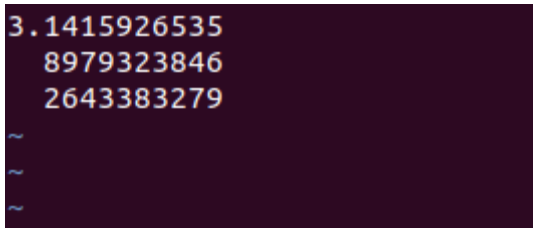
Files and Exceptions

Reading from a File

Reading an Entire File

- `pi_digits.txt`



```
with open('pi_digits.txt') as file_object:
    contents = file_object.read()
print(contents)
~
~
~
```

- 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.

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 file_reader_1.py
3.1415926535
  8979323846
  2643383279

(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

- The only difference between this output and the original file is the extra blank line at the end of the output.
- The blank line appears because `read()` returns an empty string when it reaches the end of the file; this empty string shows up as a blank line.
- If you want to remove the extra blank line, you can use `rstrip()` in the call to `print()`.

```
with open('pi_digits.txt') as file_object:
    contents = file_object.read()
print(contents.rstrip())
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 file_reader_2.py
3.1415926535
  8979323846
  2643383279
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

File Paths

- When you pass a simple filename like *pi_digits.txt* to the `open()` function, Python looks in the directory where the file that's currently being executed (that is, your *.py* program file) is stored.

- Sometimes, depending on how you organize your work, the file you want to open won’t be in the same directory as your program file.

- A relative file path tells Python to look for a given location relative to the directory where the currently running program file is stored.
  - `with open('text_files/filename.txt') as file_object:`

- Absolute paths are usually longer than relative paths, so it’s helpful to assign them to a variable and then pass that variable to `open()`:
  - ```
    file_path =
    '/home/ehmatthes/other_files/text_files/filename.txt'
    with open(file_path) as file_object:
    ```
    ```

Reading Line by Line

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

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 file_reader_3.py
3.1415926535
  8979323846
  2643383279
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Making a List of Lines from a File

- The `readlines()` method takes each line from the file and stores it in a list.

```
filename = 'pi_digits.txt'

with open(filename) as file_object:
    lines = file_object.readlines()

print(lines)

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

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 file_reader_4.py
['3.1415926535 \n', '  8979323846 \n', '  2643383279\n']
3.1415926535
  8979323846
  2643383279
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Working with a File's Contents

```
filename = 'pi_digits.txt'

with open(filename) as file_object:
    lines = file_object.readlines()

pi_string = ''
for line in lines:
    pi_string += line.strip()

print(pi_string)
print(len(pi_string))
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 pi_string_1.py
3.14159265358979323846264338327 9
32
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Large Files: One Million Digits

```
filename = 'pi_million_digits.txt'

with open(filename) as file_object:
    lines = file_object.readlines()

pi_string = ''
for line in lines:
    pi_string += line.strip()

print(f"{pi_string[:52]}...")
print(len(pi_string))
~
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 pi_string_2.py
3.14159265358979323846264338327950288419716939937510...
1000002
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Is Your Birthday Contained in Pi?

```
filename = 'pi_million_digits.txt'

with open(filename) as file_object:
    lines = file_object.readlines()

pi_string = ''
for line in lines:
    pi_string += line.strip()

birthday = input("Enter your birthday, in the form mmddyy: ")
if birthday in pi_string:
    print("Your birthday appears in the first million digits of pi!")
else:
    print("Your birthday does not appear in the first million digits of pi.")
~
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 pi_birthday.py
Enter your birthday, in the form mmddyy: 111219
Your birthday does not appear in the first million digits of pi.
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Writing to a File

Writing to an Empty File

```
filename = 'programming.txt'

with open(filename, 'w') as file_object:
    file_object.write("I love programming.")
~
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 write_message_1.py
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ cat pr
ogramming.txt
I love programming.(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9
/chapter_10$
```

Writing Multiple Lines

```
filename = 'programming.txt'

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

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 write_message_2.py
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ cat pr
ogramming.txt
I love programming.
I love creating new games.
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Appending to a File

```
filename = 'programming.txt'

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")
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 write_message.py
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ cat pr
ogramming.txt
I love programming.
I love creating new games.
I also love finding meaning in large datasets.
I love creating apps that can run in a browser.
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Exceptions

- Python uses special objects called *exceptions* to manage errors that arise during a program's execution.
- Whenever an error occurs that makes Python unsure what to do next, it creates an exception object.
- If you write code that handles the exception, the program will continue running.
- If you don't handle the exception, the program will halt and show a *traceback*, which includes a report of the exception that was raised.

- Exceptions are handled with `try-except` blocks.
- A `try-except` block asks Python to do something, but it also tells Python what to do if an exception is raised.
- When you use `try-except` blocks, your programs will continue running even if things start to go wrong.
- Instead of tracebacks, which can be confusing for users to read, users will see friendly error messages that you write.

Handling the `ZeroDivisionError` Exception

```
>>> print(5/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>>
```

Using `try-except` Blocks

```
>>> try:
...     print(5/0)
... except ZeroDivisionError:
...     print("You can't divide by zero!")
...
You can't divide by zero!
>>>
```

Using Exceptions to Prevent Crashes

```
print("Give me two numbers, and I'll divide them.")
print("Enter 'q' to quit.")

while True:
    first_number = input("\nFirst number: ")
    if first_number == 'q':
        break
    second_number = input("Second number: ")
    if second_number == 'q':
        break
    try:
        answer = int(first_number) / int(second_number)
    except ZeroDivisionError:
        print("You can't divide by 0!")
    else:
        print(answer)
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ vi div
ision_calculator.py
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 division_calculator.py
Give me two numbers, and I'll divide them.
Enter 'q' to quit.

First number: 5
Second number: 0
You can't divide by 0!

First number: 5
Second number: 2
2.5

First number: q
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Handling the `FileNotFoundError` Exception

```
filename = 'alice.txt'

try:
    with open(filename, encoding='utf-8') as f:
        contents = f.read()
except FileNotFoundError:
    print(f"Sorry, the file {filename} does not exist.")
else:
    # Count the approximate number of words in the file.
    words = contents.split()
    num_words = len(words)
    print(f"The file {filename} has about {num_words} words.")
~
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 alice.py
The file alice.txt has about 29465 words.
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Working with Multiple Files

```
def count_words(filename):
    """Count the approximate number of words in a file."""
    try:
        with open(filename, encoding='utf-8') as f:
            contents = f.read()
    except FileNotFoundError:
        pass
    else:
        words = contents.split()
        num_words = len(words)
        print(f"The file {filename} has about {num_words} words.")

filenames = ['alice.txt', 'siddhartha.txt', 'moby_dick.txt', 'little_women.txt']
for filename in filenames:
    count_words(filename)
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 word_count.py
The file alice.txt has about 29465 words.
The file siddhartha.txt has about 42172 words.
The file moby_dick.txt has about 215830 words.
The file little_women.txt has about 189079 words.
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Storing Data

- The `json` module allows you to dump simple Python data structures into a file and load the data from that file the next time the program runs.
- You can also use `json` to share data between different Python programs.
- Even better, the JSON data format is not specific to Python, so you can share data you store in the JSON format with people who work in many other programming languages

Using `json.dump()` and `json.load()`

- Let's write a short program that stores a set of numbers and another program that reads these numbers back into memory.
- The first program will use `json.dump()` to store the set of numbers, and the second program will use `json.load()`.

```
import json

numbers = [2, 3, 5, 7, 11, 13]

filename = 'numbers.json'
with open(filename, 'w') as f:
    json.dump(numbers, f)
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 number_writer.py
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ cat nu
mbers.json
[2, 3, 5, 7, 11, 13](base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d
9/chapter_10$
```

```
import json

filename = 'numbers.json'
with open(filename) as f:
    numbers = json.load(f)

print(numbers)
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 number_reader.py
[2, 3, 5, 7, 11, 13]
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```

Saving and Reading User-Generated Data

```
import json

def get_stored_username():
    """Get stored username if available."""
    filename = 'username.json'
    try:
        with open(filename) as f:
            username = json.load(f)
    except FileNotFoundError:
        return None
    else:
        return username

def get_new_username():
    """Prompt for a new username."""
    username = input("What is your name? ")
    filename = 'username.json'
    with open(filename, 'w') as f:
        json.dump(username, f)
    return username

def greet_user():
    """Greet the user by name."""
    username = get_stored_username()
    if username:
        print(f"Welcome back, {username}!")
    else:
        username = get_new_username()
        print(f"We'll remember you when you come back, {username}!")

greet_user()
~
~
~
```

```
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 remember_me.py
What is your name? Joshua
We'll remember you when you come back, Joshua!
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$ python
 remember_me.py
Welcome back, Joshua!
(base) joshua@joshua-VirtualBox:~/Documents/Python_Crash_Course_2nd_Edition/ehmatthes-pcc_2e-00ff4d9/chapter_10$
```