Intro to Coding with Python– `main()`

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (https://jcrouser.github.io/)

Plan for Today

- The `main()` function

A reminder from the syllabus

- "**References**"
  - You should use resources when you need help!
  - And you must cite them! (Give them credit for helping you)
  - In-line citations to any resources you used, including page numbers (if a printed resource) or a direct URL (if an online resource).

- Ex.

Example

*documentations.py - /Users/jcrouser/Google Drive/Teaching/Course Material/CSC111/CSC111/demos/documentations....

```
#-----------------------------------------------
#       Names: Jordan Crouser & Morganne Crouser
#        Date: 26 September 2018
#    Filename: demo.py
# Description: This is a demonstration of how to
#              propertly attribute help on a
#              CSC111 assignment
#-----------------------------------------------

name = input("Enter your name: ")
formatted_string = "{0:>10}".format(name)
print(formatted_string)

# REFERENCES
# I googled how to use the str.format(...) method
# and found the Python documentation here:
# https://docs.python.org/3/library/stdtypes.html#str.format
```

Ln: 17 Col: 60

Recap

- So far, we've been writing code in files as if we were writing it on the console:

```
11
12  def talkToUser(start):
13      print("Welcome to my program! ")
14
15
16  talkToUser(1)
17
```

- When we do this, the Python interpreter executes everything from the **top down**

An alternative

- It is better practice to write the code you want to execute inside a **`main()`** function, e.g.

```
11
12  def talkToUser(start):
13      print("Welcome to my program! ")
14
15  def main():
16       talkToUser(1)
17
18  main()
```

- This lets the interpreter "read ahead" and **then** execute

How this works

- **Remember**: the interpreter reads from the top down, which means that it reads the **definition** first

```
11
12  def talkToUser(start):
13      print("Welcome to my program! ")
14
15  def main():
16       talkToUser(1)
17
18  main()
```

How this works

- Then it reads each line inside the definition, but these don't get **executed** yet

```
11
12  def talkToUser(start):
13      print("Welcome to my program! ")
14
15  def main():
16      talkToUser(1)
17
18  main()
```

How this works

- At this stage, we've given python a "recipe" for what we want it to do when we call **`main()`**

```
11
12  def talkToUser(start):
13      print("Welcome to my program! ")
14
15  def main():
16       talkToUser(1)
17
18  main()
```

- If we stop here, **nothing will actually happen**

How this works

- The real work happens only when we actually **call** the **`main()`** function

```
11
12  def talkToUser(start):
13      print("Welcome to my program! ")
14
15  def main():
16       talkToUser(1)
17
18  main()
```

- When we do, python goes to the **`main()`** function definition and follows the instructions it finds there

Discussion

Why bother?

Just one more thing…

- Suppose I have code I wrote elsewhere
- What happens if someday we want to use the code in this file as **part of another program**?

main.py* salutations.py*

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3   """
4   Ab Mosca
5   02.20.2024
6
7   Importing functions demo
8   """
9
10  def hello():
11      print("Hello!")
12
13  def goodbye():
14      print("Good bye!")
15
16  def main():
17      print("salutations main() running")
18      hello()
19      goodbye()
20
21  main()
22
```

Just one more thing…

- What happens if someday we want to use the code in this file as **part of another program**?

main.py* salutations.py*

```
#!/usr/bin/env pyt
# -*- coding: utf-
"""
Ab Mosca
02.20.2024

Importing function
"""

def hello():
    print("Hello!"

def goodbye():
    print("Good by

def main():
    print("salutat
    hello()
    goodbye()

main()
```

main.py* salutations.py

```
# -*- coding: utf-8 -*-
"""
Ab Mosca
02.20.2024

Main function and importing notes
"""

import salutations

def talkToUser(start):
    if start:
        print("Welcome to my program! ")
        salutations.hello()
    else:
        salutations.goodbye()

def main():
     talkToUser(1)

main()
```

Just one more thing…

- What happens if someday we want to use the code in this file as **part of another program**?

main.py* salutations.py*

```
#!/usr/bin/env pyt
# -*- coding: utf-
"""
Ab Mosca
02.20.2024

Importing function
"""

def hello():
    print("Hello!"

def goodbye():
    print("Good by

def main():
    print("salutat
    hello()
    goodbye()

main()
```

main.py* salutations.py

```
# -*- coding: utf-8 -*-
"""
Ab Mosca
02.20.2024

Main function and importing notes
"""

import salutations

def talkToUser(start):
    if start:
        print("Welcome to my program! ")
        salutations.hello()
    else:
        salutations.goodbye()

def main():
    talkToUser(1)

main()
```

Just one more thing…

What will happen if I run main.py?

main.py* salutations.py*

```
#!/usr/bin/env pyt
# -*- coding: utf-
"""
Ab Mosca
02.20.2024

Importing function
"""

def hello():
    print("Hello!"

def goodbye():
    print("Good by

def main():
    print("salutat
    hello()
    goodbye()

main()
```

main.py* salutations.py

```
# -*- coding: utf-8 -*-
"""
Ab Mosca
02.20.2024

Main function and importing notes
"""

import salutations

def talkToUser(start):
    if start:
        print("Welcome to my program! ")
        salutations.hello()
    else:
        salutations.goodbye()

def main():
     talkToUser(1)

main()
```

Jus
thi

What will happen if I run main.py?

/Users/abmosca/Documents/Documents/Westfield/classes/CAIS117-S24/website/demos/main.py

main.py | salutations.py

```
2   # -*- coding: utf-8 -*-
3   """
4   Ab Mosca
5   02.20.2024
6
7   Main function and importing notes
8   """
9
10  import salutations
11
12  def talkToUser(start):
13      if start:
14          print("Welcome to my program! ")
15          salutations.hello()
16      else:
17          salutations.goodbye()
18
19  def main():
20       talkToUser(1)
21
22  main()
23
24
```

Source | Console | Object

Usage

Here you can get help of any

Help | Variable Explorer | Plots | Files

Console 2/A

```
14.0.6 ]
Type "copyright", "credits" or "license" for more
information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/abmosca/Documents/Documents/
Westfield/classes/CAIS117-S24/website/demos/main.py',
wdir='/Users/abmosca/Documents/Documents/Westfield/
classes/CAIS117-S24/website/demos')
salutations main() running
Hello!
Good bye!
Welcome to my program!
Hello!

In [2]:
```

Discussion

- **What we need:** a way to tell python to behave one way when we **run it as a "stand-alone" program**, and a different way when we `import` it

Ideas?

Python convention

- We can use an **if** statement to tell python to call the **main()** function only if the program is being run directly

main.py | salutations.py*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Ab Mosca
02.20.2024

Importing functions demo
"""

def hello():
    print("Hello!")

def goodbye():
    print("Good bye!")

def main():
    print("salutations main() running")
    hello()
    goodbye()

if __name__ == "__main__":
    main()
```

Python convention

- This is a little bit **confusing**: we named the function we created to hold our program **`main()`**

main.py | salutations.py*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Ab Mosca
02.20.2024

Importing functions demo
"""

def hello():
    print("Hello!")

def goodbye():
    print("Good bye!")

def main():
    print("salutations main() running")
    hello()
    goodbye()

if __name__ == "__main__":
    main()
```

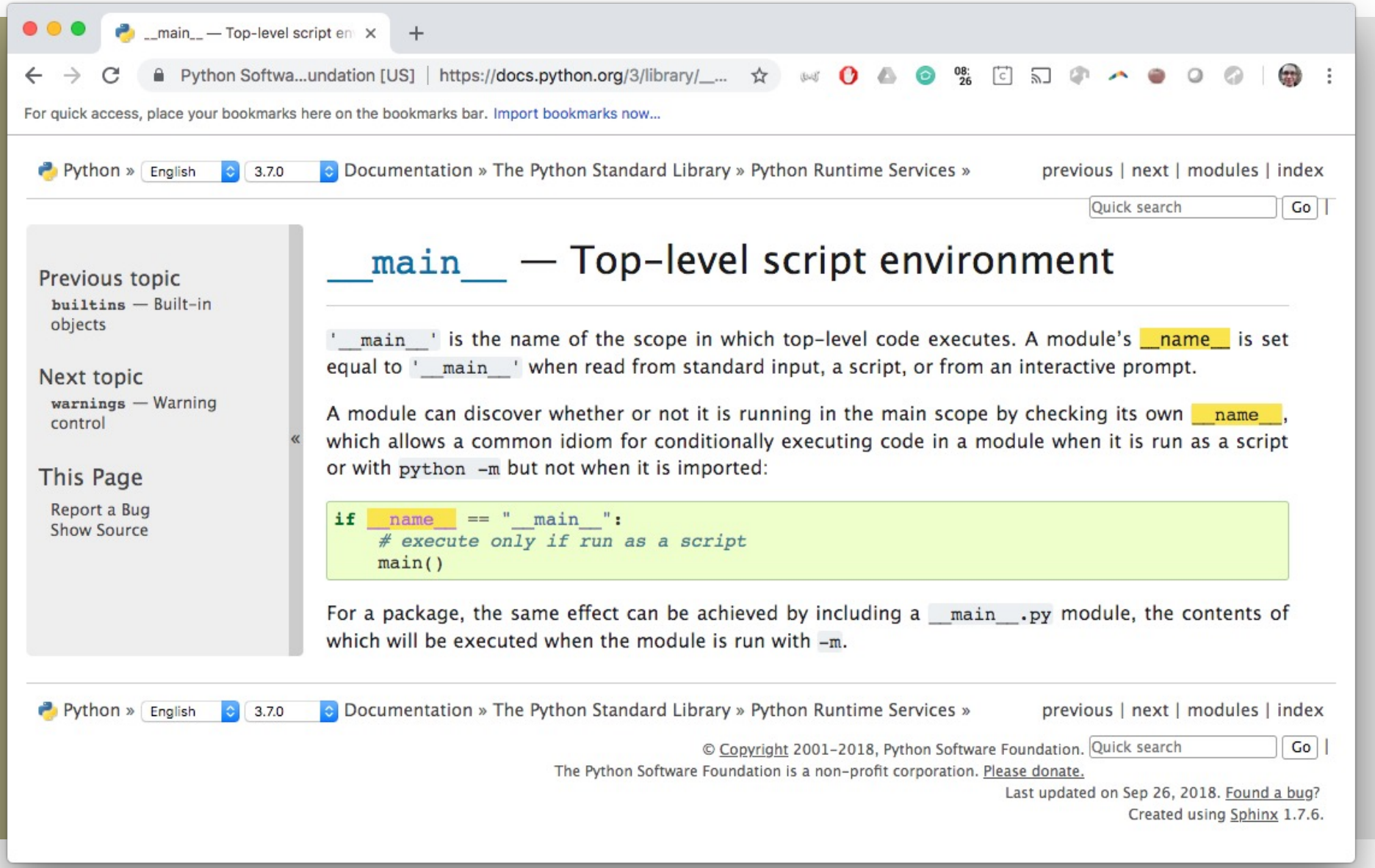
Python convention

- In our **`if` statement**, we're asking whether some variable called **`__name__`** is equal to the string **`"__main__"`** (not to mention I don't recall initializing anything called **`__name__`**…)

main.py | salutations.py*

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3   """
4   Ab Mosca
5   02.20.2024
6
7   Importing functions demo
8   """
9
10  def hello():
11      print("Hello!")
12
13  def goodbye():
14      print("Good bye!")
15
16  def main():
17      print("salutations main() running")
18      hello()
19      goodbye()
20
21  if __name__ == "__main__":
22      main()
```

To the documentation!




DEMO
TIME

15-minute exercise

- Write a program that contains a `main()` function, which contains instructions for printing out the phrase:

```
"Today is not Friday :-("
```

- Use an `if` statement combined with checking the value of the `__name__` variable to call `main()` only when the program is run directly
- Add an `else` statement so that whenever the program ("module") is `import`ed, it prints out the phrase:

```
"Maybe today...?"
```

Discussion

What did you come up with?

Takeaways

- Programs ("modules") that are well-organized are **easier to read**, more **versatile**, and potentially **more efficient**
- The first step we'll take toward organizing our code is to include a `main()` function, which includes the instructions we want our program to run
- To make it easier to `import` code we write now into later modules, we will follow the convention of including:

```
if __name__ == "__main__":
    main()
```

at the end of each module

Helpful tip: have a starter template

starter.py - /Users/jcrouser/Google Drive/Teaching/Course Material/CSC111/CSC111/demos/start...

```
#-----------------------------------------------
#       Names: Jordan Crouser & <PARTNER>
#        Date: <DATE>
#    Filename: starter.py
# Description: This is a demonstration of how to
#              organize your starter code (incl.
#              a main function and scope check)
#-----------------------------------------------

def main():
    # This is where my code will go

if __name__ == "__main__":
    main()

# REFERENCES
```

Ln: 3 Col: 21