NUMBERS AND MATH

What an exciting combination!

Objectives

- Understand the differences between ints and floats
- Work with simple (and some weird) mathematical operators
- Add comments to your code (completely unrelated, but it has to go somewhere!)

Numbers and Math

Two main types of numbers:

Integer	Floating Point
4	6.1
57	1.3333
-10	0.0

PYTHON ISN'T JUST A GLORIFIED CALCULATOR

(But you can use it as one)

```
1 + 1
```

```
5 - 3
```

getting tricky!

```
1 + 4 - 2
```

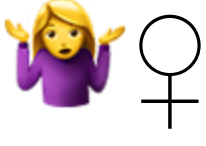
Numbers and Math

commonly used math operators

Symbol	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation
%	Modulo
//	Integer Division

WHAT

WHAT



QUIZ TIME

It's super quick, I promise

COMMENTS

NEXT UP:
VARIABLES

VARIABLES AND DATA TYPES

(mostly strings for now)

Objectives

- Understand how to assign and use variables
- Learn Python variable naming restrictions and conventions
- Learn and use some of the different data types available in Python
- Learn why Python is called a dynamically-typed language
- Understand how to convert data types
- Learn the ins and outs of Strings!
- Build a silly little program that gets user input

VARIABLE ASSIGNMENT

Variables must be assigned before they can be used.

A variable in Python is like a variable in mathematics: it is a named symbol that holds a value.

```
x = 100
khaleesi_mother_of_dragons = 1
print(khaleesi_mother_of_dragons + x)
101
```

Variables are always **assigned** with the variable name on the left and the value on the right of the equals sign.

VARIABLE ASSIGNMENT

(continued)

Variables can be:

- assigned to other variables
- reassigned at any time
- assigned at the same time as other variables

```
python_is_awesome = 100
```

```
just_another_var = python_is_awesome
python_is_awesome = 1337
all, at, once = 5, 10, 15
```

Naming Restrictions

In Python, you can name your variables whatever you want, with some restrictions:

1. Variables must start with a letter or underscore

`_cats` ✅ `2cats` ❌

2. The rest of the name must consist of letters, numbers, or underscores

`cats2` ✅ `hey@you` ❌

3. Names are case-sensitive

`CATS != Cats` `Cats != cats`

Naming Conventions

Most Python programmers prefer to use standard style conventions when naming things:

- Most variables should be **snake_case** (underscores between words)
- Most variables should be also be **lowercase**, with some exceptions:

  CAPITAL_SNAKE_CASE usually refers to constants (e.g. PI = 3.14)

  UpperCamelCase usually refers to a class (more on that later)
- Variables that start and end with two underscores (called "dunder" for **d**ouble **under**score) are supposed to be private or left alone

```
__no_touchy__
```

Data Types

In any assignment, the assigned value must always be a valid data type.

Python data types include:

data type	**description**
bool	True or False values
int	an integer (1, 2, 3)
str	(string) a sequence of Unicode characters, e.g. "Colt" or "程序设计"
list	an ordered sequence of values of other data types, e.g. [1, 2, 3] or ["a", "b", "c"]
dict	a collection of key: values, e.g. { "first_name": "Colt", "last_name": "Steele" }

Among others!

Dynamic Typing

Python is highly flexible about reassigning variables to different types:

```
awesomeness = True
print(awesomeness) # True

awesomeness = "a dog"
print(awesomeness) # a dog

awesomeness = None
print(awesomeness) # None

awesomeness = 22 / 7
print(awesomeness) # 3.142857142857143
```

We call this **dynamic typing**, since variables can change types readily.

Other languages, such as C++, are **statically-typed**, and variables are stuck with their originally-assigned type:

```
int not_awesomeness = 5;
not_awesomeness = "cool"; // !Error
```

Declaring Strings

String literals in Python can be declared with either single or double quotes.

```
my_other_str = 'a hat'
my_str = "a cat"
```

Either one is perfectly fine; but make sure you stick to the same convention throughout the same file.

String Escape Characters

In Python there are also "escape characters", which are "metacharacters" - they get interpreted by Python to do something special:

```
new_line = "hello \n world"

print(new_line)
# hello
# world
```

All escape characters start with a backslash "\".

You can do advanced things like include hexadecimal characters with "\x"

```
hexadecimal = "\x41\x42\x43"  # "abc"
```

String Concatenation

Concatenation is combining multiple strings together. In Python you can do this simply with the "+" operator.

```
str_one = "your"
str_two = "face"
str_three = str_one + " " + str_two  # your face
```

You can also use the "+=" operator!

```
str_one = "ice"
str_one += " cream"
str_one  # ice cream
```

Formatting Strings

There are also several ways to format strings in Python to **interpolate** variables.

The new way (new in Python 3.6+) => **F-Strings**

```
x = 10
formatted = f"I've told you {x} times already!"
```

The tried-and-true way (Python 2 -> 3.5) => **.format method**

```
x = 10
formatted = "I've told you {} times already!".format(x)
```

The old way => **% operator** (deprecated)

```
x = 10
formatted = "I've told you %d times already!" % (x)
```

Converting Data Types

In string interpolation, data types are implicitly converted into string form (more on this later in OOP).

You can also explicitly convert variables by using the name of the builtin type as a function (more on functions later):

```
decimal = 12.56345634534
integer = int(decimal)  # 12
```

```
my_list = [1, 2, 3]
my_list_as_a_string = str(my_list)  # "[1, 2, 3]"
```

MILEAGE CONVERTOR

converter?

CODE-ALONG

Recap

- Python is a dynamically-typed language with more than half a dozen types
- Python lets you assign and reassign variables at will to any of the types
- There are some variable naming restrictions which are required and some naming guidelines which are recommended

YOUR TURN