Python Workshop

About Python

- Python is a general purpose programming language
- It can be used for all sorts of things, including:
    - Web Development
    - Desktop Apps
    - Data Science, AI/ML

Today we will be running through some of the basics of Python.

Your Environment

- You will need to install the Python Development Environment so that you are able to write your code.
- To do this, go to this link: https://www.python.org/downloads/

Your Environment

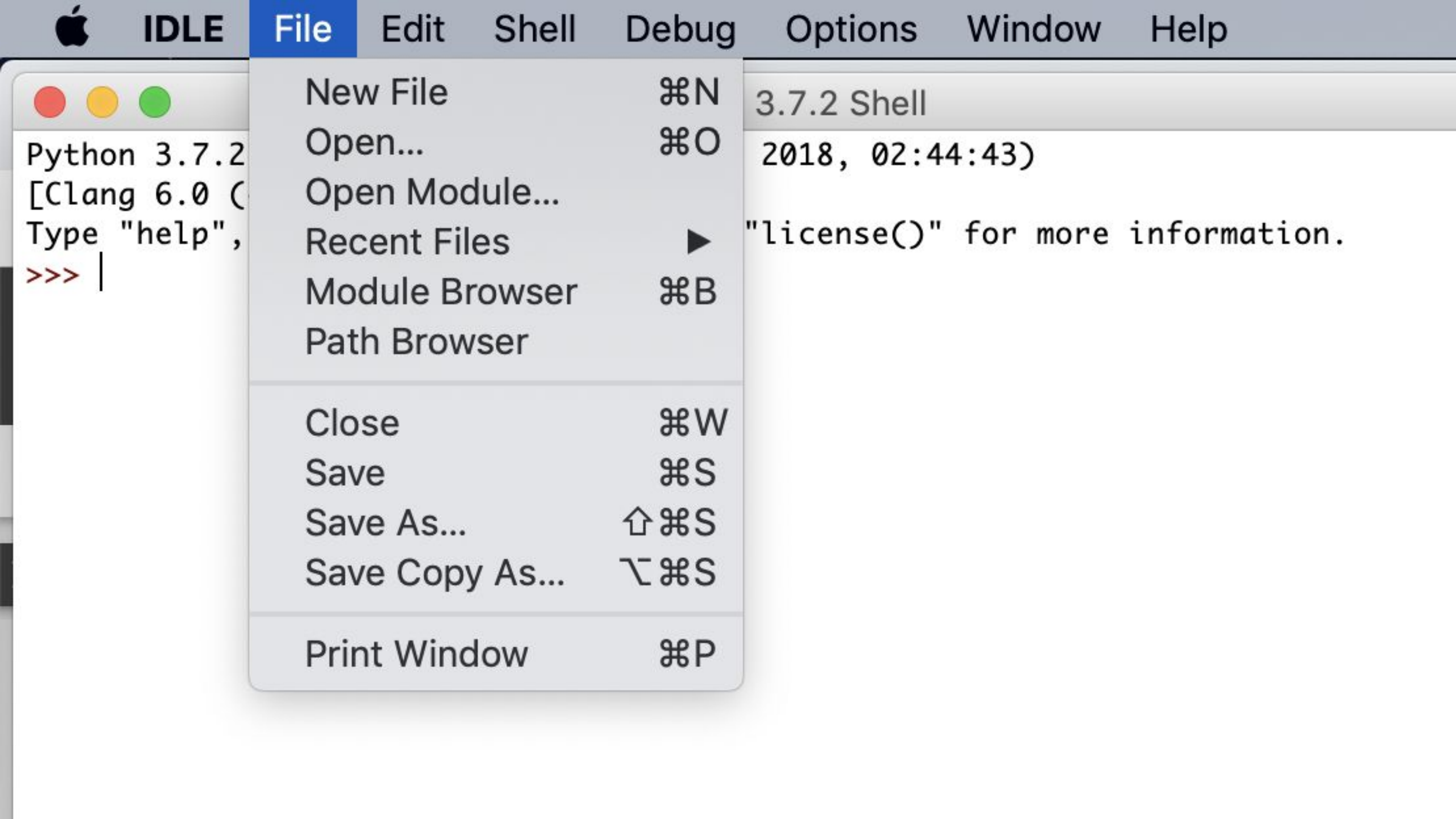
- Once you have Python installed, find it in your app tray.

Your Environment

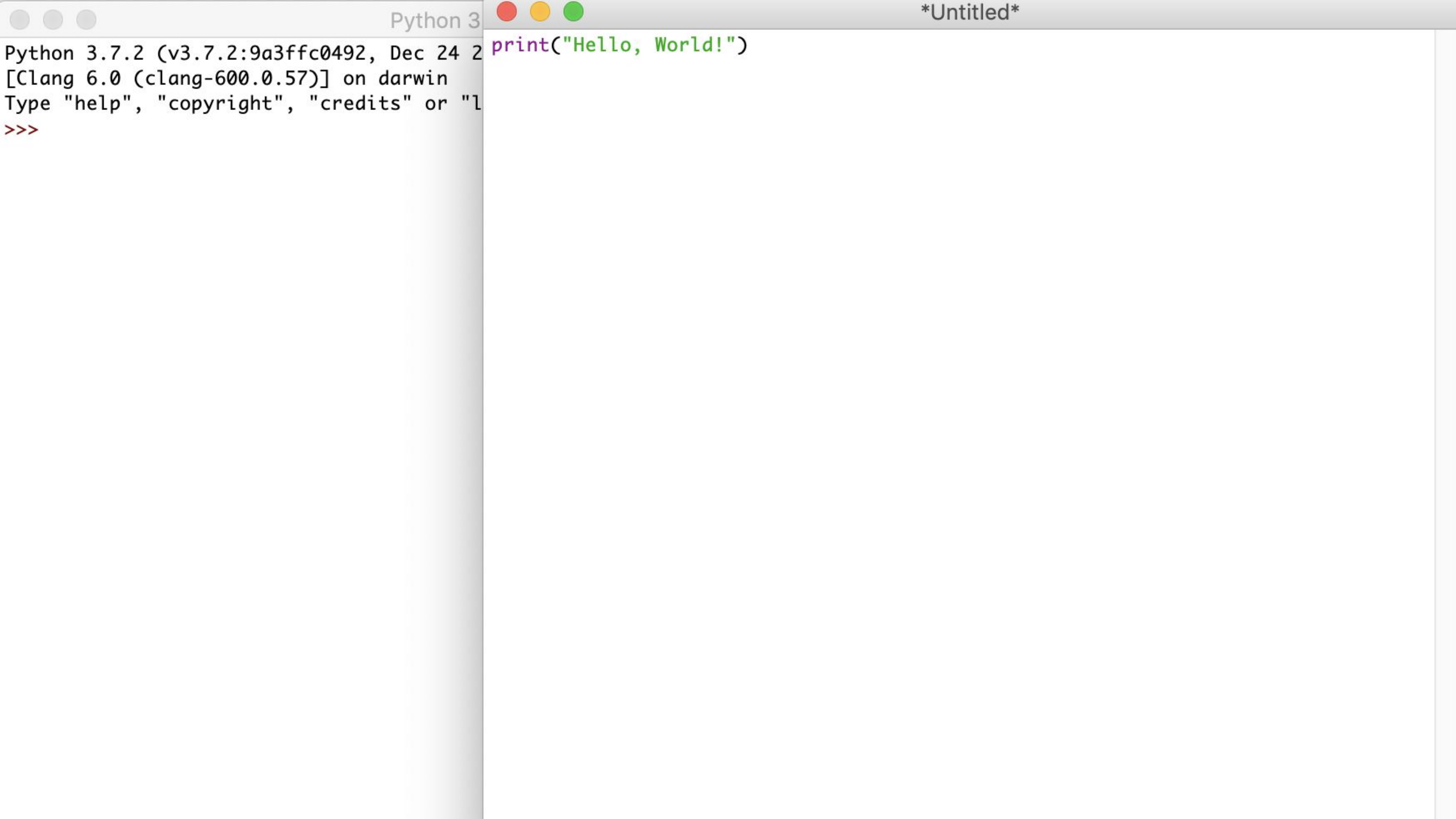
- Now create a folder somewhere you'll find entitled 'Python Tutorial'

Hello World!

- The first program we are going to write is called "Hello World", all it does is get the computer to **print** out on the screen the phrase "Hello, World!".
- It is traditional for programmers who are learning a new language to start with this program.
- To write this, all you have to do is create a new file in your IDLE and type out ‘print(“Hello, World!”)’



```
Python 3.7.2                                  2018, 02:44:43)
[Clang 6.0 (
Type "help",                                  "license()" for more information.
>>> 
```

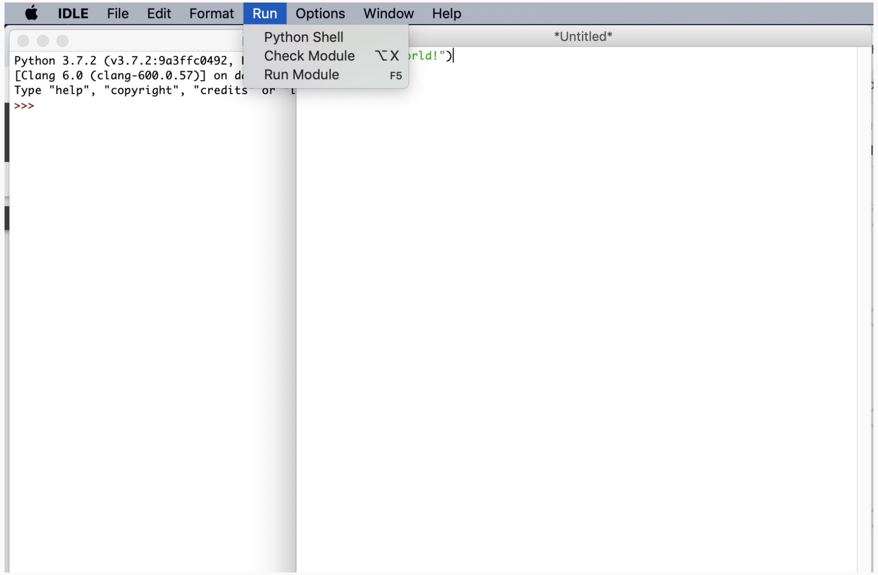


```
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "l
>>>
```

```
print("Hello, World!")
```

Hello, World!

- Let’s run our program! To do this, just click Run Module

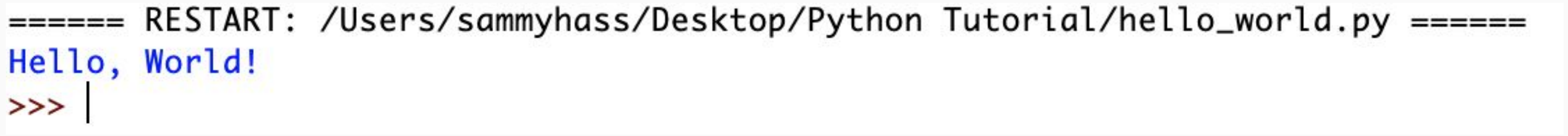


```
Python 3.7.2 (v3.7.2:9a3ffc0492,
[Clang 6.0 (clang-600.0.57)] on d
Type "help", "copyright", "credits" or
>>>
```

```
orld!")
```

Hello, World

- You will be prompted to save your file as ‘hello_world’ in the ‘Python Tutorial’ folder you created earlier.
- Your code will then run and you should see ‘Hello, World!’ printed out onto your screen.



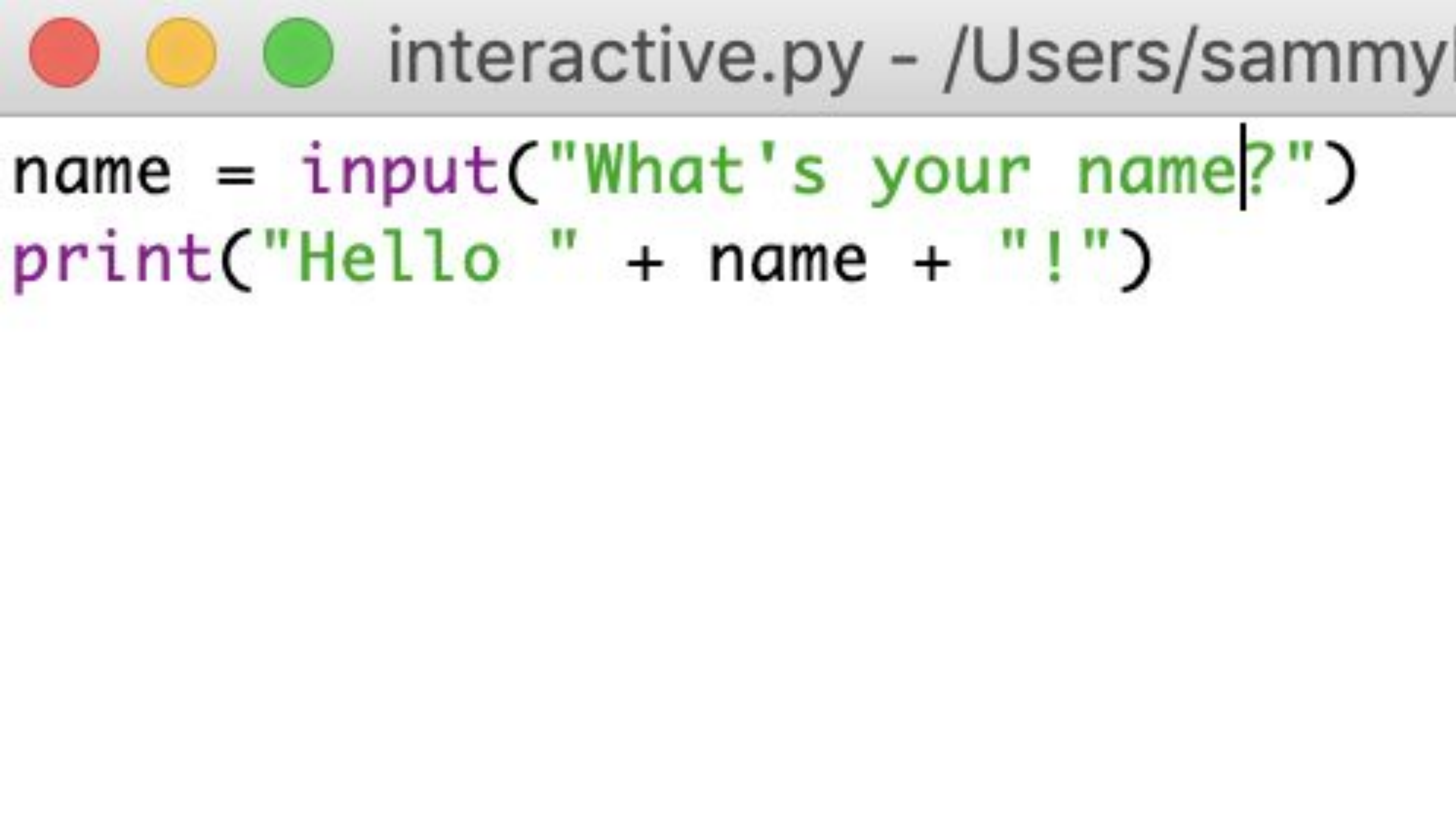
```
====== RESTART: /Users/sammyhass/Desktop/Python Tutorial/hello_world.py ======
Hello, World!
>>> |
```

Well Done!

- You’ve written your first program! Well done! You’re well on your way to becoming a programmer now!

Adding Interactivity

- Okay, now our program says hello to us, buy we want more! “Hello World” is not very personal. I want my program to say “Hello Sammy” and not “Hello World”.
- To do this, the program needs to be able to take some kind of **input** (The user's name) and then we need to store it so that we can use it again to say hello.
- Create a new file!



```
name = input("What's your name?")
print("Hello " + name + "!")
```

Adding Interactivity

- Now run your code (save it as ‘interactive’) and notice that your program asks for your name and then prints it out!

Wait what just happened?

We started off by asking our user to input their name, we then stored that input into a **variable** called 'name' - that was line 1!

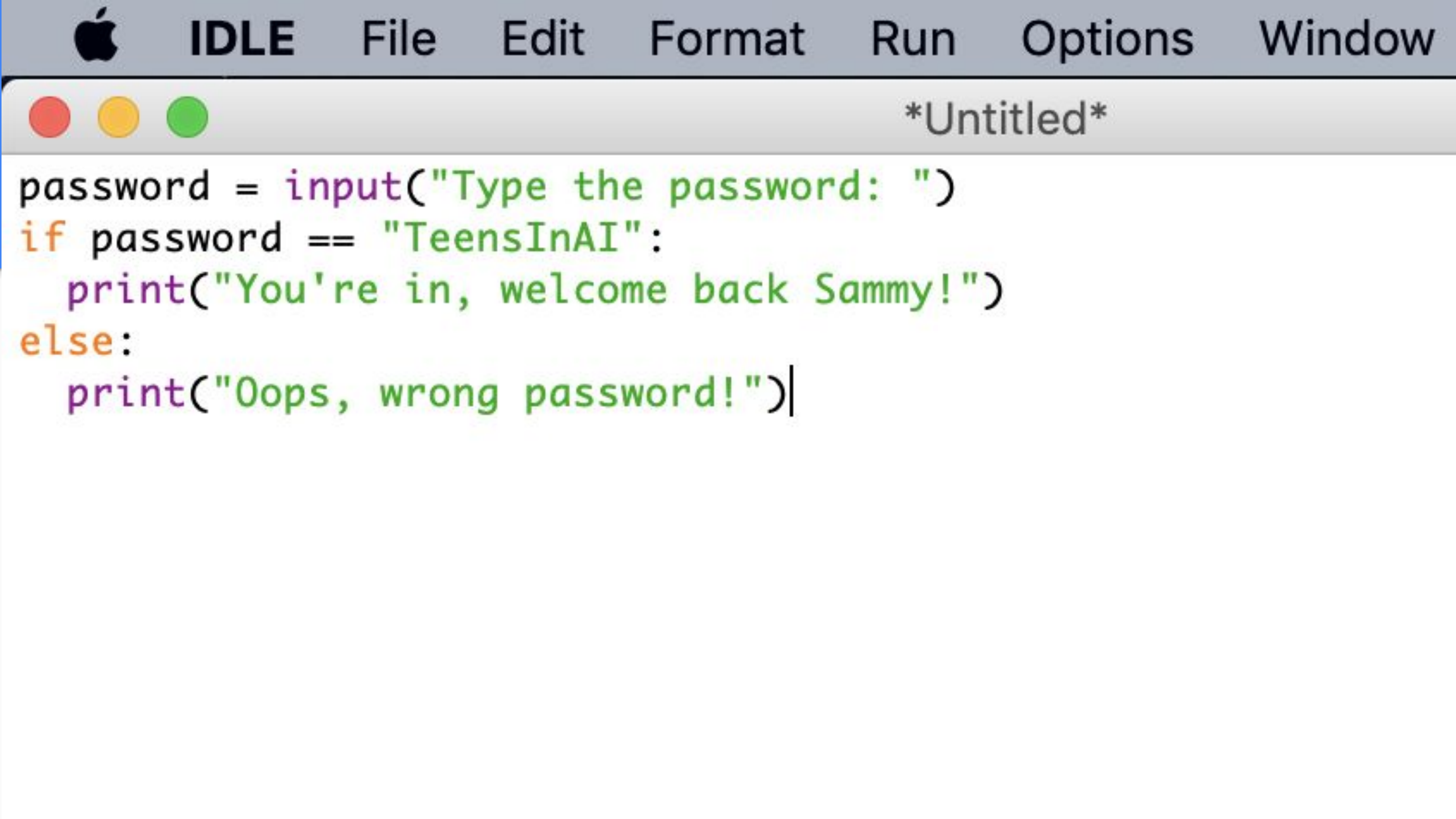
On line two we then print out "Hello " then the user's name and then an exclamation mark! In this circumstance, when we add text (also known as **strings**) to each other, we are just putting them side by side.

More on variables

- **What is a variable?** A variable is a piece of **data** that is stored in our computer's memory.
- **What kinds of data can I store in a variable?** A lot! You can store text, you can store numbers, you can store booleans (either true or false) and you can even store lists of different data types.
- **How do I make one?** Below are a couple of examples of how to declare variables.
  - PI = 3.1415926535 <- Declaring a variable containing the value of PI, a non whole number
  - name = "Sammy" <- Declaring a variable containing text
  - age = 17 <- Declaring a variable containing a whole number
  - isAwesome = True <- Declaring a variable with a boolean value (True/False)
  - favouriteNumber = PI <- Setting the value of one variable equal to the value of another
  - subjects = ["Maths", "Further Maths", "Computer Science"] <- Declaring a variable which stores a list of values.

Conditionals

- Sometimes when we’re coding, we want to do something only **if** a certain condition is met. This is where conditionals help us. A conditional statement is a **block of code** that allows us to do something under one circumstance, and another if that original circumstance is met.
- Create a new file



```python
password = input("Type the password: ")
if password == "TeensInAI":
  print("You're in, welcome back Sammy!")
else:
  print("Oops, wrong password!")
```

Conditionals

On line two, of this code, you may see something that looks rather unfamiliar, we used equals signs before to assign a variable with a value. However, when you see two equals signs together, it is actually called a **comparison operator** and its used to check if the values either side of it are the same. If the comparison comes back correct the first block of code runs **else**, the next block of code runs.

Here is a list of the comparison operators programmers most often use:

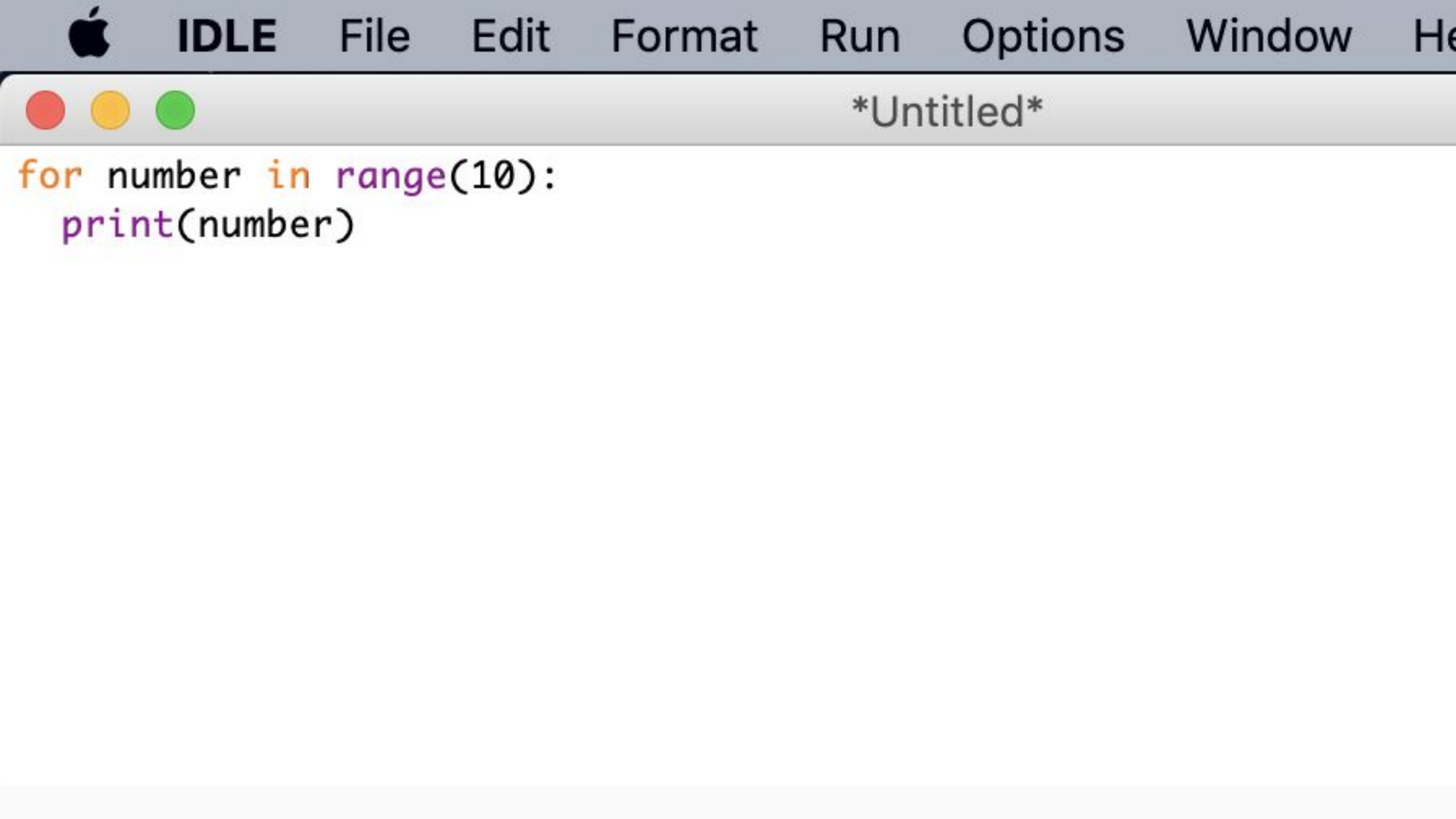
- == (Equal to, 6==6 is a True statement)
- > (Greater than, for example 6 > 5 is a True statement)
- >= (Greater than or equal to, for example 5 >= 5 is a True statement)
- < (Less than or equal to, for example 3 < 5 is a True statement)
- <= (Less than or equal to, for example 3 <= 5 is a True statement)

Loops

If we want a piece of code to run over and over again, it is not at all efficient to write and rewrite a block over and over, this is why we use loops! Loops are simply just blocks of code which allow us to run a certain set of instructions multiple times.

For Loop

The for loop is a loop that can **iterate** over members of a sequence, only stopping once it reaches the end of that sequence.



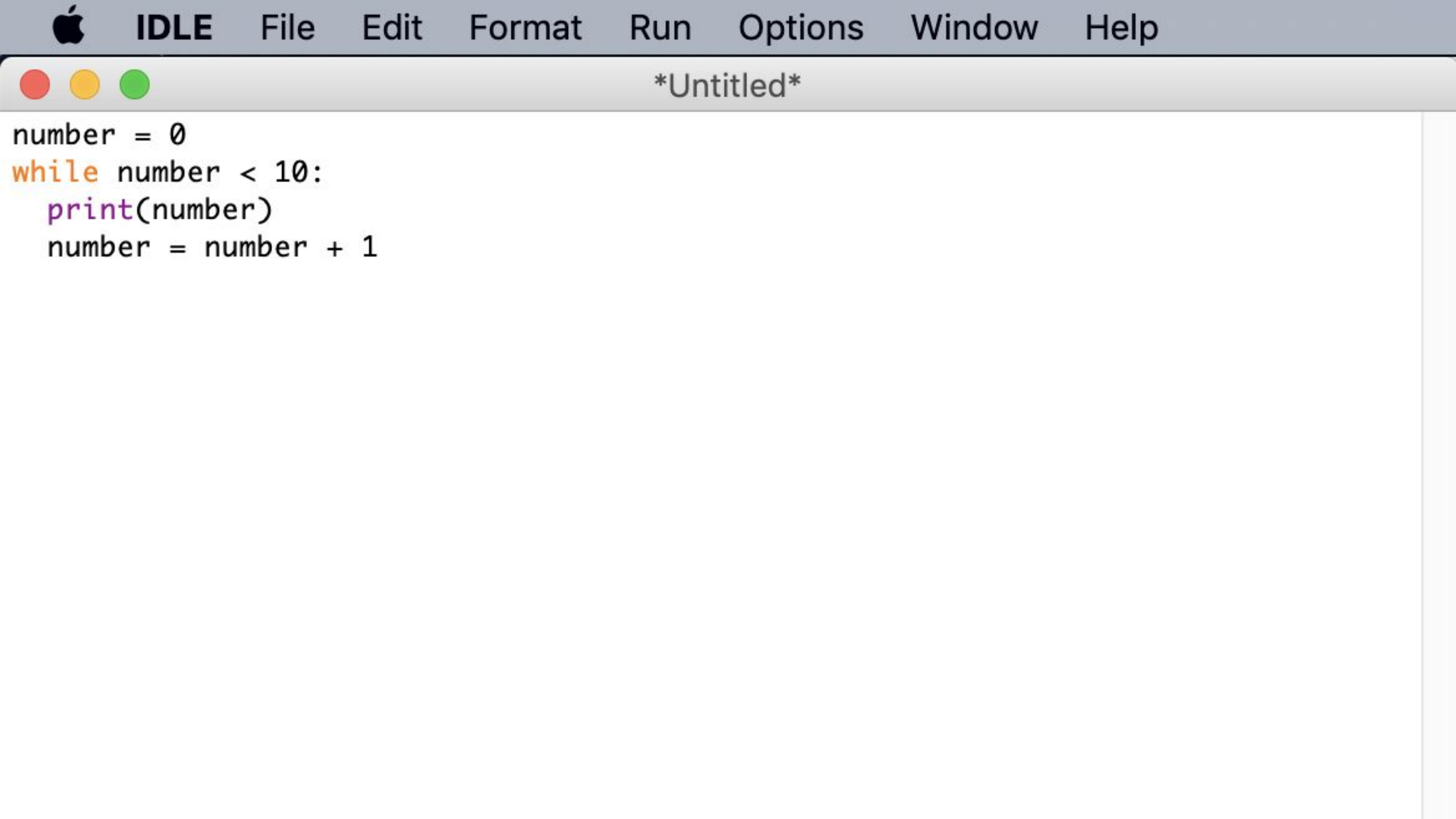
```python
for number in range(10):
  print(number)
```

For Loop

Run your code and notice that it prints out the numbers 0 through 9.

While Loop

The while loop is a loop that runs until a certain condition is met.



```
number = 0
while number < 10:
  print(number)
  number = number + 1
```

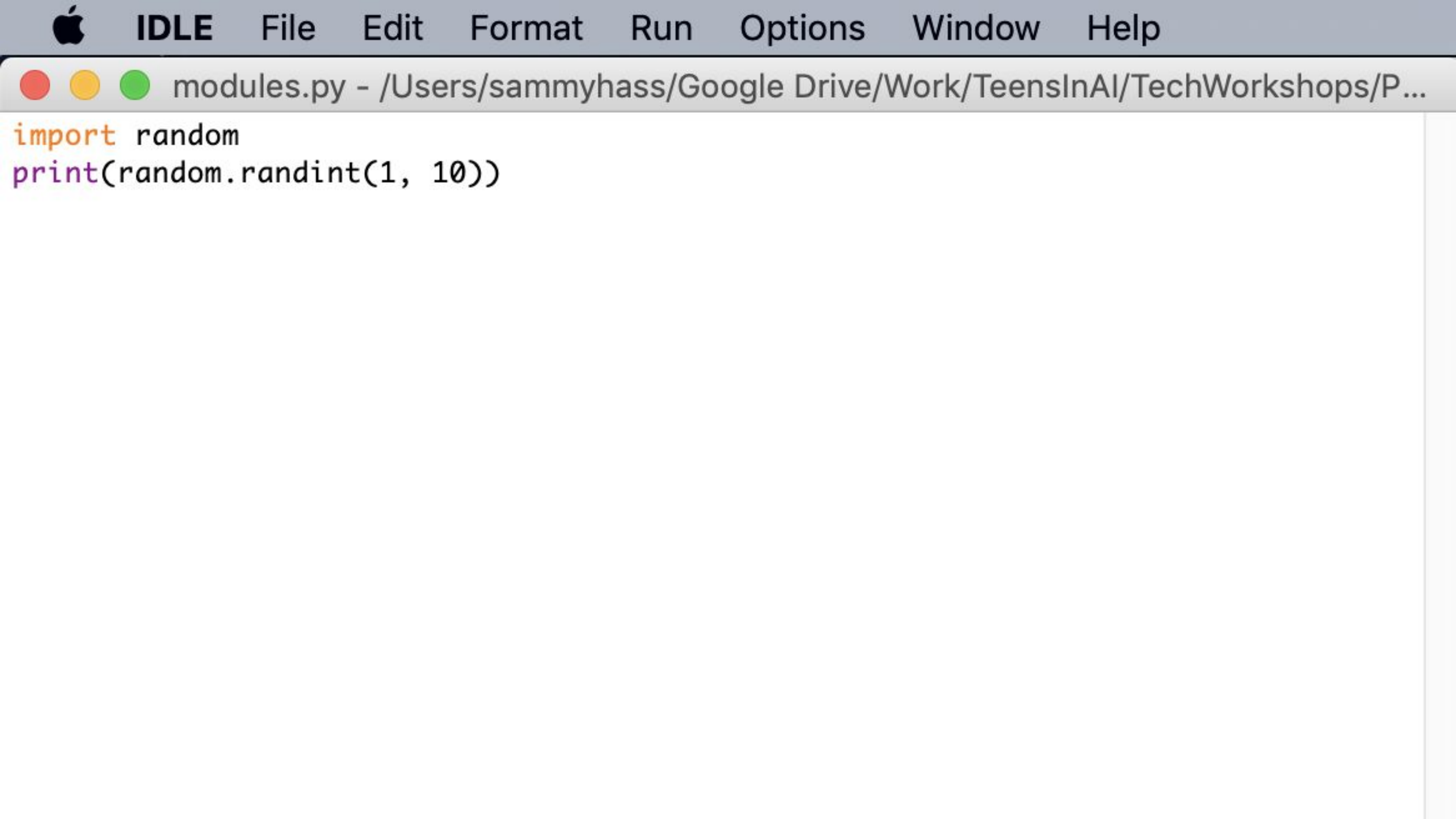
While Loop

Notice this code does the same thing as our for loop but just in a longer way. It is for this reason that when we program we try our best to use just for loops.

Modules

With python you can make use of premade code and put it in your program. This prewritten code is called a module and you can use them in all your python programs.

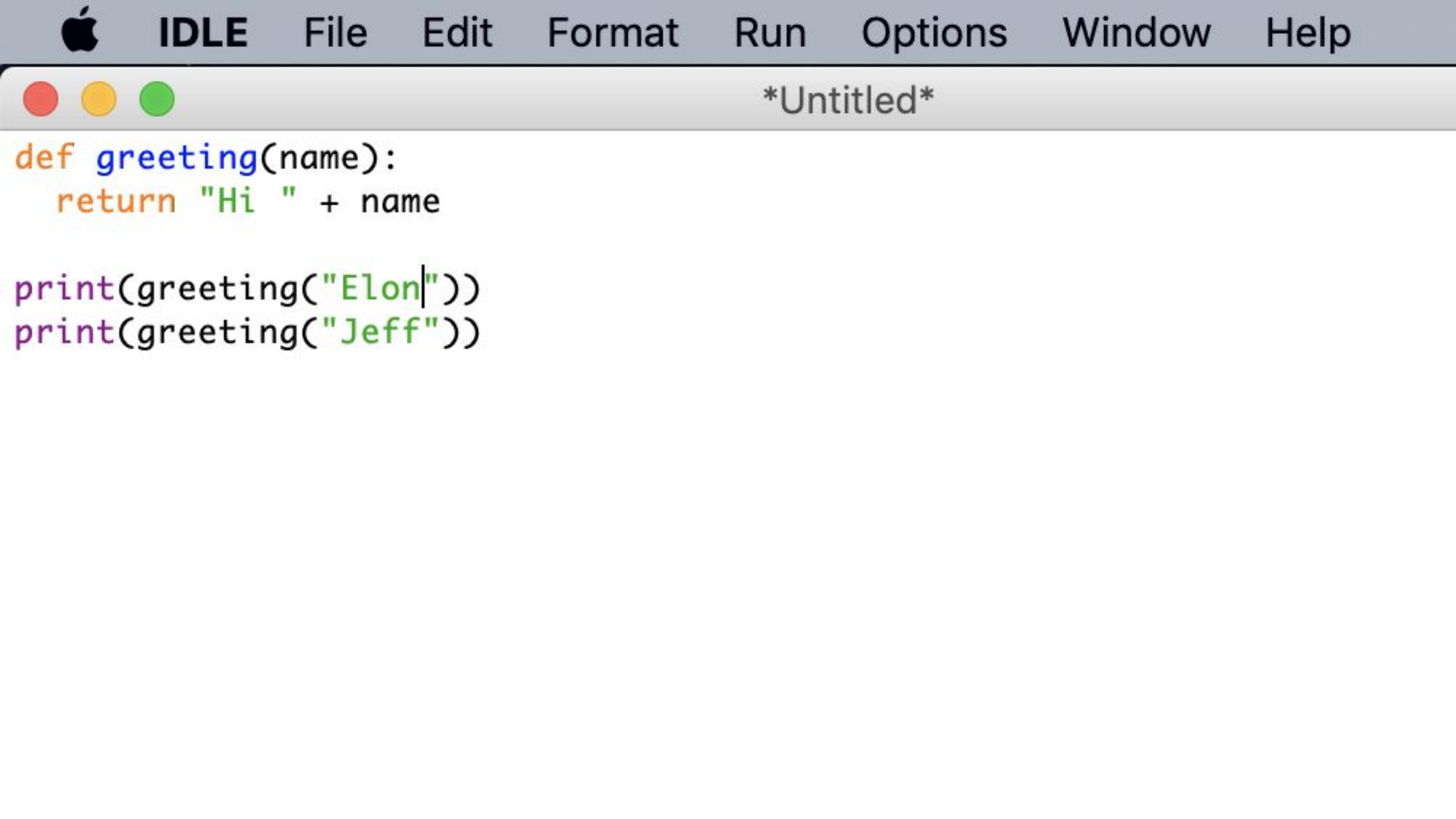
To use a module, you first have to **import** it.


IDLE
File
Edit
Format
Run
Options
Window
Help
modules.py - /Users/sammyhass/Google Drive/Work/TeensInAI/TechWorkshops/P...

```python
import random
print(random.randint(1, 10))
```

Functions

Functions are a way to use the same piece of code multiple times within your program. They are really useful!



```python
def greeting(name):
  return "Hi " + name

print(greeting("Elon"))
print(greeting("Jeff"))
```

Challenges!

Now it's your turn! Choose a couple of these challenges to do and pair up with the person next to you to solve them. If you get stuck, just ask for help. Another great way to get help, is to google, it sounds like cheating but its not! Programmers do not remember all the right ways to do things, its natural people forget things all you need to be a programmer are the basic concepts, the rest, you can learn as you go!.

1. Create a program that takes in a user's information and gives it back to the user in a presentable way.
2. Create a fizzbuzz program for numbers 1 to 100
3. Create a program which acts as a chatbot, when the user writes something, the program responds appropriately (hint: nested if statements)
4. Create a multiple choice quiz where a user is alerted if they get a question right or wrong
5. Want more of a challenge? Try using matplotlib to show images