Day 1: Python Basics

Monday, 7/12/21

Agenda

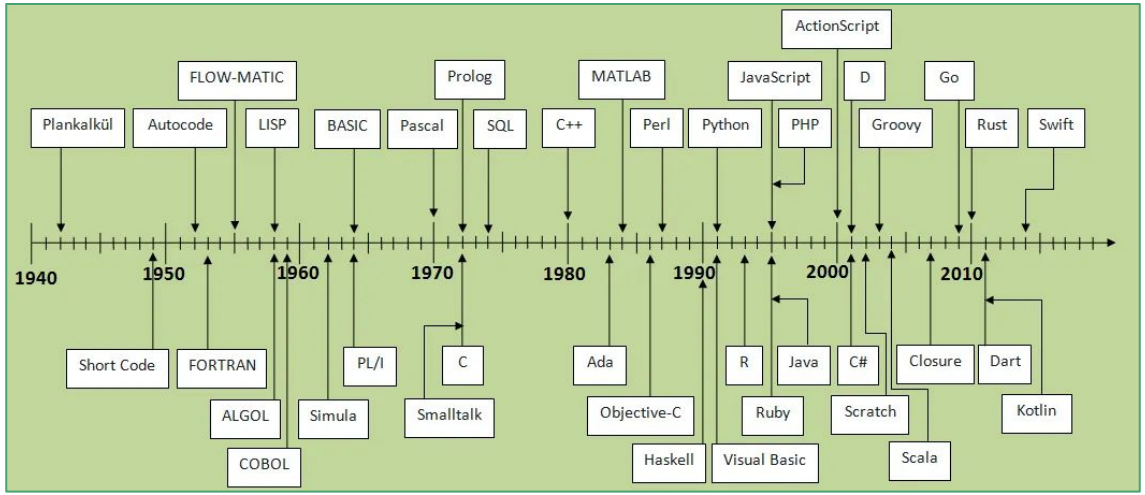
1. Introduction
    a. What is Python?
    b. Why are we using Python?
2. How programs work
    a. Input ➜ Output
    b. Examples
    c. Data types
3. Live coding
    a. Switch to Replit

INTRODUCTION

Python: a friendly programming language

What is *Python*?

- Python is a programming language
- Programs give the computer a set of instructions or tasks to do
- There have been many different programming languages over the years:



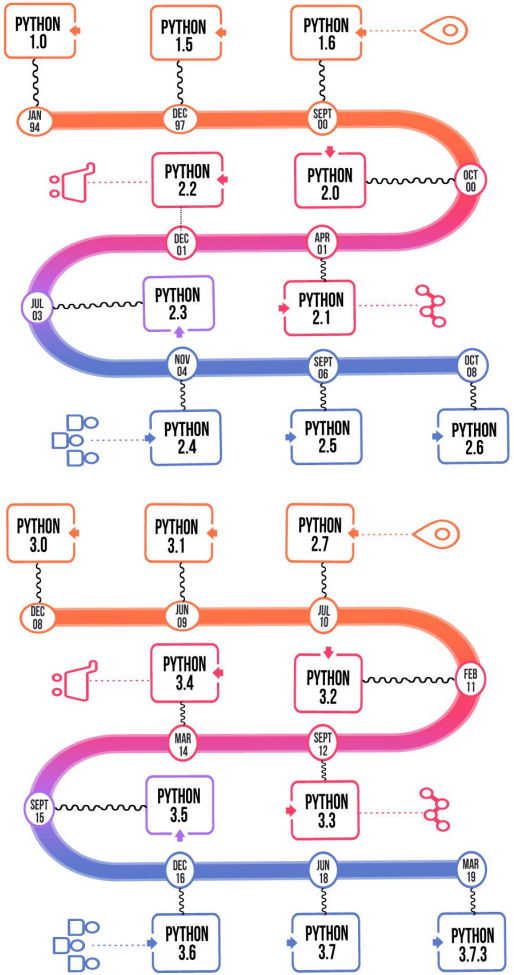
Python: Quick Facts

- The first version of Python was released in 1994
- Since then, there have been many versions of Python
  - Similar to software, each *version* of Python is unique
  - We'll use the version of Python that comes with replit:
    - Python 3.8.10

```
~/LecturesLecture-01$ python
Python 3.8.10 (default, May  5 2021, 03:01:07)
```

- Some of the functionality of Python 3.8.10 **will not apply** to ALL versions of Python!

^ Something to keep in mind when you work with different (old or new) versions of Python in the future.



Python: Quick Facts

- Guido van Rossum - creator of Python
  - In December 1989, he wanted a hobby project to work on during his work's Christmas break
  - The hobby project? *Python*
- Rossum's stated goals for Python:
  - An easy and intuitive language
  - Open source, so anyone can contribute to its development
  - Code that is as understandable as plain English
  - Suitability for everyday tasks, allowing for short development times

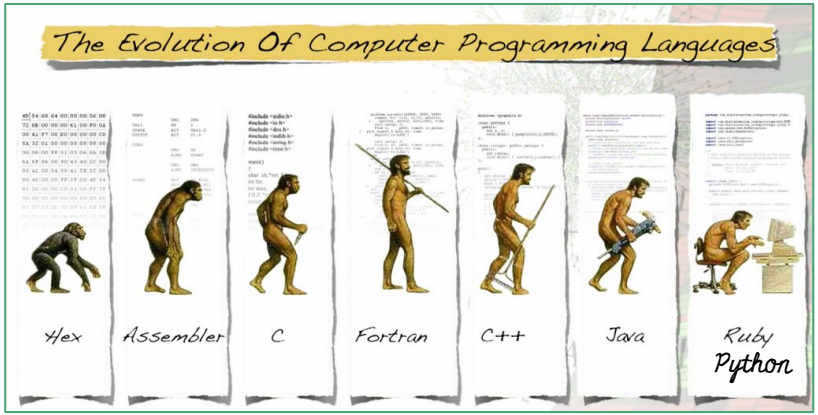
Guido van Rossum

Now at Microsoft (since 2019)

Python: Quick Facts

Learning to program for the first time = many new concepts at once. Python helps!

- **Easy to write** (*an easy and intuitive language*)
- **Multipurpose** (*suitability for everyday tasks*)
- **Widely supported** (*open source, so anyone can contribute to its development*)
- **Easy to read** (*code that is as understandable as plain English*)

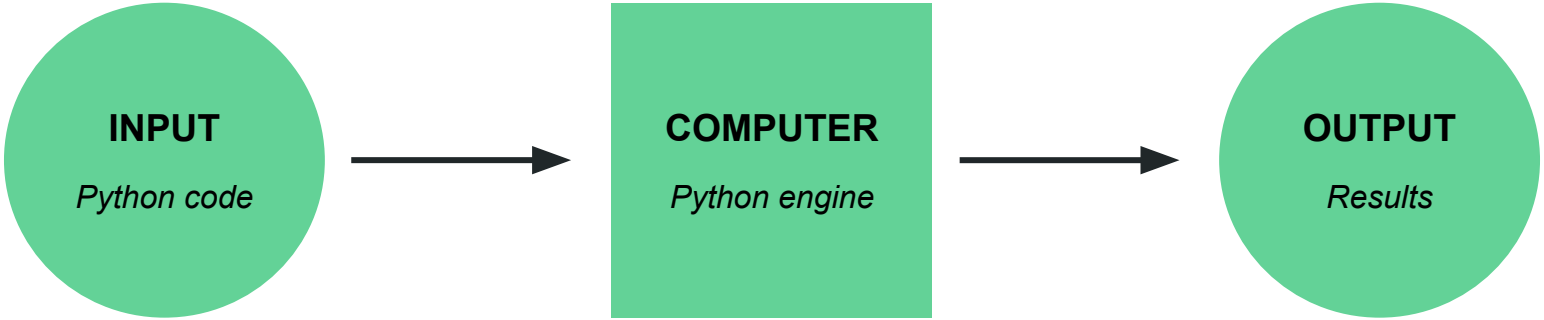


CODING BASICS

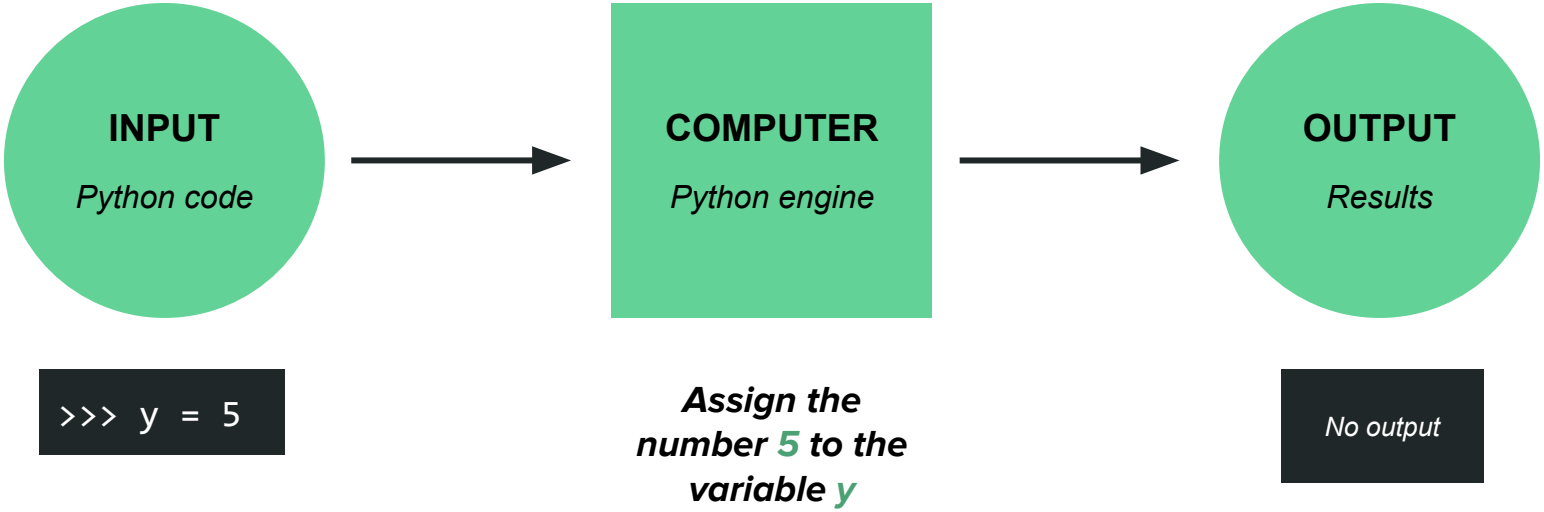
How do programs work?

Coding basics

- Programs we write get compiled and executed by the computer
  - Think of this as the computer processing our code as a set of instructions, or as data input
  - Data goes **in** to the computer, the computer does something with it, then data comes **out**



Different inputs give different results



(Aside: declaring and assigning values to variables)

Assign the value 5 to the variable y

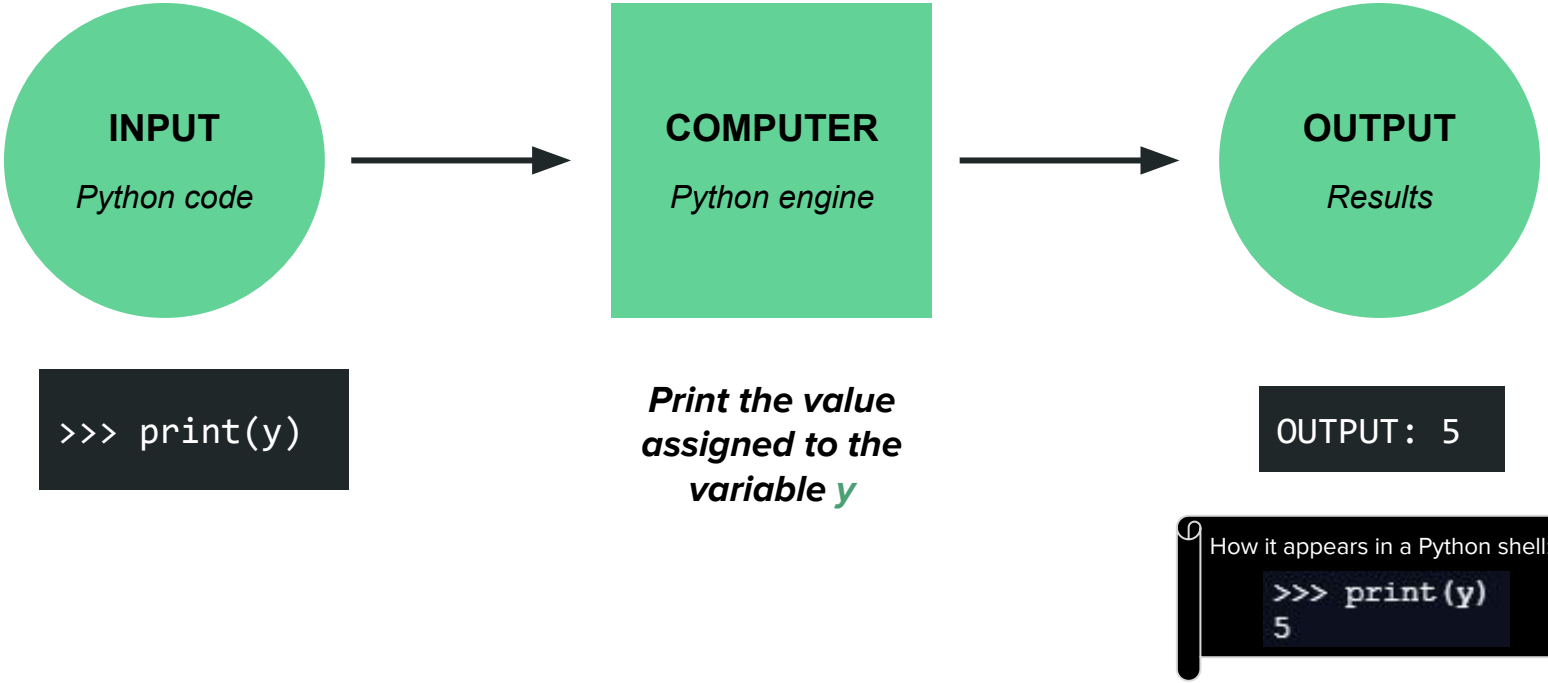
```
y = 5
```

The word to the left of the '=' is our variable that we're declaring in Python. It can be named whatever want, but it cannot contain spaces

The '=' sign means that we are assigning the value on the right to the variable on the left

The value on the right of the '=' is the value that gets *assigned* to the variable

Different inputs give different results



(Aside: printing / outputting to the screen)

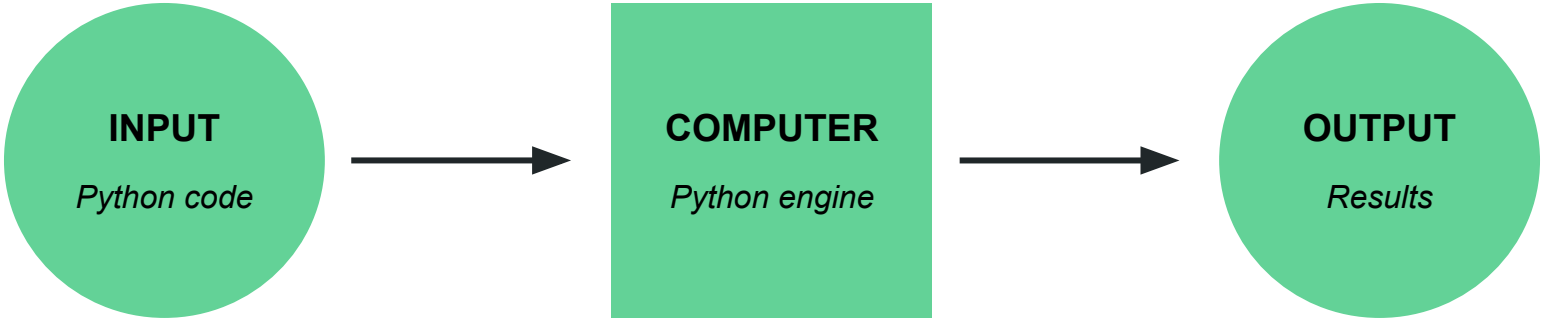
Print the value assigned to the variable y

```
print(y)
```

This is called a “print function”. It outputs whatever is inside the ( )’s to the screen

The value inside of the ( ) gets outputted to the screen

Different inputs give different results




```
>>> print(5+2)
```

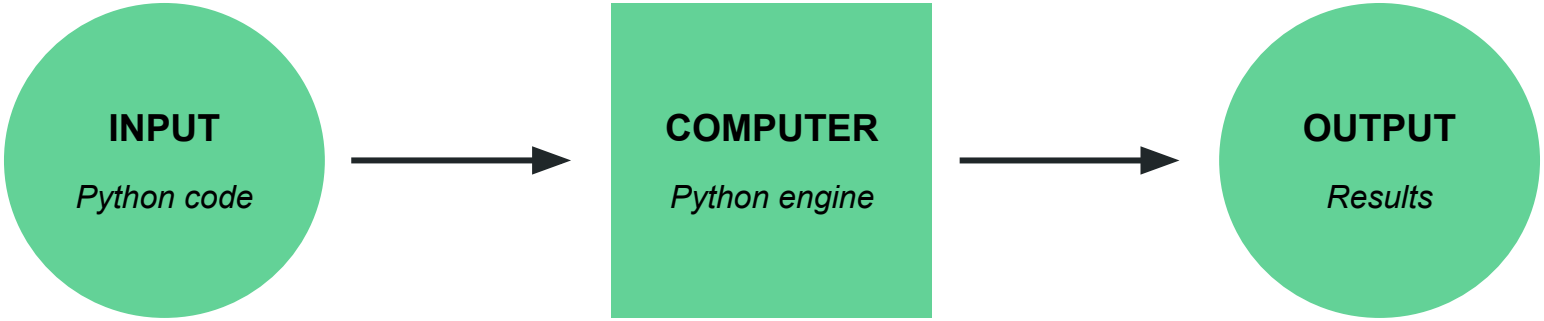
Print the value
5+2

```
OUTPUT: 7
```

How it appears in a Python shell:

```
>>> print(5+2)
7
```

Different inputs give different results

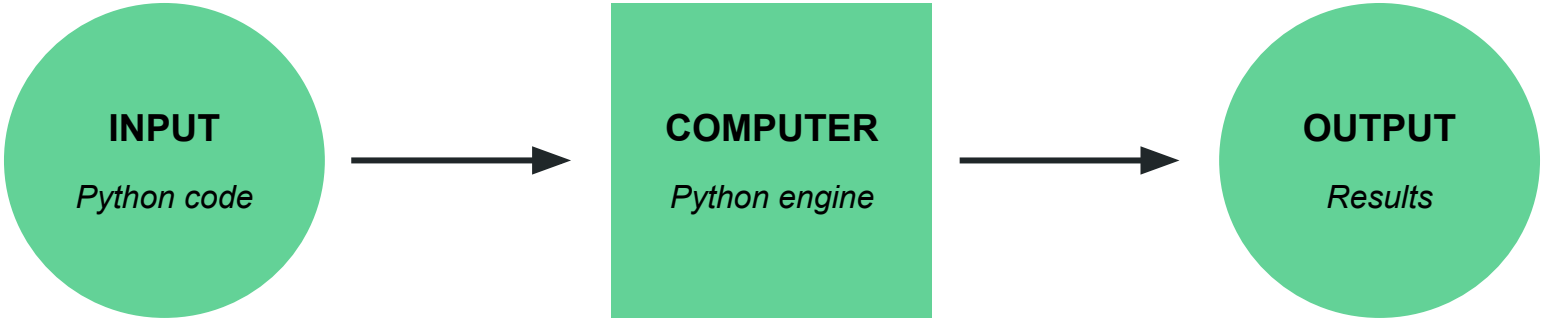



```
>>> print(“Hello World!”)
```

Print the value
“Hello World!”

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

Different inputs give different results




```
>>> x = ((1 + 2) * 5) + 5
>>> print(x)
```

Assign the number ((1 + 2) * 5) + 5 to the variable x

Print the value assigned to the variable x

```
>>> x = ((1 + 2) * 5) + 5
>>> print(x)
20
```

Your turn!

What are the outputs to the following inputs?

1

```
>>> w = 2
>>> print(w)

OUTPUT: ?
```

2

```
>>> message = “Hi mom!”
>>> print(message)

OUTPUT: ?
```

3

```
>>> x = 2 * 3
>>> print(x * 2)

OUTPUT: ?
```

Answers

What are the outputs to the following inputs?

1

```
>>> w = 2
>>> print(w)
```

```
>>> w = 2
>>> print(w)
2
```

2

```
>>> message = “Hi mom!”
>>> print(message)
```

```
>>> message = "Hi mom!"
>>> print(message)
Hi mom!
```

3

```
>>> x = 2 * 3
>>> print(x * 2)
```

```
>>> x = 2 * 3
>>> print(x * 2)
12
```

PYTHON TYPES

Examples of data types

Python types

This little Python program seems intuitive:

```
>>> message = “Hi mom!”
>>> print(message)
```

```
OUTPUT: Hi mom!
```

Python types

But this is less intuitive...

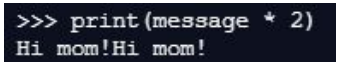
```
>>> message = “Hi mom!”
>>> print(message * 2)
```

What do you think the output will be?

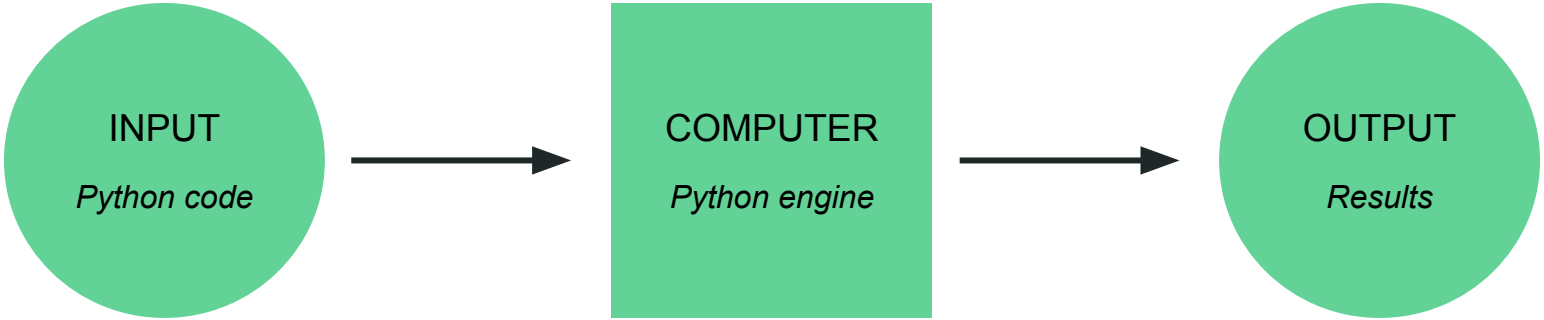
Python types

The output is the text “Hi mom!” repeated twice, back to back, like so:

```
>>> message = “Hi mom!”
>>> print(message * 2)

OUTPUT: Hi mom!Hi mom!
```



Python types




```
>>> message = “Hi mom!”
>>> print(message * 2)
```

Assign the string “Hi mom!” to the variable message

Multiply message by 2
Print the result

```
OUTPUT: Hi mom!Hi mom!
```

Python types

What is similar about these two pieces of code?

```
>>> message = “Hi mom!”
>>> print(message * 2)

OUTPUT: Hi mom!Hi mom!
```

```
>>> x = 2 * 3
>>> print(x * 2)

OUTPUT: 12
```

Python types

Both of the print statements look similar: `print(variable * 2)`

```
>>> message = “Hi mom!”
>>> print(message * 2)

OUTPUT: Hi mom!Hi mom!
```

```
>>> x = 2 * 3
>>> print(x * 2)

OUTPUT: 12
```

Python types

But why is the output on the right `12`, and not the number 6 printed twice - `66`?

After all, `“Hi mom!” * 2` outputs ‘Hi mom!’ twice - `Hi mom!Hi mom!`

```
>>> message = “Hi mom!”
>>> print(message * 2)

OUTPUT: Hi mom!Hi mom!
```

```
>>> x = 2 * 3
>>> print(x * 2)

OUTPUT: 12
```

Python types

`message` is a **str** (String / text)

`x` is an **int** (Integer / number)

Python knows “behind the scenes” whether the variable you’re declaring is a number, a decimal value, a string, or something else entirely (which we’ll learn more about later)

```
>>> message = “Hi mom!”
>>> print(message * 2)

OUTPUT: Hi mom!Hi mom!
```

Python knows `message` is a **str**

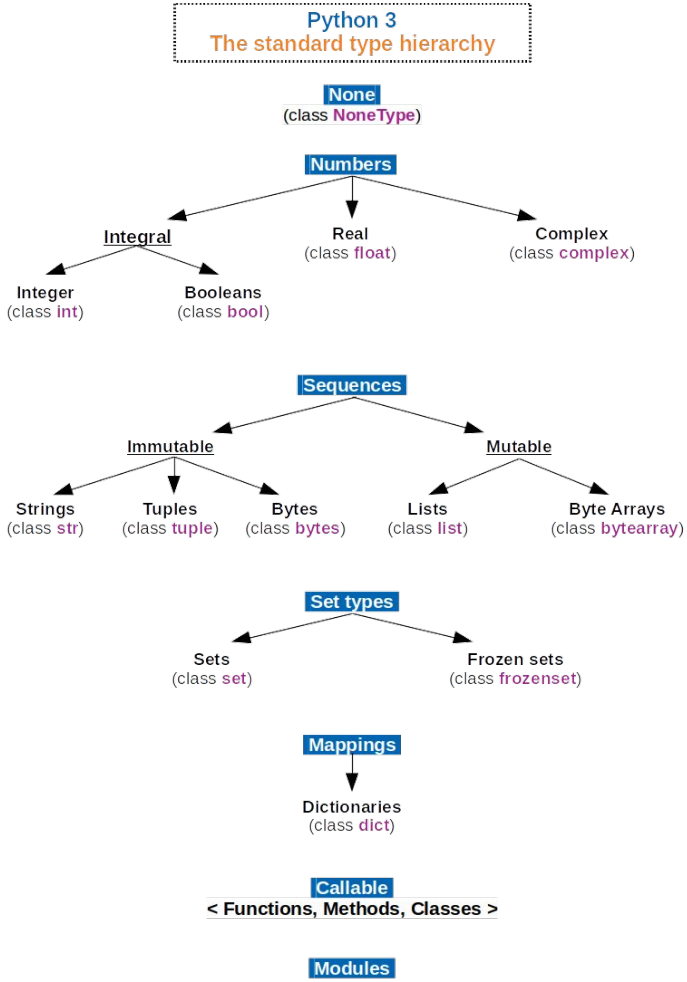
```
>>> x = 2 * 3
>>> print(x * 2)

OUTPUT: 12
```

Python knows `x` is an **int**

Python types

There are a bunch of different Python “types”!

- Numbers
  - Integer (e.g., 2), Float (e.g., 3.14)
- Sequences
  - Strings (e.g., “hello world”)
- Etc.



Python types

Name	Data Type	Description
Integer	int	a number that can be written without fractions: 22 10 0 -300
Floating-point	float	a number that has a decimal component: 3.14 2.73 10.0
String	str	sequence of characters: "hello world" "hey88340" '2018'

For today, we'll focus on:

- **Integers** (*int*)
  - Whole numbers
- **Floating-points** (*float*)
  - Decimal numbers
- **Strings** (*str*)
  - Text surrounded by quotes: "..." or '...'

Python types

We will work our way through the different data types throughout our first week together.

Name	Data Type	Description
Integer	int	a number that can be written without fractions: 22 10 0 -300
Boolean	bool	logical value that indicates **True** or **False**
Floating-point	float	a number that has a decimal component: 3.14 2.73 10.0
String	str	sequence of characters: "hello world" "hey88340" '2018'
List	list	sequence of comma-separated numbers, strings etc.: [10, '2018', "hi"]
Dictionary	dict	collection of key-value pairs: {"key1": "value1", "key2": "value2"}
Tuple	tuple	sequence of comma-separated numbers, strings etc.: (10, 20.0, "world", 5)

As I said at the start of this lecture, there is a lot to learn right at the beginning of coding. Throughout the remainder of this program, please feel free to stop me if you want more clarification on anything in particular. Questions?

LIVE CODING

Manipulating **Numbers** & **Strings** in Python