
Reading from a File

Ex1a_file_reader.py & Ex1b_file_reader.py & Ex1c_file_reader.py

- Establish connection between the program and the file
  - *File is said to be **opened** for input.*
  - *The keyword **`with`** closes the file once access to it is no longer needed.*
  - *All you have to do is **open** the file and work with it as desired, trusting that Python will **close** it automatically when the **`with`** block finishes execution.*
  - *Use **`rstrip`**`()` method returns a right trim version of the string/strip newline characters.*



```
3.1415926535
8979323846
2643383279
```



```
3.1415926535
  8979323846
  2643383279
```

Making a List

Ex3_pi_birthday.py

- Using ***`with`***, the file object returned by ***`open`***`()` is only available inside the ***`with`*** block that contains it.
- Store the file’s lines in a list inside the block and then work with that list later.
- ***`readlines()`*** method returns a **list** of lines from the file.

```
>>> filename = 'digits.txt’
    with open(filename) as file_object:
        lines = file_object.readlines()
    for line in lines:
        print(line.rstrip())
```

Open() and *close()*

Ex2a_open_close.py & Ex2b_file_reader.py & Ex2c_file_reader.py

- **Reading** is the default mode for opening a file.
- **Close** the file when you are done with it.

Reading Line by Line

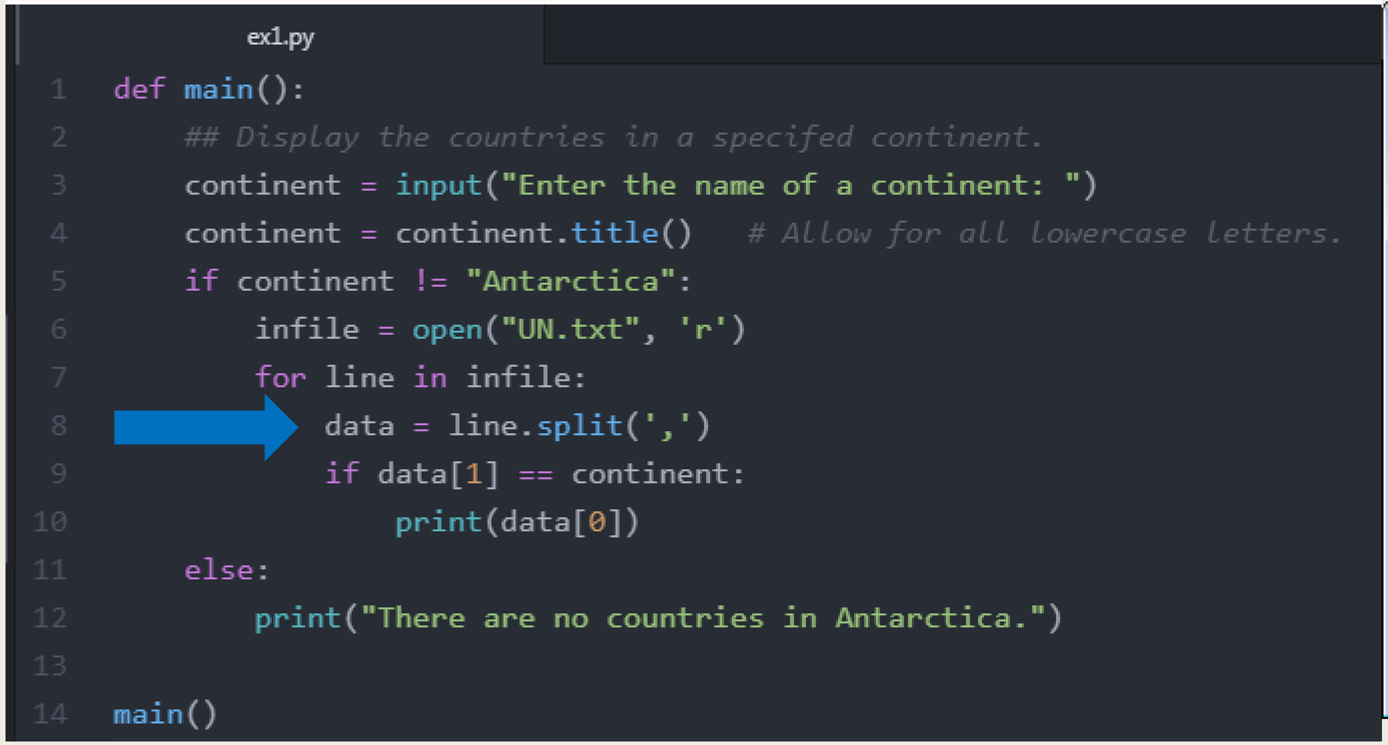
Ex4a_reading_line_by_line.py

- Use a ***for*** loop on the file object to examine each line from a file one at a time

```
>>> filename = 'pi_digits.txt’
    with open(filename) as file_object:
        for line in file_object:
            print(line.rstrip())
```

Accessing the Data in a CSV File

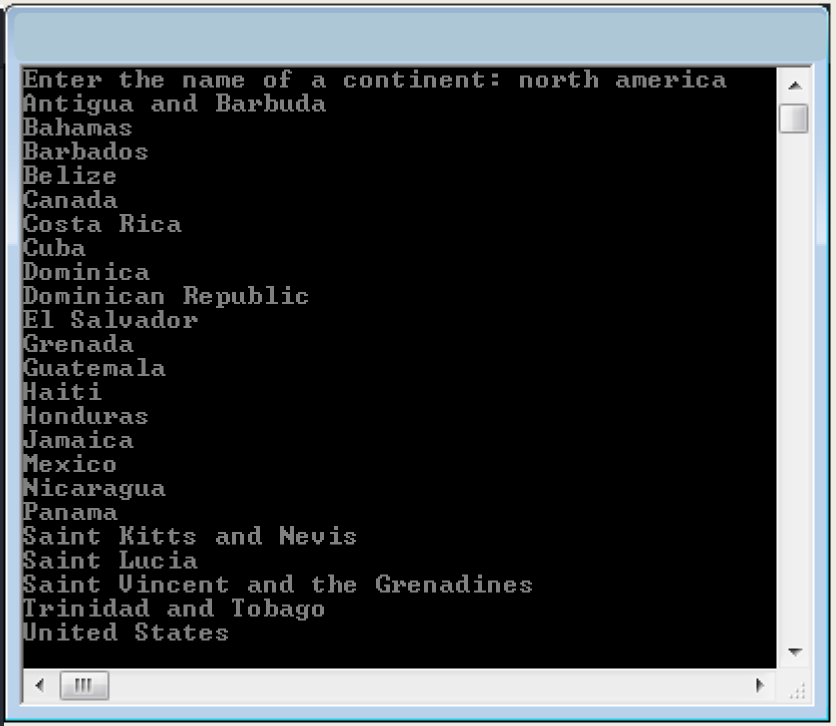
Ex4b_access_data.py



ex1.py

```python
def main():
    ## Display the countries in a specifed continent.
    continent = input("Enter the name of a continent: ")
    continent = continent.title()   # Allow for all lowercase letters.
    if continent != "Antarctica":
        infile = open("UN.txt", 'r')
        for line in infile:
            data = line.split(',')
            if data[1] == continent:
                print(data[0])
    else:
        print("There are no countries in Antarctica.")

main()
```



```
Enter the name of a continent: north america
Antigua and Barbuda
Bahamas
Barbados
Belize
Canada
Costa Rica
Cuba
Dominica
Dominican Republic
El Salvador
Grenada
Guatemala
Haiti
Honduras
Jamaica
Mexico
Nicaragua
Panama
Saint Kitts and Nevis
Saint Lucia
Saint Vincent and the Grenadines
Trinidad and Tobago
United States
```

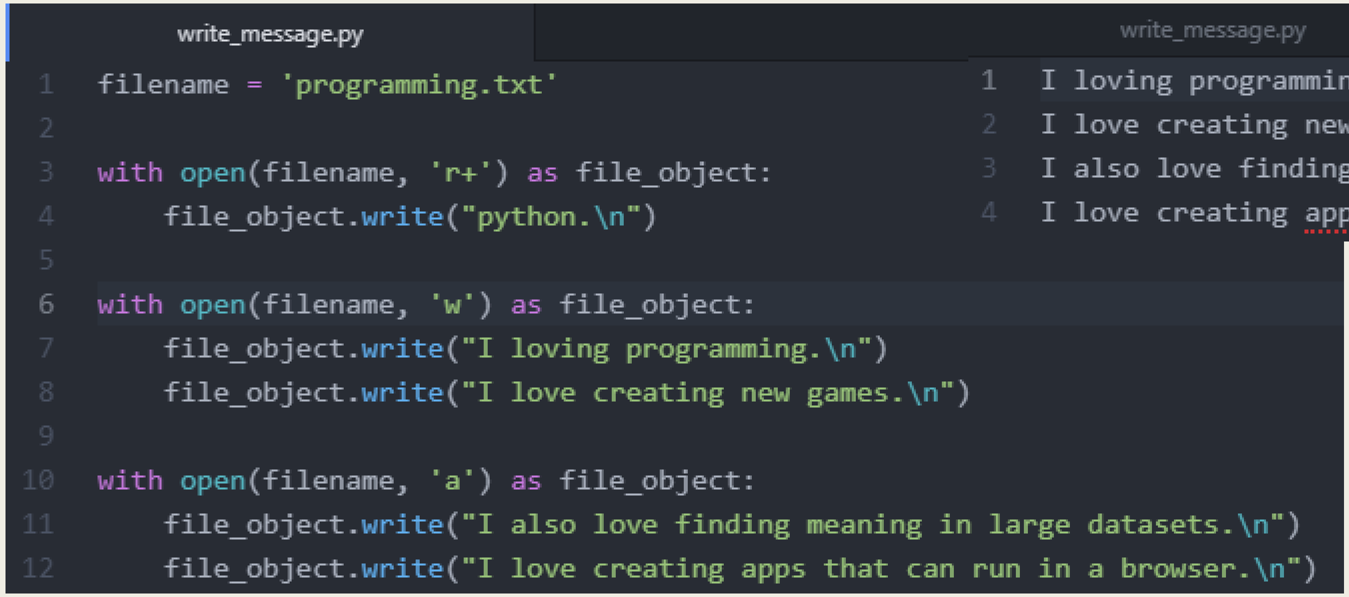
Open()

- The key function for working with files in Python is the `open()` function.
- The `open()` function takes two parameters; **filename**, and **mode.**
- There are four different methods (modes) for opening a file:
  "r" - Read - Default value. Opens a file for reading, error if the file does not exist
  "a" - Append - Opens a file for appending, creates the file if it does not exist
  "w" - Write - Opens a file for writing, creates the file if it does not exist
  "x" - Create - Creates the specified file, returns an error if the file exists
- In addition you can specify if the file should be handled as binary or text mode:
  "t" - Text - Default value. Text mode
  "b" - Binary - Binary mode (e.g. images)

Writing to a File

Ex5_write_message.py

- Use `open(filename, 'w')`, the second argument `'w'` tells Python to open the file in `write mode`.
- `open(filename, 'a'), 'a' is append mode`
- `open(filename, 'r+'), 'r+' is read and write mode`



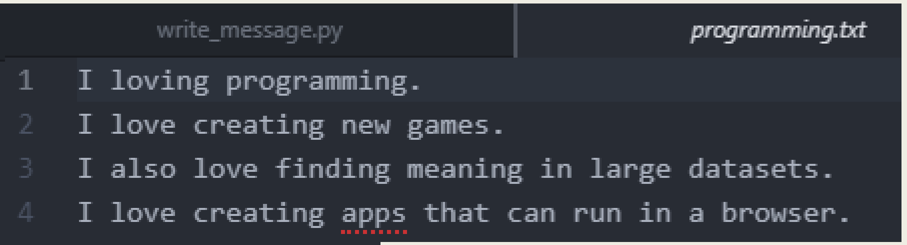
write_message.py

```python
filename = 'programming.txt'

with open(filename, 'r+') as file_object:
    file_object.write("python.\n")

with open(filename, 'w') as file_object:
    file_object.write("I loving programming.\n")
    file_object.write("I love creating new games.\n")

with open(filename, 'a') as file_object:
    file_object.write("I also love finding meaning in large datasets.\n")
    file_object.write("I love creating apps that can run in a browser.\n")
```



write_message.py | programming.txt

```
I loving programming.
I love creating new games.
I also love finding meaning in large datasets.
I love creating apps that can run in a browser.
```

File Paths

- When pass a simple file to the `open()` function, Python looks in the directory where the file that's currently being executed is stored.

- Use relative file path for a given location relative to the directory where the currently running the program file is stored.
```
fileName = ‘text_files/filename.txt’
```

- Use absolute path for any location on the system.
```
fileName = ‘/home/data/text_files/filename.txt’
fileName = ‘C:\\myData\\text_files\\filename.txt’
```

OS Path Module

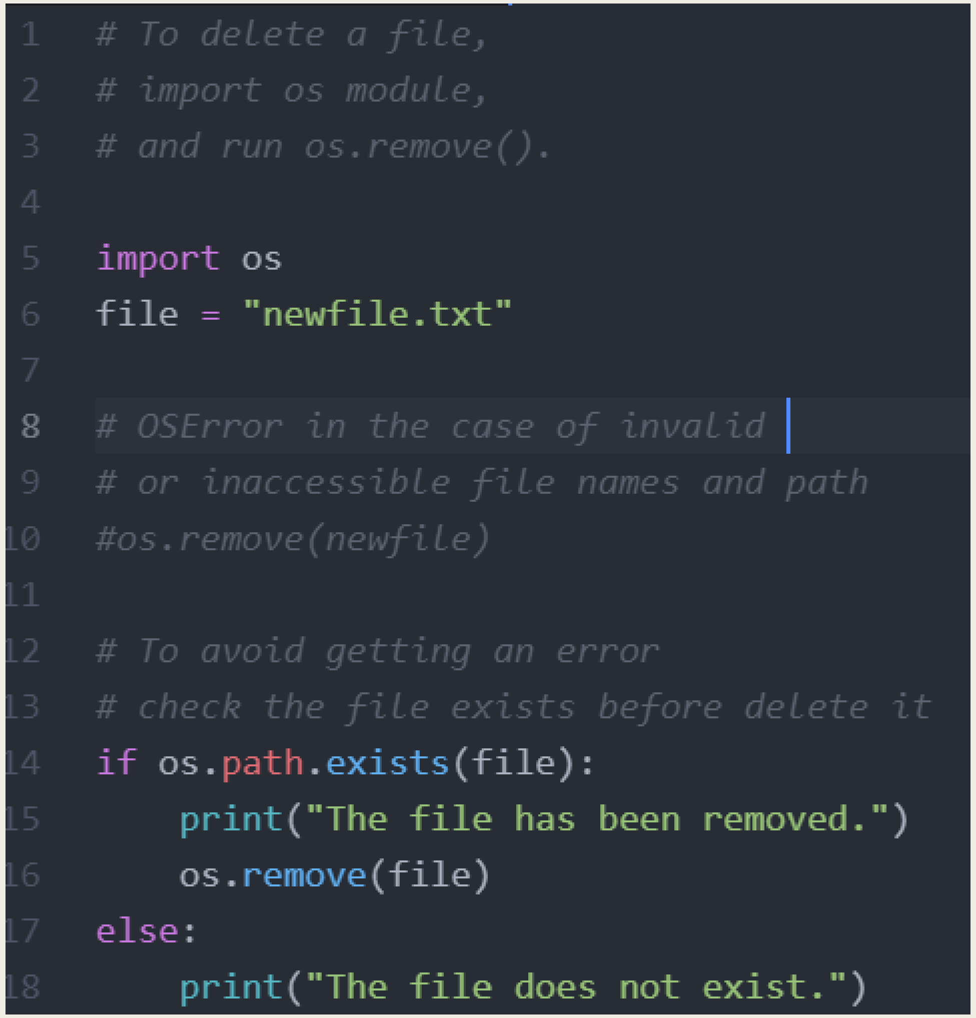
Ex6_fileame_path.py

- This module contains some useful functions on pathnames.
- These functions are used for different purposes such as for merging, normalizing and retrieving path names in python .
- All of these functions accept either only bytes or only string objects as their parameters.
- The result is an object of the same type, if a path or file name is returned.
- As there are different versions of **operating system** so there are several versions of this module in the standard library.
- https://docs.python.org/3/library/os.path.html

Delete File

Ex6a_file_delete.py & Ex6b_file_delete.py & Ex6c_remove_folder.py

- import the `os` module
- Use `os.remove()method`



```python
# To delete a file,
# import os module,
# and run os.remove().

import os
file = "newfile.txt"

# OSError in the case of invalid
# or inaccessible file names and path
#os.remove(newfile)

# To avoid getting an error
# check the file exists before delete it
if os.path.exists(file):
    print("The file has been removed.")
    os.remove(file)
else:
    print("The file does not exist.")
```

Exceptions (1 of 5)

- Exceptions occur due to circumstances beyond programmer's control
  - *Invalid data are input*
  - *File cannot be accessed*
- Even though might be user’s fault
  - *Programmer must anticipate*
  - *Include code to work around the occurrence*

Exceptions (2 of 5)

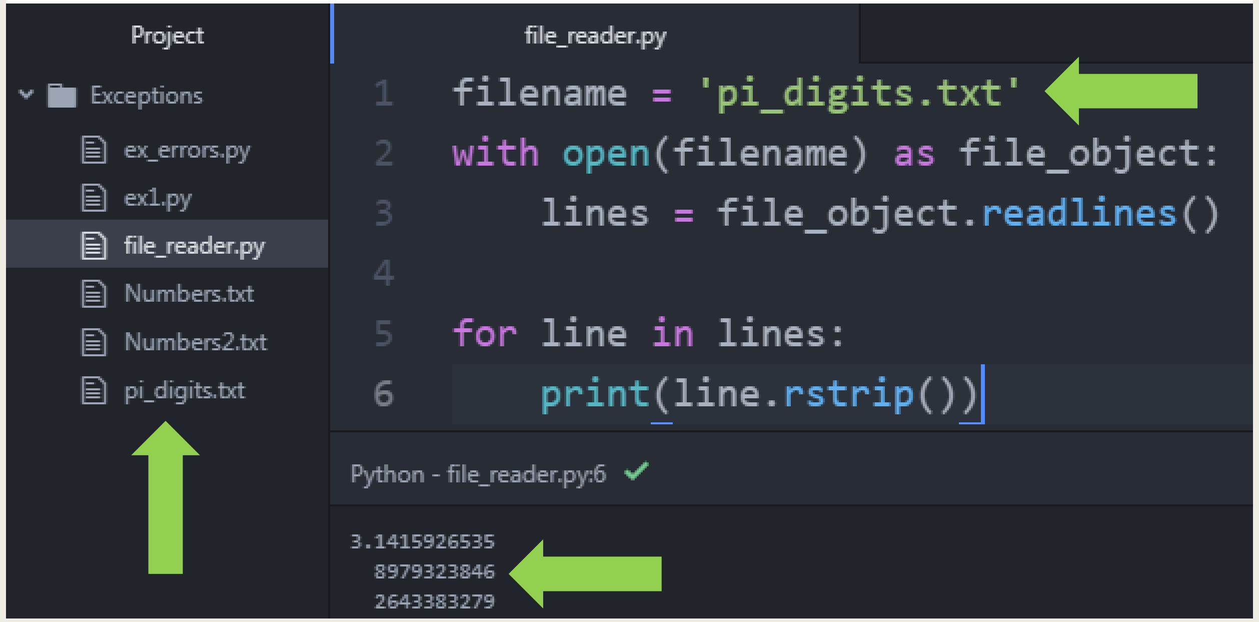
Ex7a_catch_exception.py

Exception name	**Description and example**
ImportError	Import statement fails to find requested module. import nonexistentModule
FileNotFoundError	Requested file doesn’t exist or is not located where expected. open("NonexistentFile.txt", 'r')
AttributeError	An unavailable functionality (usually a method) is requested for an object. print(x.upper()) #where x=123
IndexError	An index is out of range. letter = "abc"[7]

Exceptions (3 of 5)

Exception name	Description and example
KeyError	KeyError No such key in dictionary. word = d['c'] # where d = {'a':"apple", 'b':"banana"}
NameError	The value of a variable cannot be found. term = word # where word was never created
TypeError	Function or operator receives the wrong type of argument. x = len(23) or x = 6 / '2' or x = 9 + 'W' or x = abs(-3,4)
ValueError	Function or operator receives right type of argument, but inappropriate value. x = int('a') or L.remove(item) #where item is not in list
ZeroDivisionError	The second number in a division or modulus operation is 0. num = 1 / 0 or num = 23 % 0

Exceptions (4 of 5)

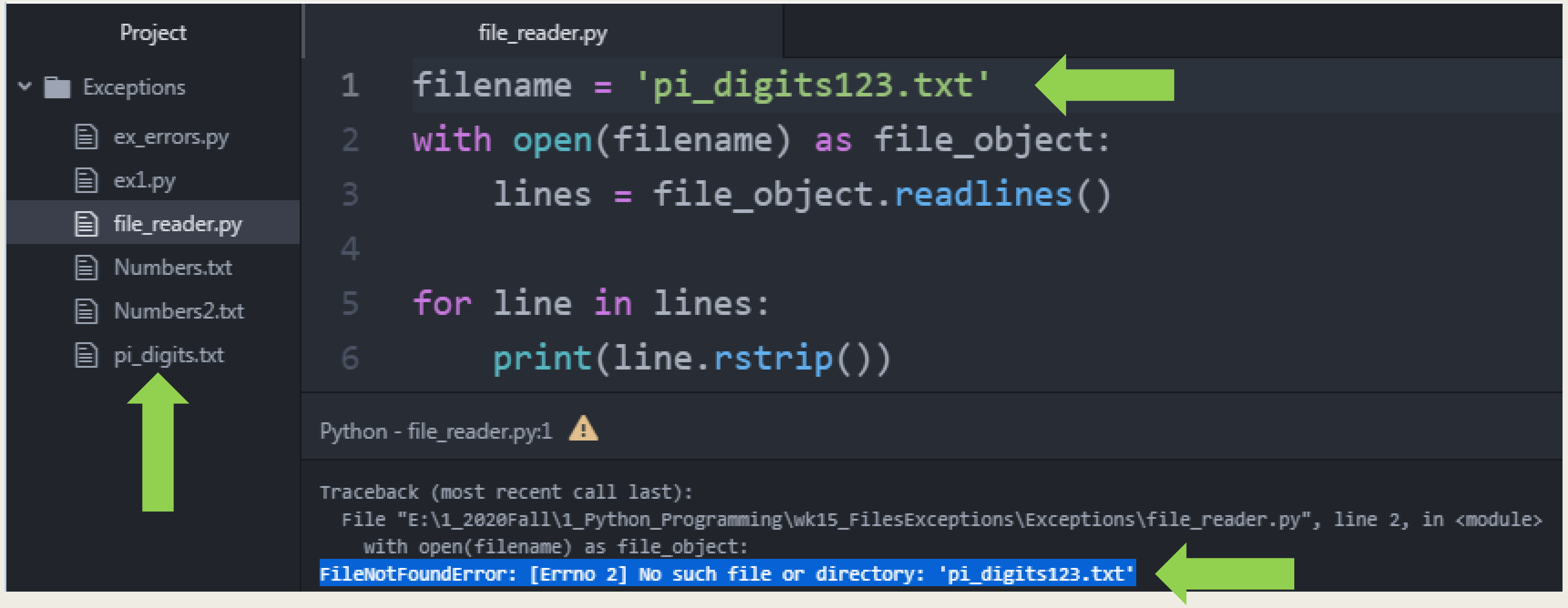


```python
filename = 'pi_digits.txt'
with open(filename) as file_object:
    lines = file_object.readlines()

for line in lines:
    print(line.rstrip())
```

```
3.1415926535
  8979323846
  2643383279
```

Exceptions (5 of 5)



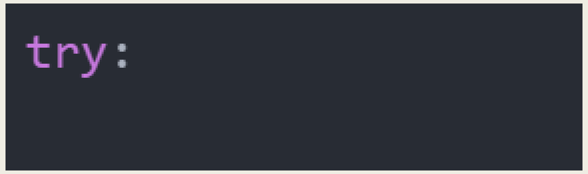
```python
filename = 'pi_digits123.txt'
with open(filename) as file_object:
    lines = file_object.readlines()

for line in lines:
    print(line.rstrip())
```

```
Traceback (most recent call last):
  File "E:\1_2020Fall\1_Python_Programming\wk15_FilesExceptions\Exceptions\file_reader.py", line 2, in <module>
    with open(filename) as file_object:
FileNotFoundError: [Errno 2] No such file or directory: 'pi_digits123.txt'
```

The *try* block

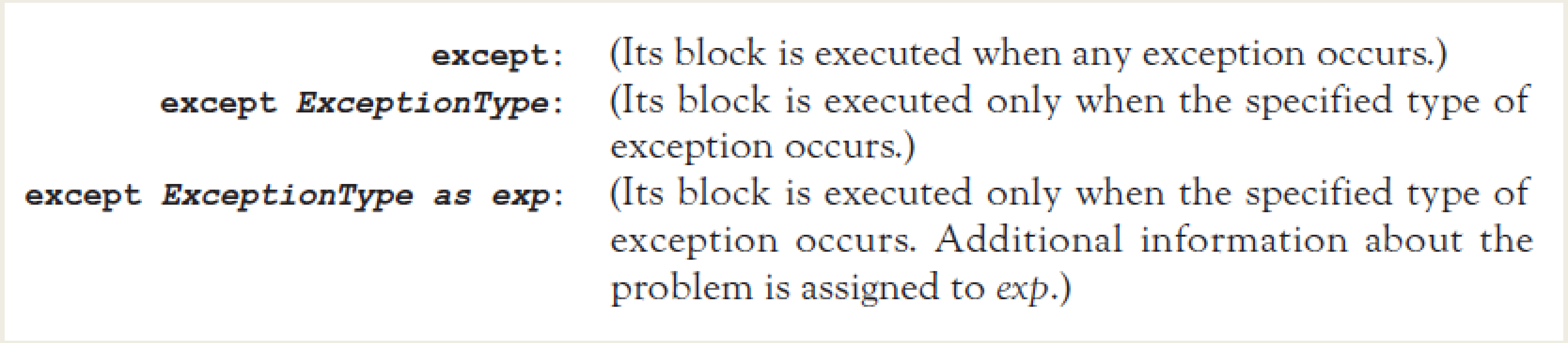
- A program is said to be **robust** if it performs well under atypical situations.
- **Robust** program explicitly handles previous exception
  - *Protecting the code with a* ***`try`*** *statement.*



```
try:
```

The *except* block

- Three types of except clauses:



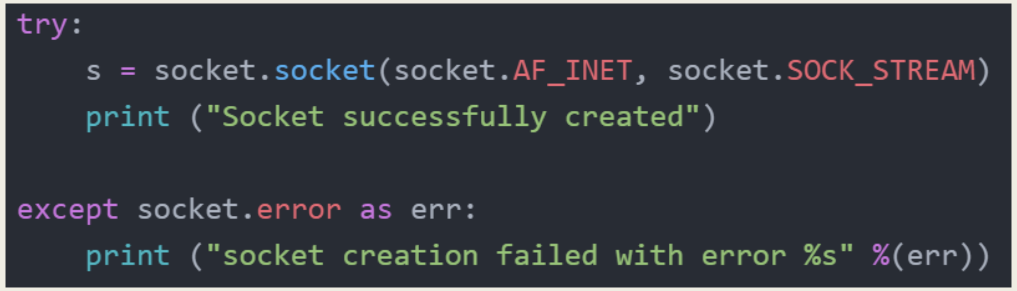
`except:`	(Its block is executed when any exception occurs.)
`except ExceptionType:`	(Its block is executed only when the specified type of exception occurs.)
`except ExceptionType as exp:`	(Its block is executed only when the specified type of exception occurs. Additional information about the problem is assigned to *exp*.)

The *else* and *finally* clauses

- **try** statement also can include a single **else** clause
  - *Follows the* **except** *clauses*
  - *Executed when no exceptions occur*
- **try** statement can end with a **finally** clause
  - *Usually used to clean up resources such as files that were left open*
- **try** statement must contain either an **except** clause or a **finally** clause.
  - *Usually used to clean up resources such as files that were left open*

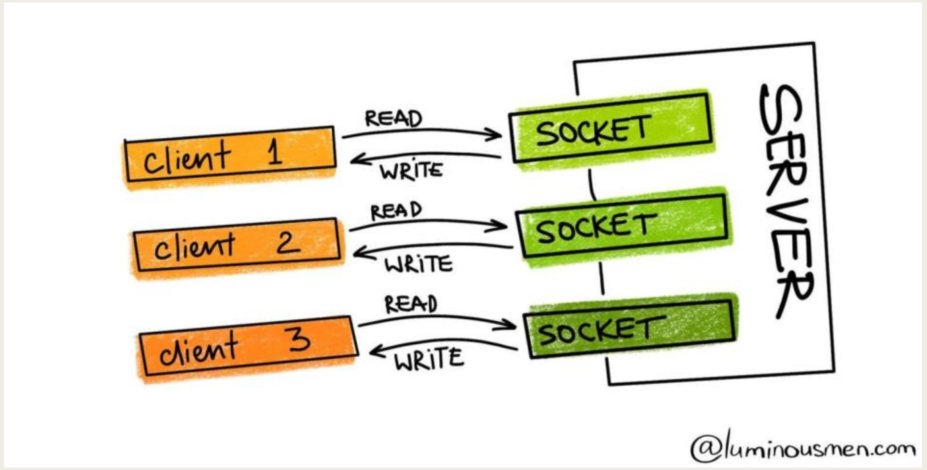
An Example of Exception Handling: Socket Programming
Ex7b_socket_programming.py

- Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection.
- The server forms the listener socket while the client reaches out to the server.
- They are the real backbones behind web browsing.
- In simpler terms, there is a server and a client.



```python
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print ("Socket successfully created")

except socket.error as err:
    print ("socket creation failed with error %s" %(err))
```



Summary (1 of 3)

- The functions `open(filename, ‘r’)`,`open(filename, ‘w’)`, `open(filename, ‘a’)` create file objects connected to the named text file.
- These objects are used for **reading** content from the file, **writing** content to the file, and **adding** content to the file respectively.
- After a file is opened for writing or appending, a statement of the form `write(str)` writes `str` to the file via a buffer.
- The `close()` method makes sure that all data still in the buffer is written to the file and then terminates the connection.

Summary (2 of 3)

- **CSV files** store tabular data with each line containing the same number of fields, where the fields are separated by commas.
- The *`split`* method is needed to extract information from a CSV file.
- The data from a CSV file can be placed into an **Excel spreadsheet** and analyzed with Excel; and data from an Excel spreadsheet can be transferred to a CSV file and analyzed with Python.
- After the `os` module has been imported, a closed file can be renamed with a statement of the form ***`os.rename(oldFileName, newFileName)`***, deleted with a statement of the form ***`os.remove(filename)`***, and have its existence verified with a Boolean function of the form ***`os.path.exists(filename)`***.

Summary (3 of 3)

- The words `try`, `except`, `else`, and `finally` are **reserved** words and therefore are colorized.
- The `try` statement is one of the primary tools for creating **robust** programs.
- A single except clause may refer to several types of errors. If so, the error names are listed in a tuple.
- **Refactoring** is the process of restructuring existing code without changing its external behavior. Refactoring is intended to improve the design, structure, and/or implementation of the program, while preserving its functionality.
- The advantages of refactoring: improved code readability and reduced complexity; these can improve the source code's maintainability and create a simpler, cleaner, or more expressive internal architecture or object model to improve extensibility.
- The goal for refactoring is improved performance; software engineers face an ongoing challenge to write programs that perform faster or use less memory.

Files and Exceptions Terminologies

- ❑ 1 File class
- ❑ 2 File Object
- ❑ 3 File Read
- ❑ 4 File Write
- ❑ 5 File Append
- ❑ 6 File Create
- ❑ 7 open()
- ❑ 8 close()
- ❑ 9 with
- ❑ 10 Text file(txt)
- ❑ 11 Comma Separated Values (CSV)
- ❑ 12 Excel spreadsheet
- ❑ 13 split()
- ❑ 14 os.path.exists()
- ❑ 15 os.remove()
- ❑ 16 os.getcwd()
- ❑ 17 import os
- ❑ 18 Import csv
- ❑ 19 Catch
- ❑ 20 Exception Handling
- ❑ 21 try
- ❑ 22 except
- ❑ 23 as
- ❑ 24 else
- ❑ 25 finally
- ❑ 26 Robust
- ❑ 27 Runtime errors
- ❑ 28 Socket Programming
- ❑ 29 Refactoring