PROGRAMMING IN PYTHON

A BEGINNER WORKSHOP

Welcome!

This workshop is an introduction to the popular coding language **Python**. It is designed for those who have **little to no programming experience**. The workshop is broken up into **4 sections**. Each section contains multiple exercises, as well as a project at the end. We will be covering topics including variables, conditionals, loops, functions, data structures, file I/O, error handling, and object oriented programming. I hope you enjoy the workshop; have fun coding!

By the end of this workshop, you will:

- Understand the fundamentals of computer science
- Be able to write, run, and debug Python code
- Utilise different data structures and variable types
- Read and write files using Python
- Create games and solve problems using your own Python programs

Before the workshop:

- ▷ Download and install the latest version of Python
- ▷ Check to see if the IDLE editor is installed (should have been downloaded with Python)
- ▷ Download Zoom (if participating in the live classes)
- ▷ Download the starter code files from the GitHub repo
- ▷ Have a place to take notes (laptop, notebook, etc.)

Course Outline

Section 1

- Variables
- Types
- User Input
- **Project**: Mad Libs

Section 2

- Operators
- Conditionals
- Loops
- Functions
- **Project**: Guess the Number

Section 3

- Lists
- Dictionaries
- Error Handling
- File I/O
- **Project**: Message Encryptor

Section 4

- Classes
- Objects
- **Project**:

Meet your Instructor

Hi! My name is Brittney. A few fun facts about me:

- I am a CS major at Colgate University
- I'm from Long Island, NY
- I've coded with Girls Who Code and Amazon
- I know the first 30 digits of pi by heart

Section 1

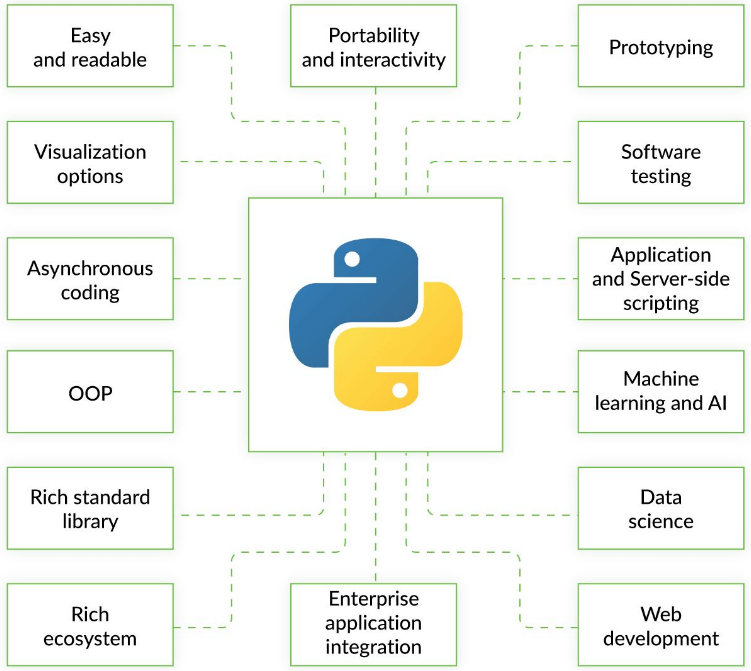
Language Overview & Basics

What is Python?

Python is an object-oriented, high-level programming language. **High-level languages** are designed to be human readable, and code is converted to a low-level language behind the scenes for the computer to process. It’s simple and readable format makes it a great language for beginner coders.

What can you do with Python?

- ▷ Game design
- ▷ Web development
- ▷ Automate tasks
- ▷ Data analysis
- ▷ Machine learning
- ▷ And lot's more!



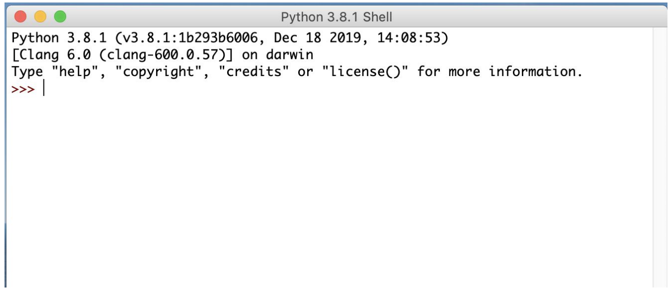
How is it different from other languages?

Python is **dynamically-typed**, which means that variable types can change over the course of a program.

Python is also different from other languages in terms of its **syntax** - the order and structure of the code.

What is IDLE?

IDLE is the Python IDE we will be using for this workshop. An **IDE** (Integrated Development Environment) allows us to write and run code in the same application. We could also choose to write and run code from the terminal.



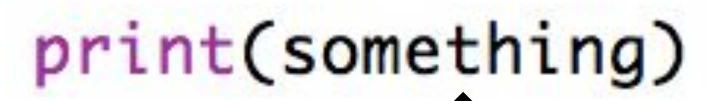
```
Python 3.8.1 (v3.8.1:1b293b6006, Dec 18 2019, 14:08:53)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

The IDLE shell - used to test snippets of code.

Statements

A **statement** is a command given to the computer to perform a specific action.

An example of this is the **print()** statement, which tells the computer to print something to the screen/console.



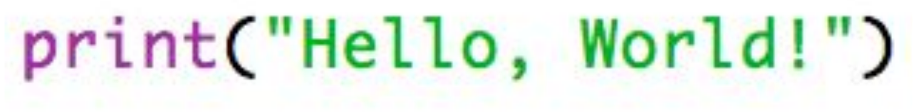
```
print(something)
```

parameter

(don’t really worry about this right now)

Hello World!

Now to write your first line of code! Open your IDE (IDLE) and click “File”, “New File”. Then type the following:


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

Click “Save As...” then save your file as “HelloWorld.py”. Now click “Run”, the “Run Module”.

Congrats! You are now a computer programmer!

```
Hello, World!
```

What is an Algorithm?

An **algorithm** is a list of instructions meant to solve a given problem. Think of it as a recipe for the computer to follow. An algorithm is made up of multiple **statements**.

```
Get ingredients
    Get knife
    Get bread
    Get peanut butter
    Get jelly
Spread peanut butter on one slice
    Open peanut butter jar
    Dip knife in jar
    Remove knife with peanut butter on it
    Place knife butter side down on bread
    Spread on bread with knife
Spread jelly on another slice
Place two slices together
```

Important Notes on Python Syntax

- All Python files will use the file extension “.py”
- In Python, **indentation matters**! In other languages, symbols such as semi-colons distinguish when a line of code ends
- To add a **comment** (non-runnable statement) in Python, use the “#” symbol
- Python supports **single** and **double quotes** for Strings

Variables

A **variable** is a container that stores a piece of information to be used later. We define a variable using the following syntax:

variable name = value

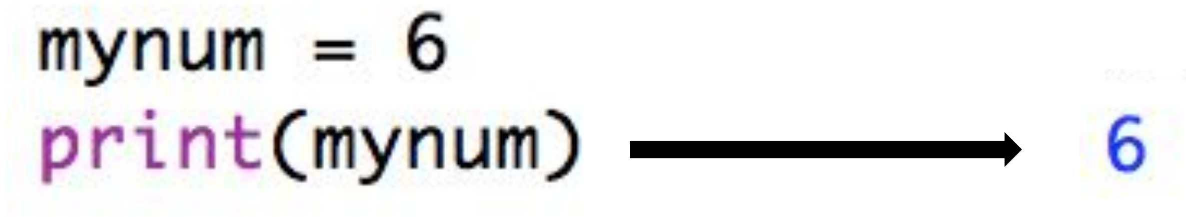
always on the left

always on the right

We can access a variable’s value by referring to its name. You can think of a variable name like a label on a box.

Variables

Example:



```python
mynum = 6
print(mynum)
```

```
6
```

Reference vs. Object

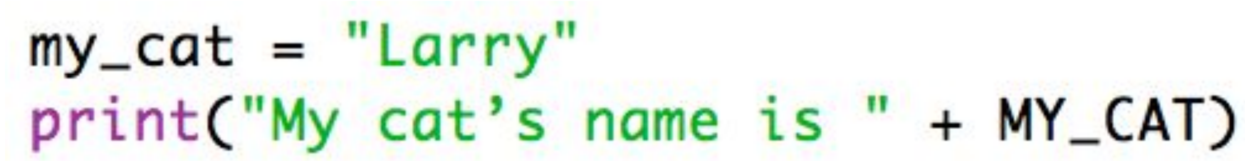
In Python, when we set a variable equal to a value, the variable name is **referring to** the item on the right. Therefore, the name is a **reference**, and the item is an **object**. We will learn more about objects during section 4.

Naming variables

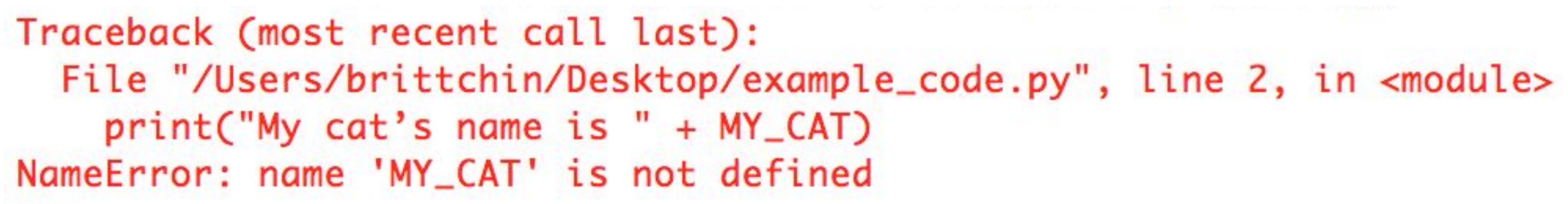
- Variables must start with a **letter or underscore**
- There are **no spaces allowed** - use underscores or camelCase
- Cannot contain **symbols** ($@&) or **operators** (+=-)
- You cannot use **keywords** as variable names (ex: class, True, for)
- Variables are **Case Sensitive** - good practice to use lowercase for all naming!

Always use **meaningful** variable names - names that make sense for what you are storing

What will happen when you run this code?



```
my_cat = "Larry"
print("My cat’s name is " + MY_CAT)
```

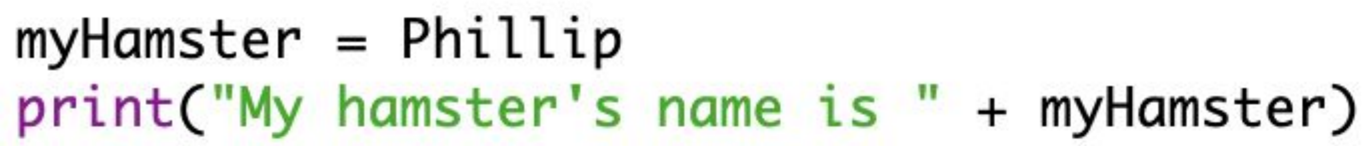


```
Traceback (most recent call last):
  File "/Users/brittchin/Desktop/example_code.py", line 2, in <module>
    print("My cat’s name is " + MY_CAT)
NameError: name 'MY_CAT' is not defined
```

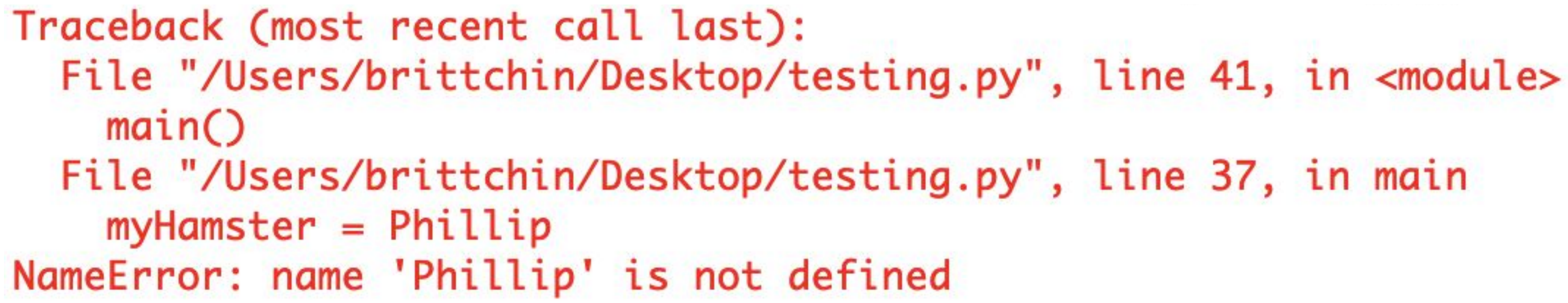
Error Messages

Getting errors is a normal part of computer programming. We will talk more about different types of errors and error handling later on in this workshop. When you get an error message, it will often tell you exactly **what and where** the problem is. We use these error messages to **debug**, identify and remove error messages from, our code.

What will happen when you run this code?



```
myHamster = Phillip
print("My hamster's name is " + myHamster)
```



```
Traceback (most recent call last):
  File "/Users/brittchin/Desktop/testing.py", line 41, in <module>
    main()
  File "/Users/brittchin/Desktop/testing.py", line 37, in main
    myHamster = Phillip
NameError: name 'Phillip' is not defined
```

Types

A **type** is a classification that tells the compiler **how** the programmer intends to use the data. They define what kind of operations can be applied to the variable without causing an error. Some examples of types include the following:

String - a sequence of characters enclosed in quotation marks (ex: “hello”, “I like coding!”, “1234”)

Integer - negative or positive whole numbers (ex: 42, 7, -2)

Float - numbers with a decimal (ex: 4.25, 100.00)

Boolean - a True or False value

What will happen when you run this code?

```python
name = "Anna"
num1 = 110
num2 = 40.0

print(type(name))
print(type(num1))
print(type(num2))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

NOTE: You can check the type of a variable by using the **type()** function.

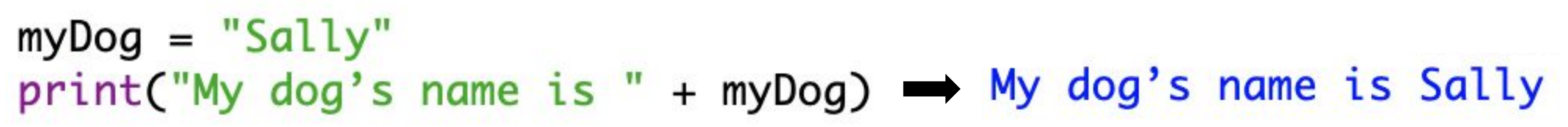
Variable Practice

In the starter code folder, open the file **variable_practice.py** in the **section 1** folder. Follow the instructions for creating, modifying, and printing variables.

Combining Strings

Strings can be combined using the "+" sign. This is called **string concatenation**. When concatenating strings, be sure to add spaces, or the words will be connected.

Example:

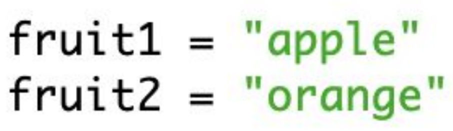


```
myDog = "Sally"
print("My dog’s name is " + myDog) ➡ My dog’s name is Sally
```

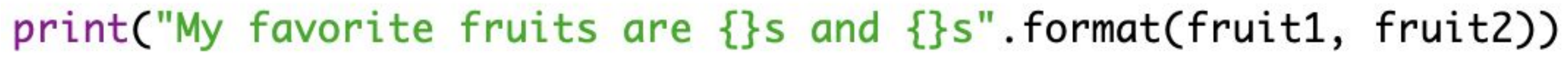
String Formatting

Concatenating a lot of strings with "+" can be messy. You can also format strings by substituting variables into **curly braces {}** and calling the method **.format()** on the string.

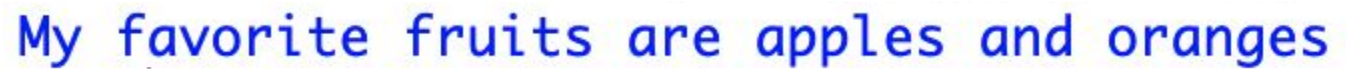
Example:



```
fruit1 = "apple"
fruit2 = "orange"
```



```
print("My favorite fruits are {}s and {}s".format(fruit1, fruit2))
```



```
My favorite fruits are apples and oranges
```

Concatenation Rules

Be careful when concatenating variables of **different types**! When concatenating a string to a variable of a different type, use a **comma** instead of “+”. This is not a problem when using .format(). Commas will automatically add spaces between words.

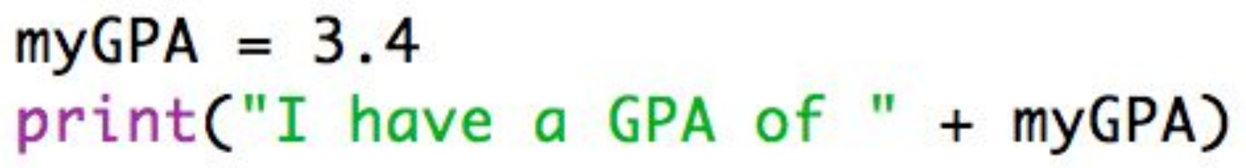
Example:

```
age = 20
print("I am", age, "years old.")
```

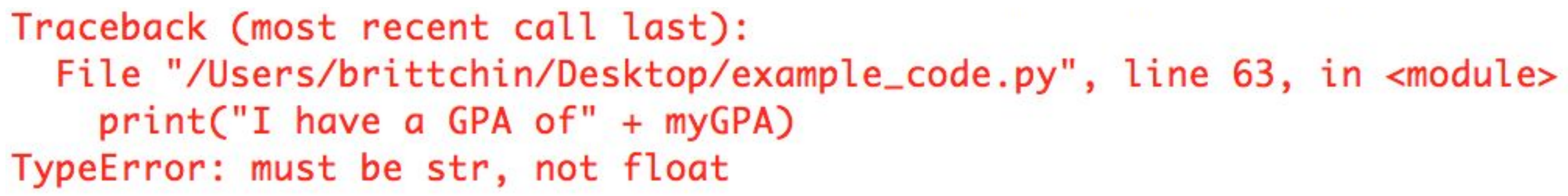
$\longrightarrow$

```
I am 20 years old.
```

What will happen when you run this code?



```python
myGPA = 3.4
print("I have a GPA of " + myGPA)
```



```
Traceback (most recent call last):
  File "/Users/brittchin/Desktop/example_code.py", line 63, in <module>
    print("I have a GPA of" + myGPA)
TypeError: must be str, not float
```

String Indexing

Strings are made up of individual **characters**. These characters could be letters, numbers, whitespace, or other symbols. We can access individual characters of a through by **indexing**.

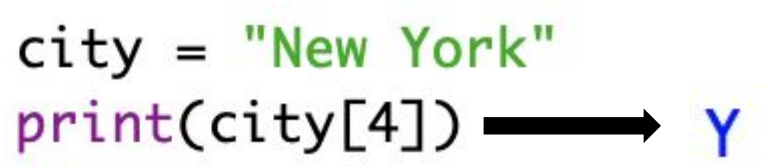
Each character is given a corresponding number or **index**. In computer science, we **always start counting at 0**.

H	**e**	**l**	**l**	**o**		**W**	**o**	**r**	**l**	**d**	**!**
0	1	2	3	4	5	6	7	8	9	10	11

String Indexing

We can reference a specific character in a string using **square brackets []**. When we refer to an index of a string, it will return the character at the position corresponding to the index.

Example:



```
city = "New York"
print(city[4])
```

$\longrightarrow$

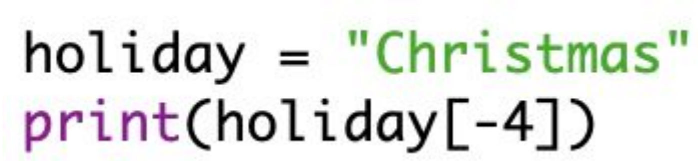
```
Y
```

String Indexing

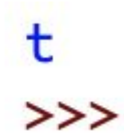
Python also supports negative indexing. We can also count backwards from the end of the string, starting at **-1**.

H	**e**	**l**	**l**	**o**		**W**	**o**	**r**	**l**	**d**	**!**
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

What will happen when you run this code?

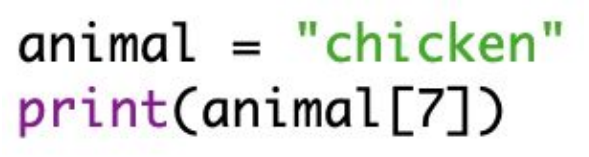


```python
holiday = "Christmas"
print(holiday[-4])
```

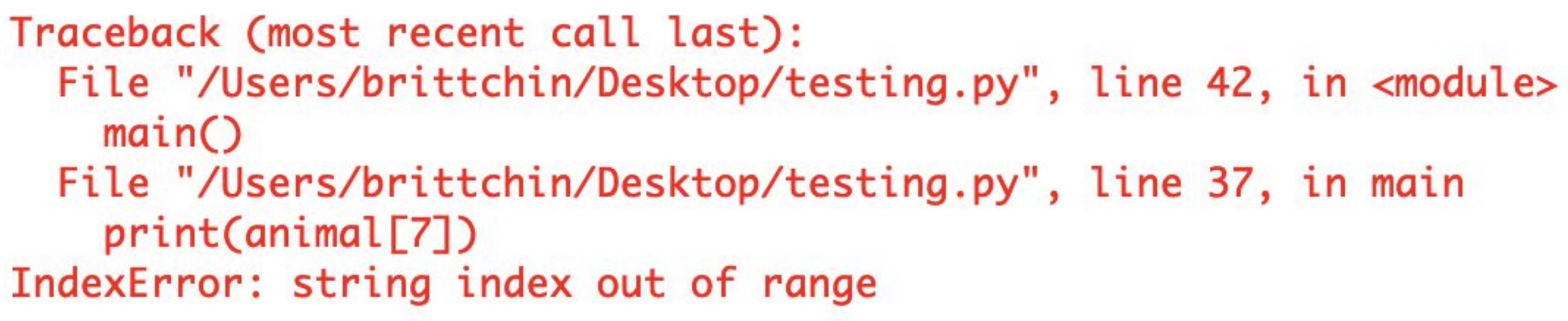


```
t
>>>
```

What will happen when you run this code?



```
animal = "chicken"
print(animal[7])
```



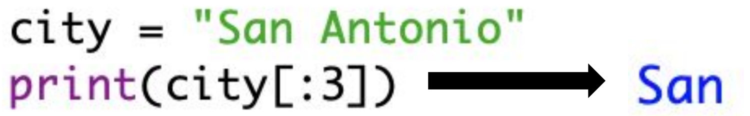
```
Traceback (most recent call last):
  File "/Users/brittchin/Desktop/testing.py", line 42, in <module>
    main()
  File "/Users/brittchin/Desktop/testing.py", line 37, in main
    print(animal[7])
IndexError: string index out of range
```

String Slicing

We can also obtain a range of characters from a string using **slicing**. A **slice** is a sequence of characters within a string.

We can get multiple characters from a string by **creating a range** separated by a colon [x:y], where x is the starting index (inclusive) and y is the ending index (exclusive). If no start is provided, it will start at the beginning of the string. If no end is provided, it will end at the end of the string.

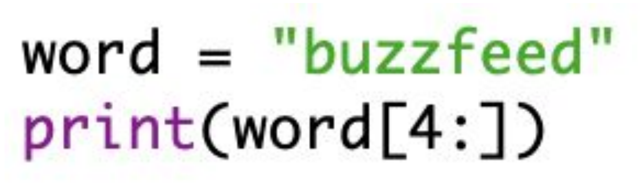
Example:



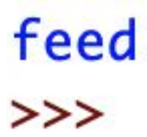
```
city = "San Antonio"
print(city[:3])
```

```
San
```

Fill in the Blank

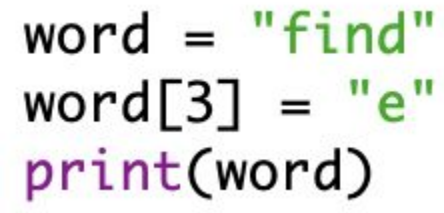


```
word = "buzzfeed"
print(word[4:])
```

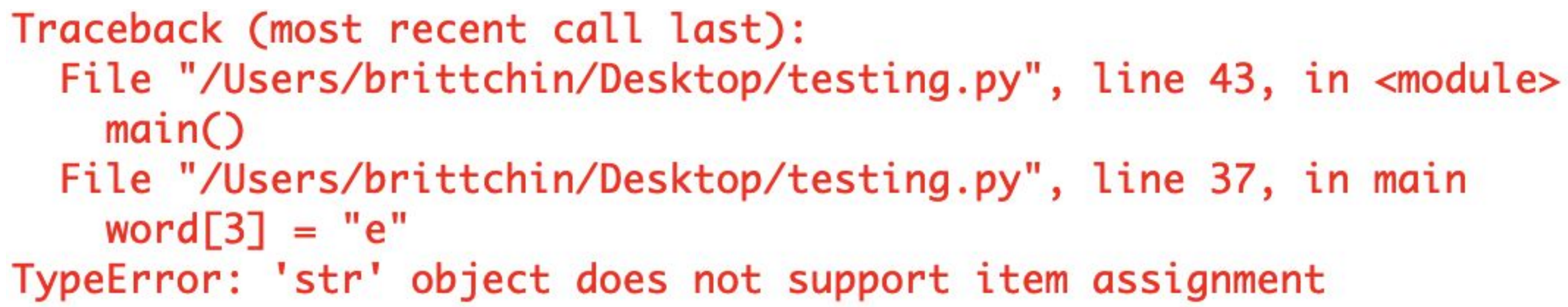


```
feed
>>>
```

What will happen when you run this code?



```python
word = "find"
word[3] = "e"
print(word)
```



```
Traceback (most recent call last):
  File "/Users/brittchin/Desktop/testing.py", line 43, in <module>
    main()
  File "/Users/brittchin/Desktop/testing.py", line 37, in main
    word[3] = "e"
TypeError: 'str' object does not support item assignment
```

String are **immutable**, which means they cannot be modified directly after creation. Instead, you must use methods such as slicing and re-assign it to the variable.

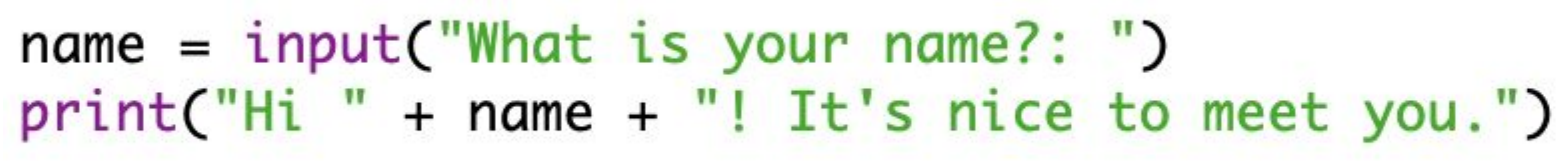
Storing User Input

User input allows a user to interact with your program by typing something into the console. In Python, we do this by using the **input()** function, which takes a **String** parameter that will be displayed to the user.

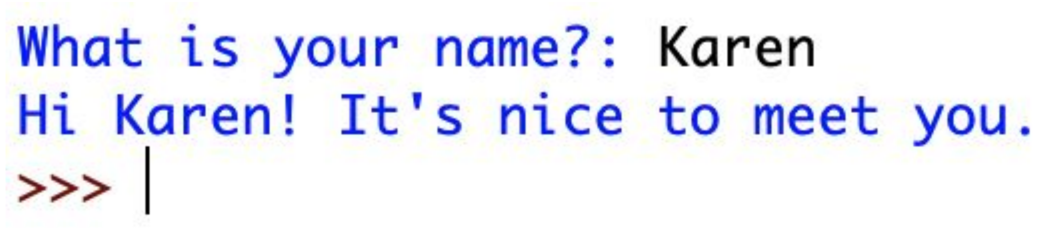
Everything returned by the input function is of **type String**, regardless of what the user inputted!

User Input

Example:

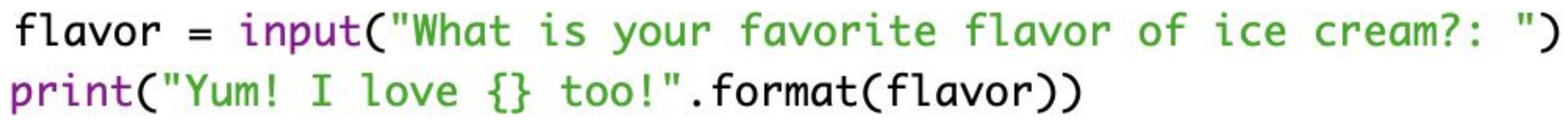


```python
name = input("What is your name?: ")
print("Hi " + name + "! It's nice to meet you.")
```

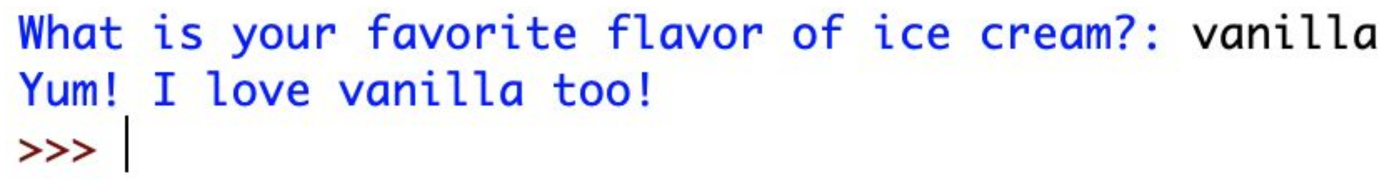


```
What is your name?: Karen
Hi Karen! It's nice to meet you.
>>> |
```

What will happen when you run this code?

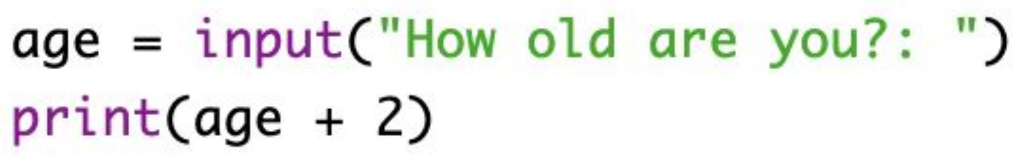


```python
flavor = input("What is your favorite flavor of ice cream?: ")
print("Yum! I love {} too!".format(flavor))
```

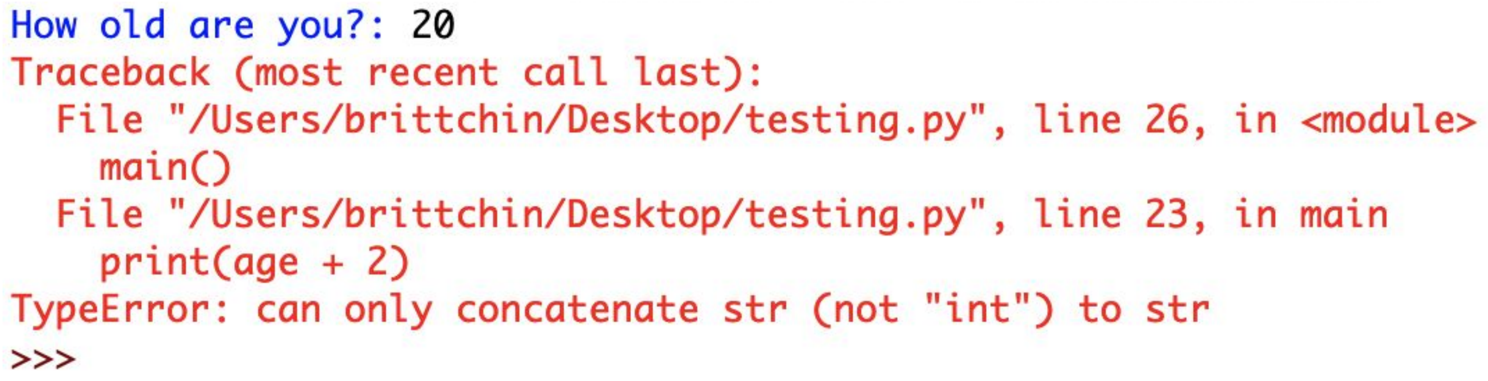


```
What is your favorite flavor of ice cream?: vanilla
Yum! I love vanilla too!
>>> |
```

What will happen when you run this code?



```
age = input("How old are you?: ")
print(age + 2)
```

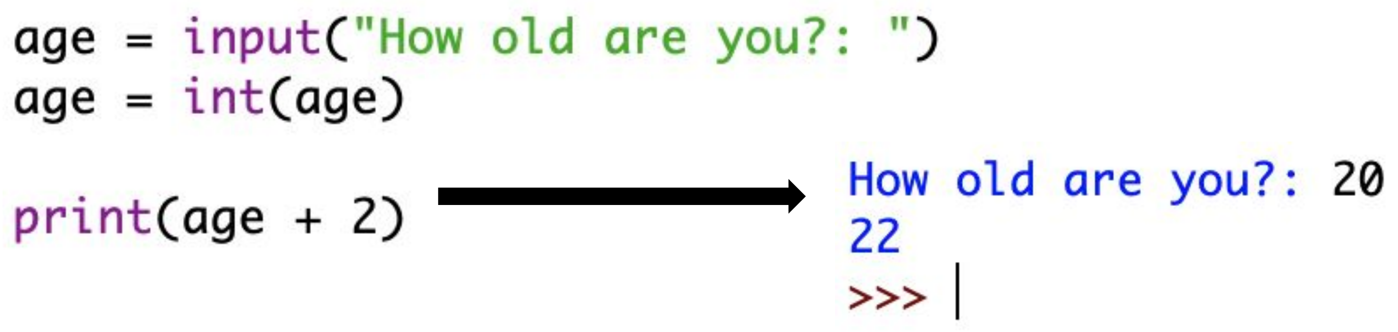


```
How old are you?: 20
Traceback (most recent call last):
  File "/Users/brittchin/Desktop/testing.py", line 26, in <module>
    main()
  File "/Users/brittchin/Desktop/testing.py", line 23, in main
    print(age + 2)
TypeError: can only concatenate str (not "int") to str
>>>
```

Typecasting

We can solve this problem using **typecasting**! Typecasting converts the type of a variable, allowing you to perform other actions on it. The main casting functions in Python are **str()**, **int()**, and **float()**.

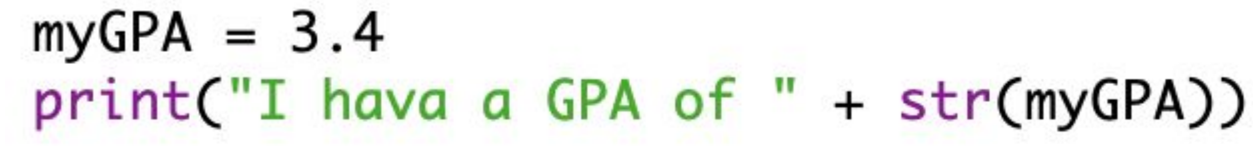
Example:



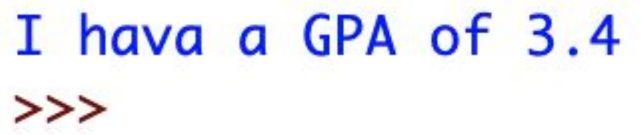
```python
age = input("How old are you?: ")
age = int(age)

print(age + 2)
```

```
How old are you?: 20
22
>>> |
```

The Example from Earlier

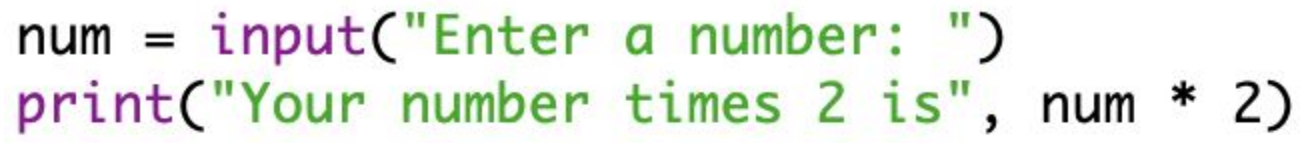


```
myGPA = 3.4
print("I hava a GPA of " + str(myGPA))
```

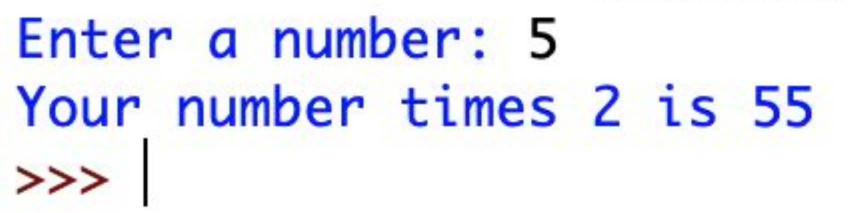


```
I hava a GPA of 3.4
>>>
```

What will happen when you run this code?

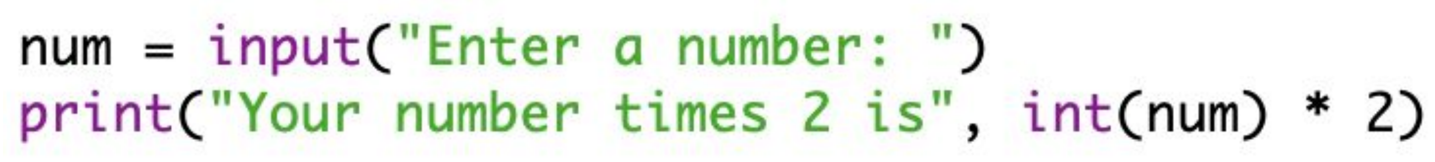


```
num = input("Enter a number: ")
print("Your number times 2 is", num * 2)
```

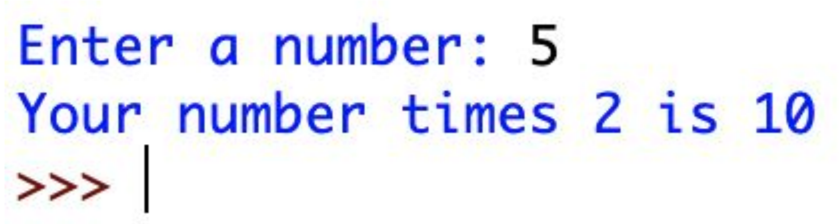


```
Enter a number: 5
Your number times 2 is 55
>>> 
```

Code updated



```python
num = input("Enter a number: ")
print("Your number times 2 is", int(num) * 2)
```



```
Enter a number: 5
Your number times 2 is 10
>>> 
```

Section 1 Summary

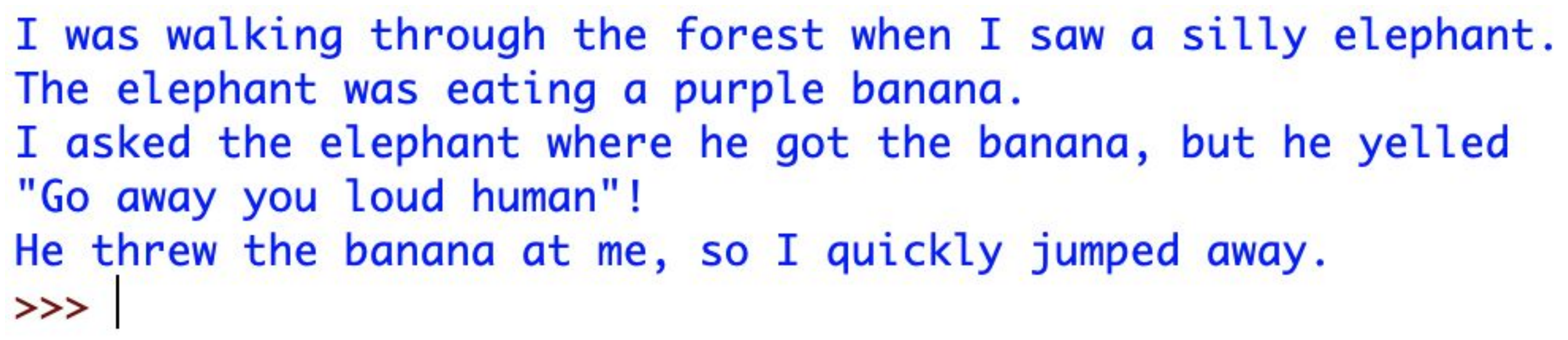
- Computer programs consist of sets of instructions for the computer to interpret and follow
- **Variables** store information for use in a program
- Data **types** determine what actions can be performed on a variable
- Types can be converted for manipulation using the Python **int()**, **float()**, and **str()** casting functions
- You can get information from the user using the **input()** function

SECTION 1 PROJECT

Mad Libs

Mad Libs

For this activity, you will create a **Mad Libs** story, where the user will input different words and a story is created based on the words inputted. The starter code for this project is in **mad_libs_starter.py**.



```
I was walking through the forest when I saw a silly elephant.
The elephant was eating a purple banana.
I asked the elephant where he got the banana, but he yelled
"Go away you loud human"!
He threw the banana at me, so I quickly jumped away.
>>> 
```

Mad Libs

Concepts You Will Use:

- Variables
- Print Statements
- User Input