Power of Python

Cybersecurity Boot Camp

Lesson 3: Day 1

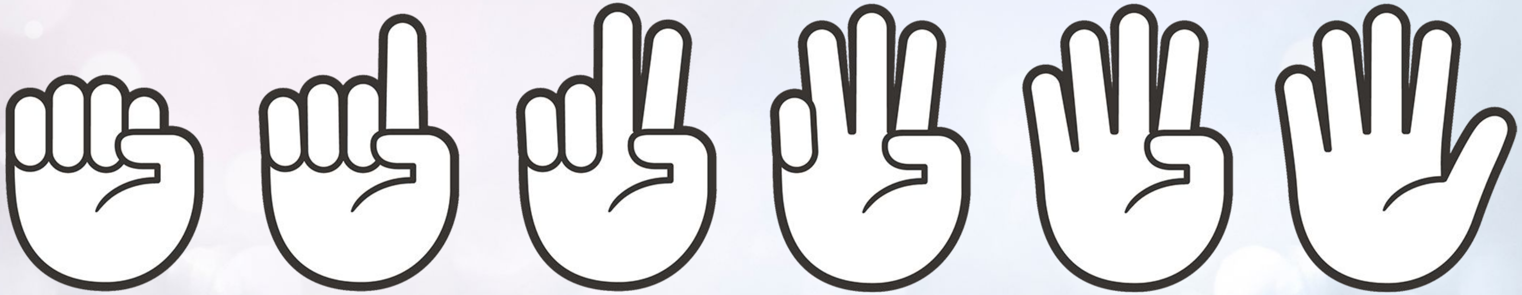
This Week: Python

This week, we will expand our toolset by introducing the Python programming language.

Class Objectives

By the end of class today, you will be able to:

- ☐ Explain how and why Python is used in cybersecurity.
- ☐ Create and run Python files via the terminal using VS Code.
- ☐ Use the `print()` function to print lines to the console.
- ☐ Use basic Python elements like variables and operators.
- ☐ Employ the Python `input()` function to retrieve, store, and utilize user inputs.
- ☐ Reference and store collections of data using lists.
- ☐ Create and reference data in dictionaries.
- ☐ Use `listVariable.append(Value)`, `listVariable.index(Value)`, and `listVariable.remove(Value)` to add, return, and delete values from a list.
- ☐ Use `len(listVariable)` function to return the length of a list.



FIST TO FIVE:

Raise a Fist

If you've *never* worked with Python and barely know what it is.

Raise a Five

If you work with Python on a daily basis.

Raise a One, Two, Three, or Four

If you fall somewhere in between.

What Is Python?

What Is Python?

Python is a high-level, general purpose programming language used for a variety of applications.

- It has an enormous community of developers.
- It is used in a wide variety of industries and jobs.
- Prioritizes readability for ease of use.

Why Python?

It is the preferred choice over other programming languages on the market for the following 3 reasons:

01

Python is a high-level, general purpose programming language used for a variety of applications. It has an enormous community of developers with backgrounds in a wide variety of industries and jobs.

02

The syntax makes it incredibly readable, making it an excellent introductory programming language, while still being immensely powerful. Python forces you to follow certain rules to make the code readable. Well written code makes it easier for everyone to read, which makes team collaboration easier.

03

Learning Python is a huge competitive advantage for technically oriented cybersecurity jobs.

Why Python?

Learning Python is a huge competitive advantage for technically oriented cybersecurity jobs.

Skill	Number of Job Listings
Information Security	151,470
Linux	73,735
Project Management	69,072
Network Security	63,519
Cryptography	53,997
Network Engineering	47,651
SQL	43,651
Unix	40,945
Python	40,750

Based on national job listings in cybersecurity (Source: Burning Glass)

Bash vs. Python

Bash vs. Python

In reality, we'll use both Python and Bash.

Bash/Shell Scripts/Unix are used for OS-level interaction, dealing with processes, and interacting with the file system.

Python is preferred for simple or complex logic-based applications, and for automating complex tasks.

Activity: Install Python 3.x

We will be installing the Python programming language.

Macs come with Python installed and most likely it will be the 2.x version.
For this class you need 3.x because Python 2.x has different syntax and the code.

Windows users will have to install Python if they haven't already installed it.

Instructions sent via Slack

Suggested Time:
15 minutes

Your Turn: Install Python 3.x

Instructions:

Check to see if you have Python installed. Follow these steps:

1. Open a terminal (or Git Bash) window.

2. Then enter the command: `python -version` (Windows) or `python3 -version` (Mac).

3. If `Python 3.x.x` is returned then you have the correction version installed. (If you see the phrase Python 2.x.x (or receive an error), install Python using the steps below).

To install Python, follow these steps:

1. Navigate to the Python website and download the latest version. (https://www.python.org/downloads/)

2. Complete the default installation steps. **Important:** Windows users must select the **Add Python 3.x to PATH** option from the Install Python dialogue.

3. Check that you have the latest version of Python installed again by opening a terminal or Git Bash window and entering the `python --version` (Windows) or `python3 --version` (Mac) command.

VS Code

Python is the programming language.

VS Code is the editor in which we’ll write the code.

Activity: Visual Studio Code

Now that you have the the current version of Python installed, you will then need to install Visual Studio Code. VS Code is the editor we will use to write our code.

Instructions sent via Slack

Suggested Time:
5 minutes

Your Turn: Install Visual Studio Code

Instructions:

01 Go to code.visualstudio.com to download and install the latest version of Visual Studio Code.

02 When the installation is complete, open VS Code and click the Extensions icon in the Activity bar on the side of VS Code.

03 In the search bar, type Python and download the first extension that appears.

Suggested Time: 5 minutes

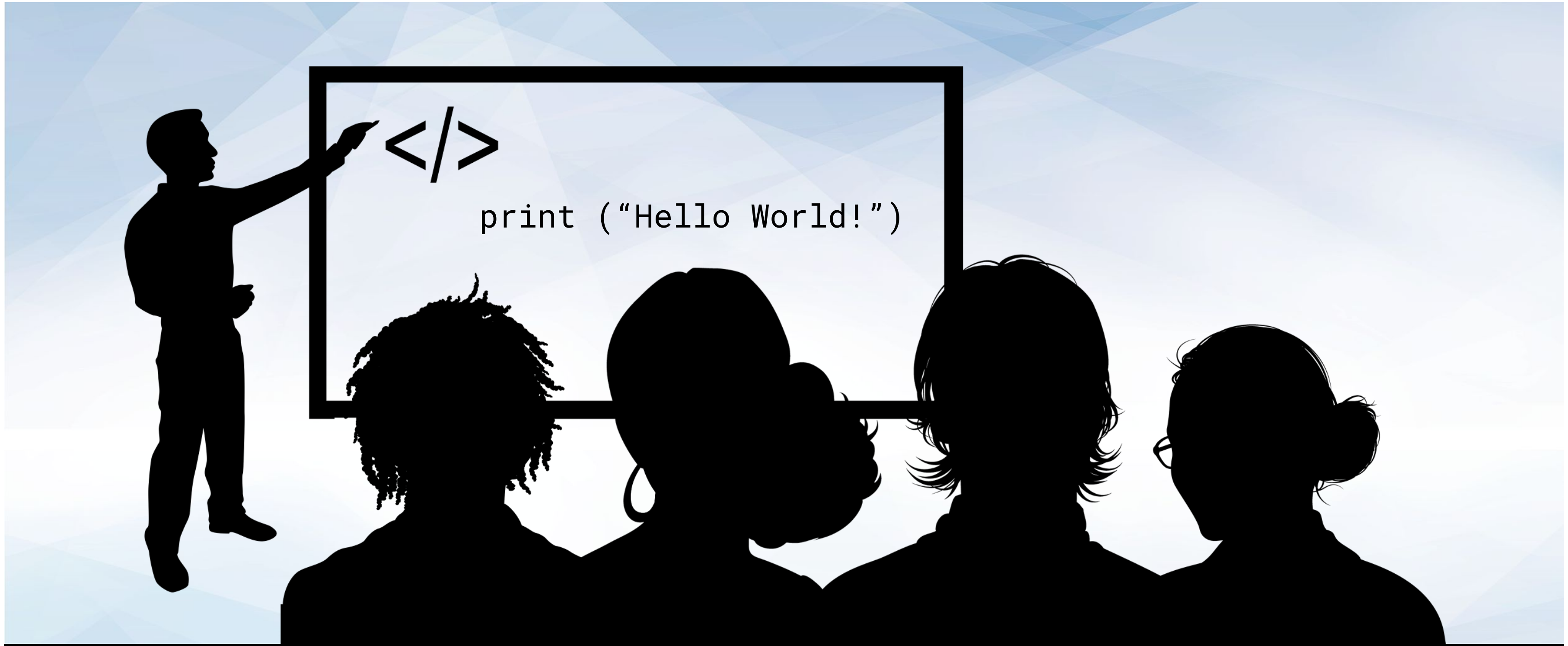
Your Turn: Install Visual Studio Code

Instructions:

1. Go to code.visualstudio.com to download and install the latest version of Visual Studio Code.

2. When the installation is complete, open VS Code and click the Extensions icon in the Activity bar on the side of VS Code.

3. In the search bar, type Python and download the first extension that appears.



Instructor Demonstration

Hello World!

Activity: Goodnight World!

Now that you have seen how to create and run a Python script, you will now make a program of your own and practice running it from within the terminal.

Suggested Time:
7 minutes

Activity: Goodnight World!

Instructions:

1. Create a new file in VS Code and save it as `GoodnightWorld.py`
2. Write a line of code that will print out the line "Goodnight World!" to the terminal when it is run.
3. Go to the folder where you saved `GoodnightWorld.py` and run the application.

Tips:

- The `print()` function allows you to print lines to the console.
- Remember to put the phrase in quotes so that it prints properly.

Programming Fundamentals:
`Variables and Data Types`

Programming Fundamentals

Programming is just piecing logic together to solve larger problems and produce output. All of programming is basically just manipulating data.

Data	**Logic**
1. Numbers	1. Operators
2. Strings	2. Conditionals
3. Booleans	3. Loops
4. Lists	4. Functions
5. Dictionaries	5. Modules

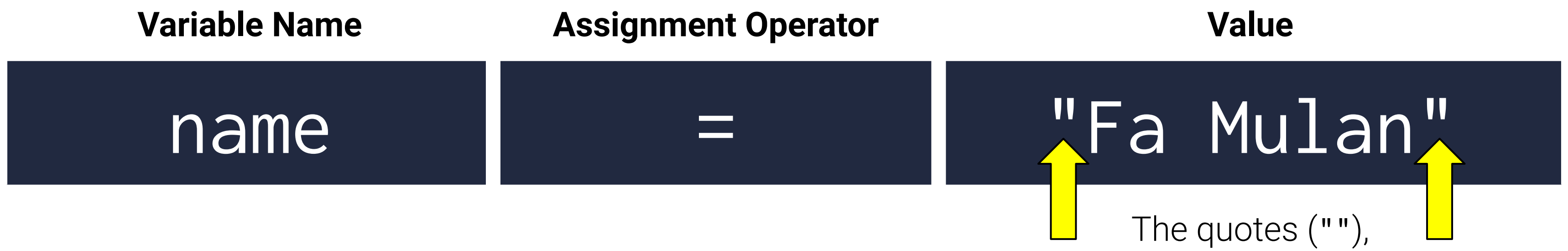
Variables

Variables are the holding ground for data. They essentially allow us to attach **specific values** to **keywords** for use later on in an application.

Variables

Variables are the means by which we can store and reference data.
In Python, we assign values to variables.

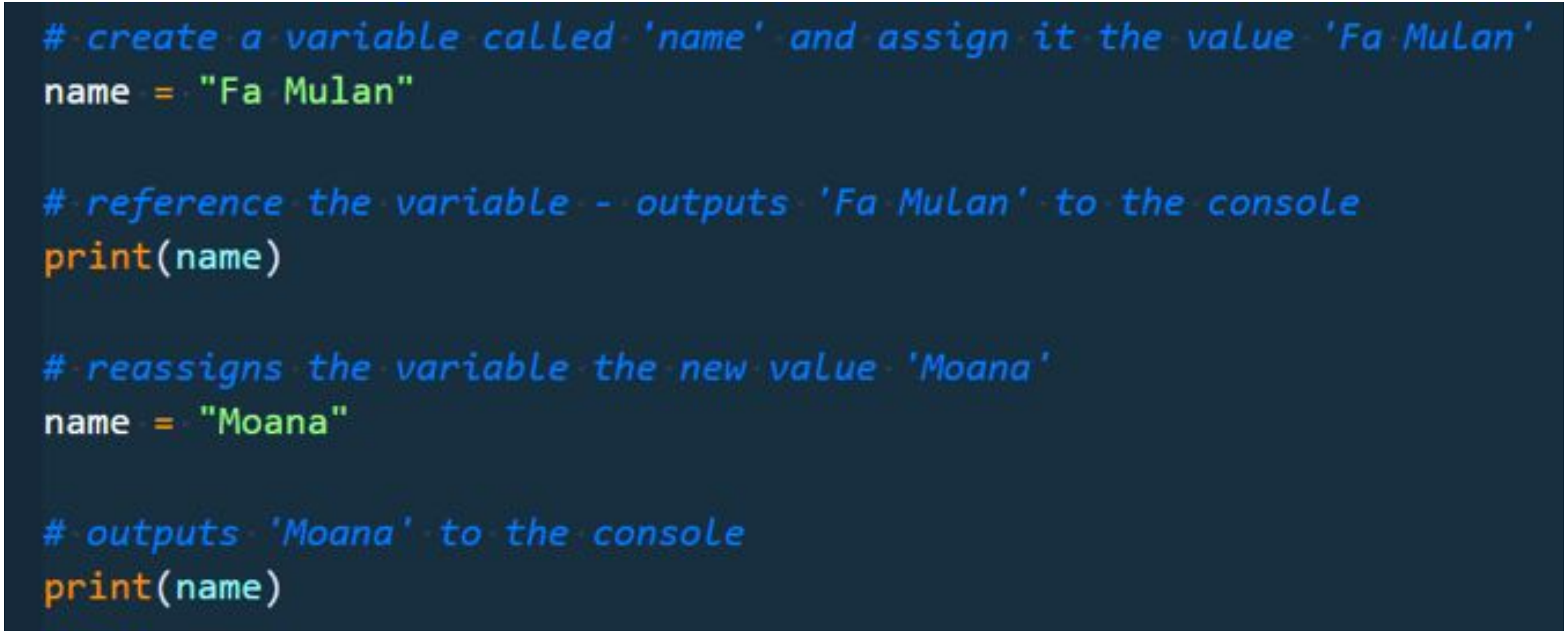
The assignment statement:



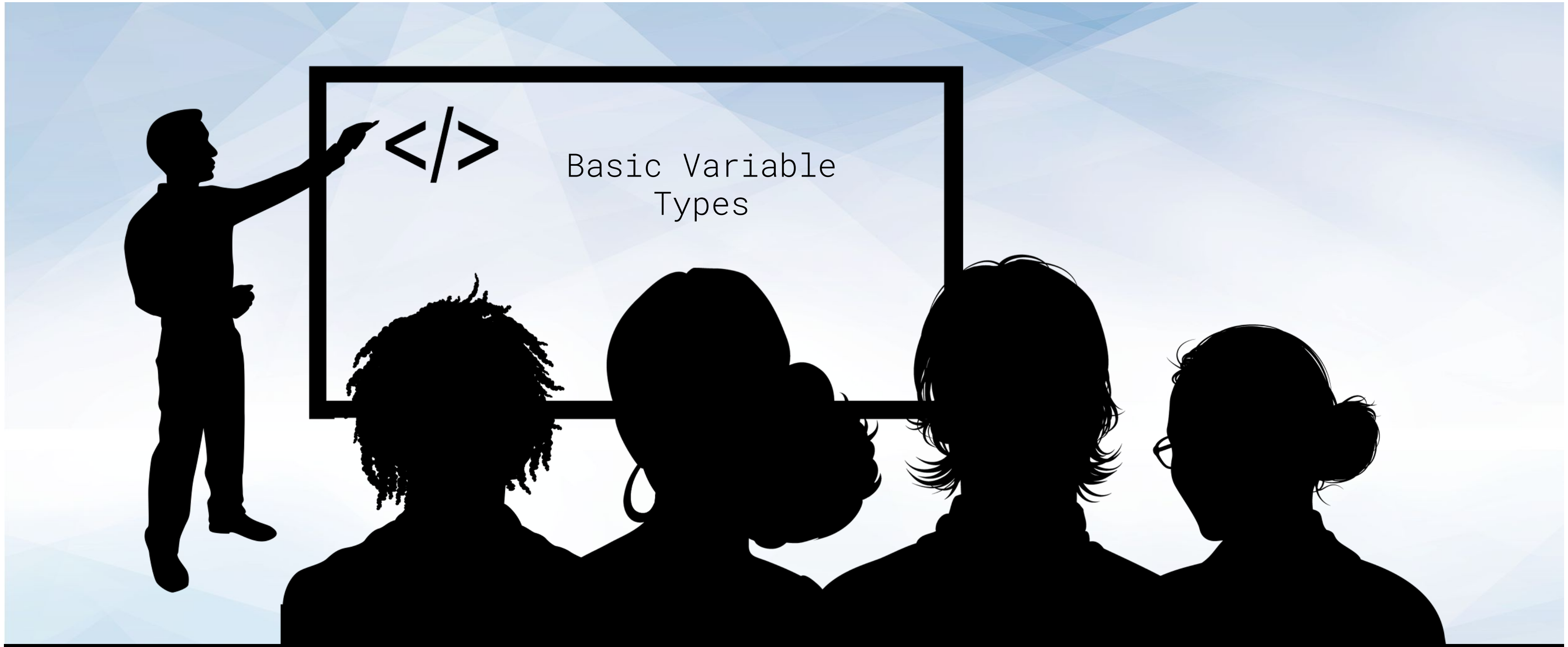
The quotes (`""`),
convey that `"Fa Mulan"` is a **string**.

Variables

Variables that have been assigned a value can then be referenced and reassigned in the code later, as shown in the following image:



```python
# create a variable called 'name' and assign it the value 'Fa Mulan'
name = "Fa Mulan"

# reference the variable - outputs 'Fa Mulan' to the console
print(name)

# reassigns the variable the new value 'Moana'
name = "Moana"

# outputs 'Moana' to the console
print(name)
```



Instructor Demonstration

Basic Variable Types

Variable Types

Integers and Floats

Integers are **whole numbers** that can be either positive or negative.

While there are limits to how large integers can be in other programming languages, *Python allows for integers of any length.*

Floats are numbers specified with a **decimal point**.

Integers and Floats

Integers

```
49
```

Floats

```
6.38
```

Strings

Strings are any collection of characters **bounded by a pair of quotation marks**.

Strings can contain numbers within them but these numbers are seen as characters without any numeric value.

Strings

Strings are any collection of characters.

```
"Hello World!"
```

```
"13 peas, please."
```

```
"#@$^!"
```

Booleans

Booleans are logic data which denote whether something is considered **true** or **false**.

Booleans

Booleans can only have the value `true` or `false`

`true`

`false`

Operators

Variables and data types can be altered using **operations** such as **simple arithmetic** and **concatenation.**

Operators are symbols that alter data.

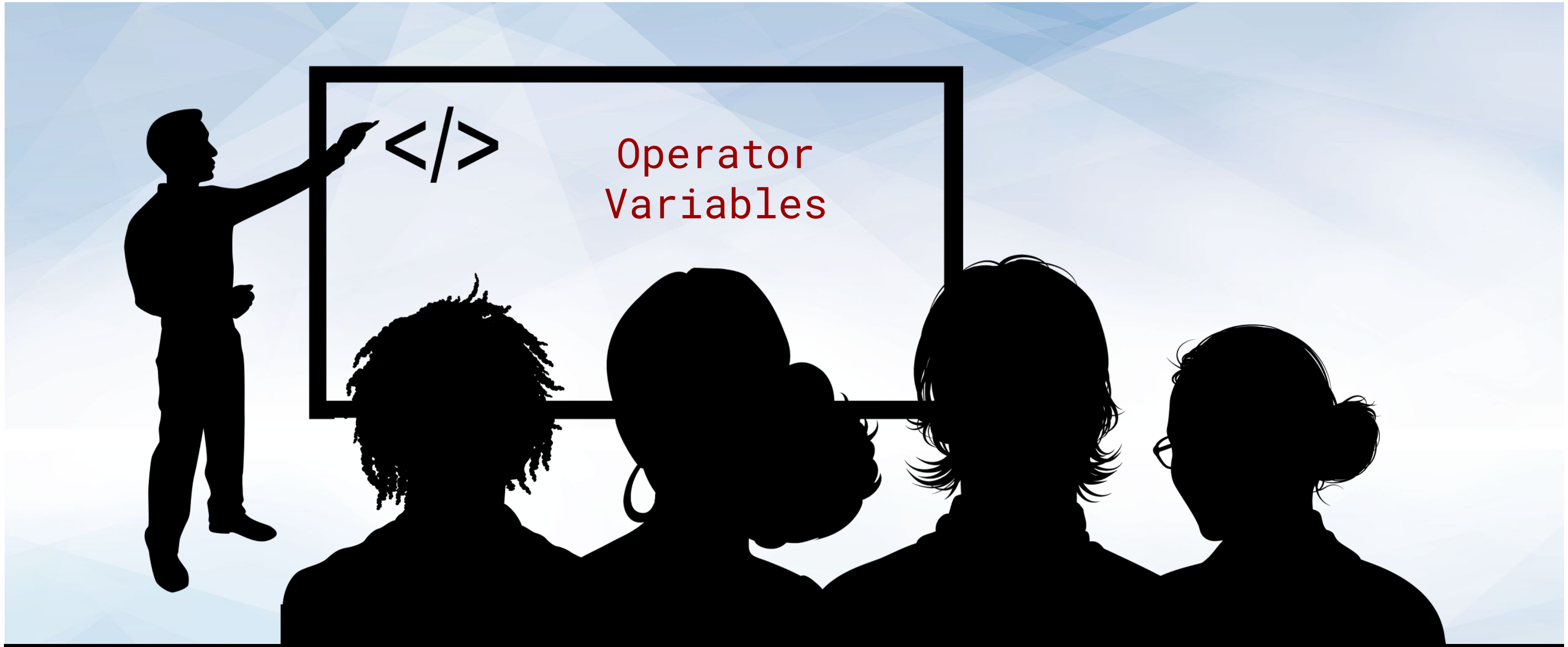
We've already seen an operation when we designated the following value :

The assignment operator = stores a value to a variable

```
name = "Fa Mulan"
```

There are many more! For example:

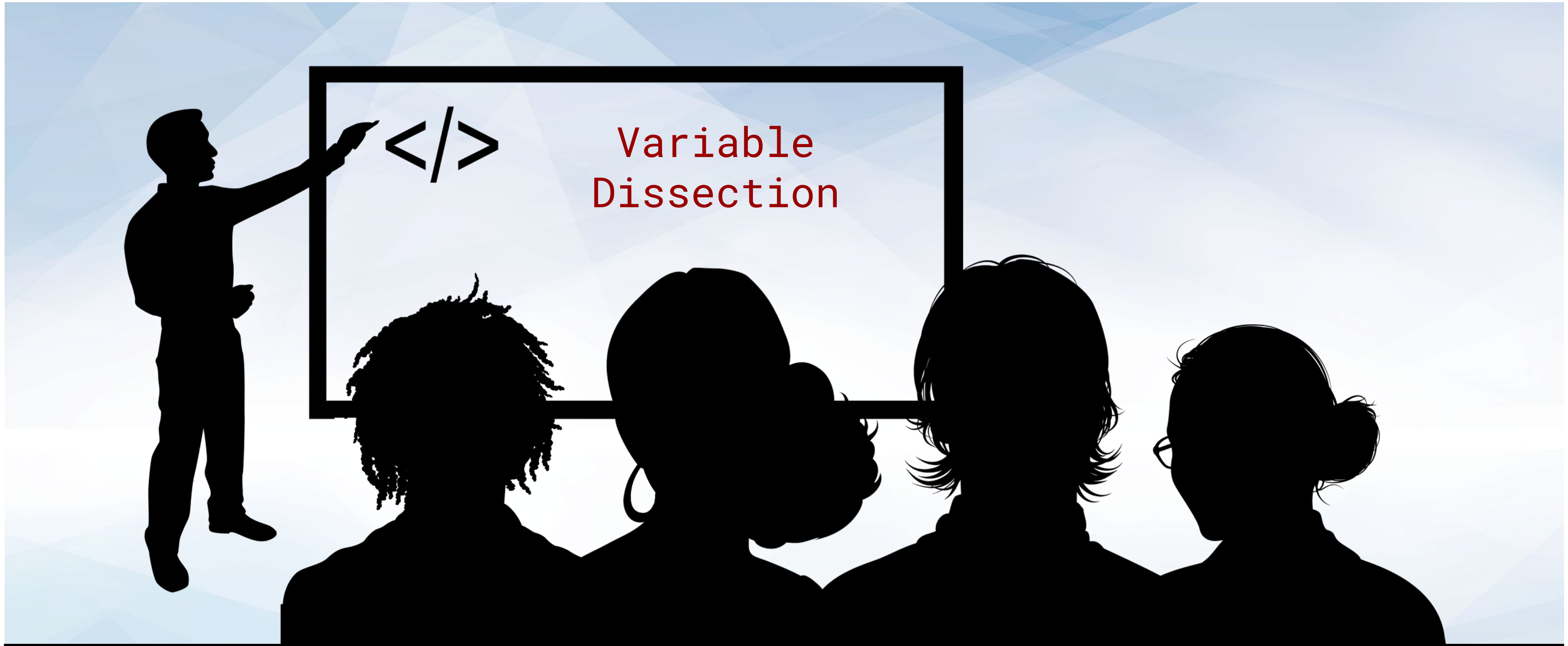
Arithmetic:	`+, -, *, /, etc.`
Comparison:	`<, >, ==, !=, etc.`
Logical:	`and, or, not`



Instructor Demonstration

Operator Variables

Questions?



Instructor Demonstration

Variable Dissection

Activity: Variable Address

Now that we have an understanding of variables and data types we will create some variables for a website, including those that store the daily hits in order to print out a summary of this information.

Instructions sent via Slack

Suggested Time:
12 minutes

Variable Address

Instructions:

1. Create a variable called `URL` which will contain a URL in string form.

2. Create a variable called `IP_address` which will contain an IP address in string form.

3. Create a new variable for each weekday and, using integers, set them equal to how many hits the site got on those days. **Note:** You can choose an arbitrary number for each.

4. Create a variable called `weekly_hits` and set it equal to the sum of the hits on each day of the week.

5. Create a variable called `average_hits` which takes the `weekly_hits` and divides it by the number of weekdays in a week.

6. Print out each variable to the terminal.

Questions?

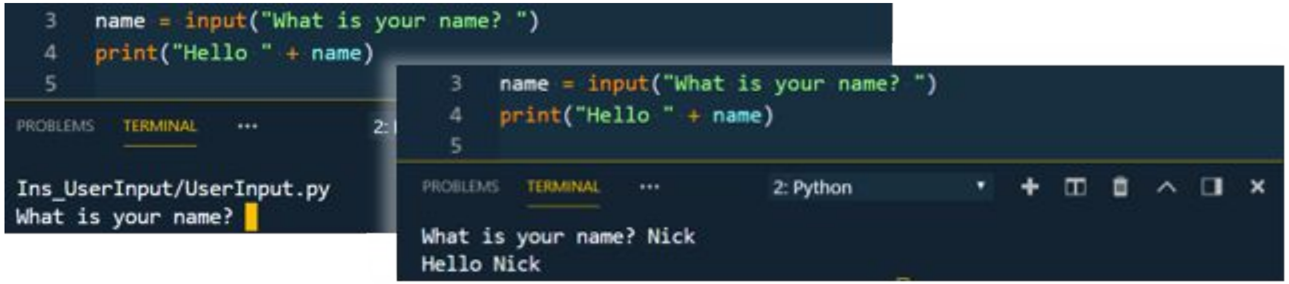
User Input

User Input

To get input from the user in Python, use `input()` and store the result to a variable.

`input` waits to execute any code following it until *after* the user has entered a value.

The value always comes back as a `string` (so you may need to convert it).

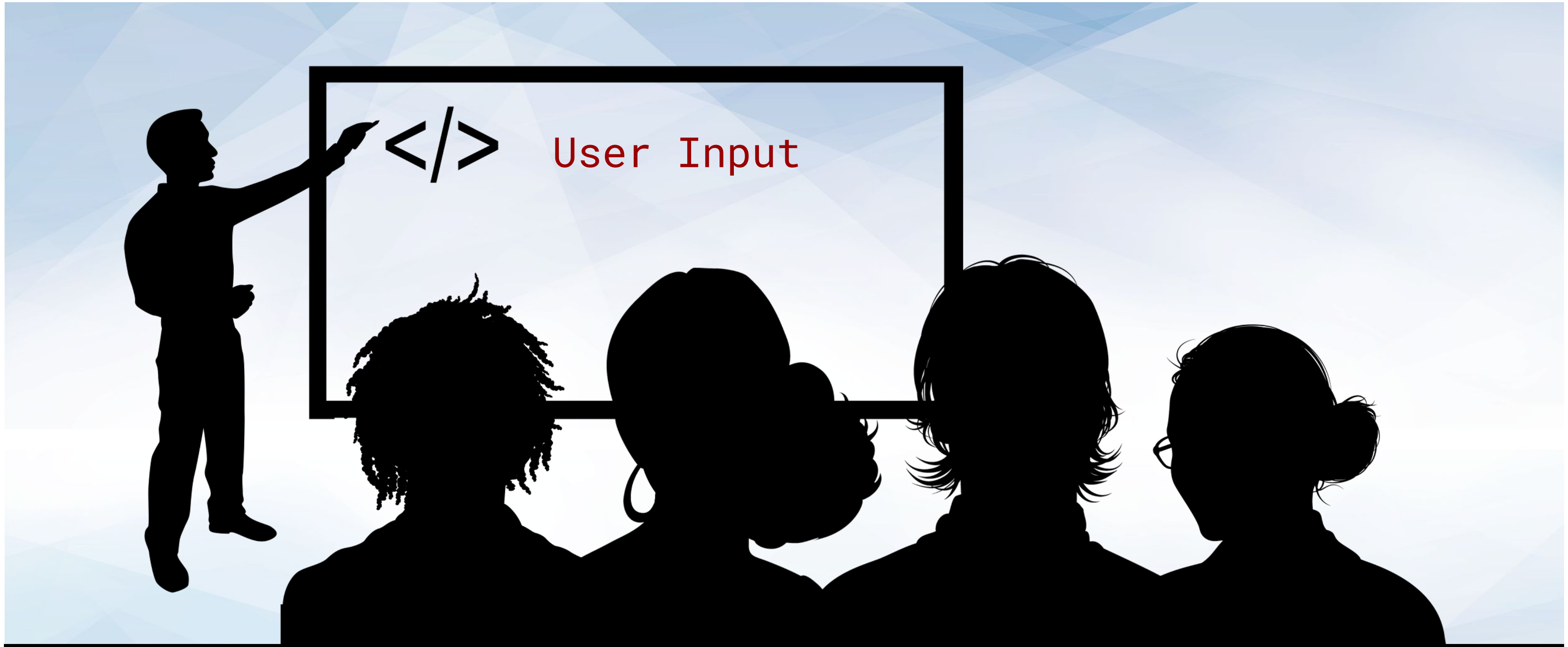


```python
name = input("What is your name? ")
print("Hello " + name)
```

```
Ins_UserInput/UserInput.py
What is your name?
```

```python
name = input("What is your name? ")
print("Hello " + name)
```

```
What is your name? Nick
Hello Nick
```



Instructor Demonstration

User Input

Activity: Down to Input

In this activity, you will create an application by gathering information from your neighbor and then running some code.

Instructions sent via Slack

Suggested Time:
10 minutes

Your Turn: Down to Input

Instructions:

1. Create two different variables, user_name and neighbors_name, that will take the input of your first name and your neighbor's first name.

2. Create two more variables, months_you_coded and months_neighbor_coded, that will take the input of how many months each of you have been coding.

3. Create another variable, months_neighbor_coded that combines the total number of months that each of you have been coding.

4. Print out the following two statements:

The first should say: `I am [user_name] and my neighbor is [neighbor_name].`

The second should say: `Together we have been coded for total_months_coded.`

Example: “`My name is Nick and my neighbor’s name is Jacob. Together we have been coding for 204 months!`”

Questions?

Take a Break!

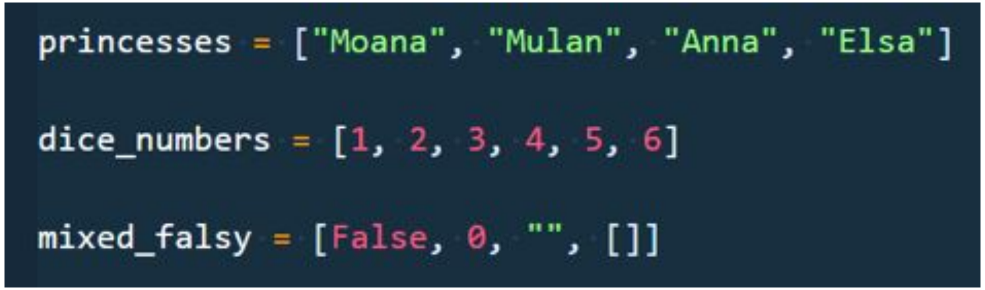
Lists

Lists

Lists are **collections** of data.

These collections can be made up of **strings**, **numbers**, **booleans**, other **arrays**, **dictionaries**, etc.

They typically denote related data, e.g. student names, devices connected to network, etc.

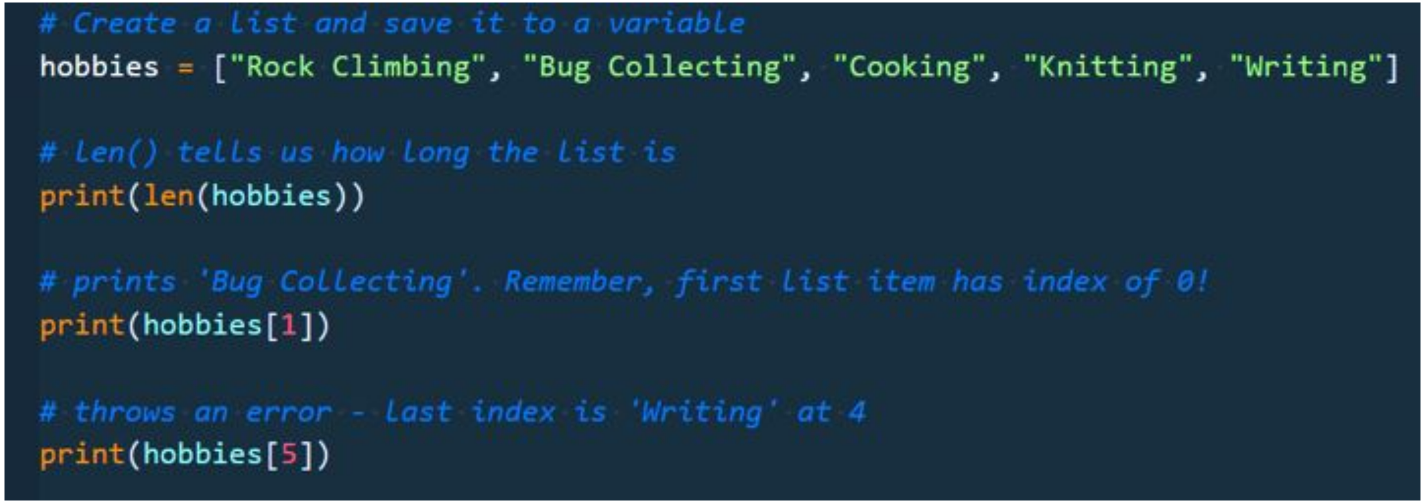


```
princesses = ["Moana", "Mulan", "Anna", "Elsa"]

dice_numbers = [1, 2, 3, 4, 5, 6]

mixed_falsy = [False, 0, "", []]
```

Lists

Each **element** of the array is marked by an **index**. Indexes always start at 0.

To reference the value at a specific index you include the `name of the list` with a square bracket `[ ]`. Inside the bracket you use the `element’s index`.



```
# Create a list and save it to a variable
hobbies = ["Rock Climbing", "Bug Collecting", "Cooking", "Knitting", "Writing"]

# len() tells us how long the list is
print(len(hobbies))

# prints 'Bug Collecting'. Remember, first list item has index of 0!
print(hobbies[1])

# throws an error - last index is 'Writing' at 4
print(hobbies[5])
```

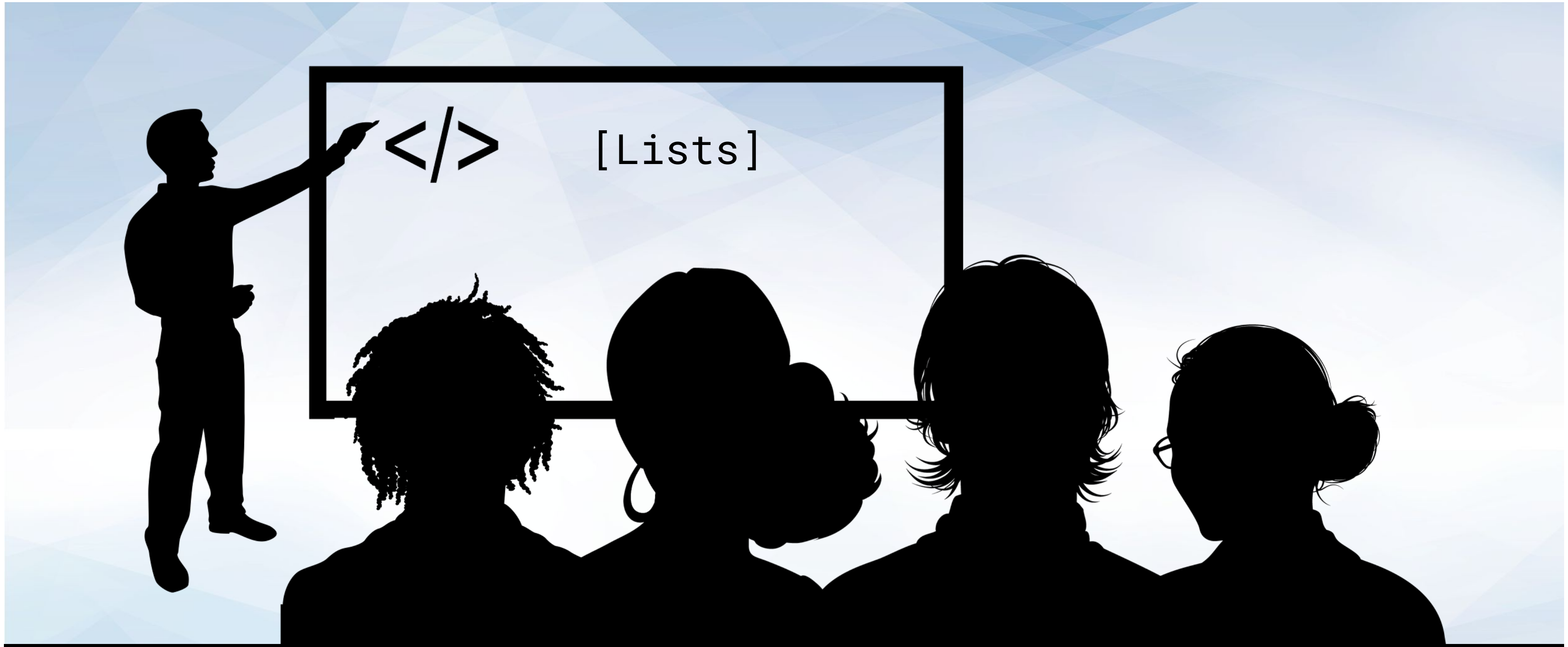
Lists

List Name: zoo_animals

Zebra	Rhino	Giraffe	Owl
Index 0	Index 1	Index 2	Index 3

Coded in Python using a List:

```
zoo_animals = ["Zebra", "Rhino", "Giraffe", "Owl"]
```



Instructor Demonstration
Lists

Activity: Messy Lists

In this activity, you will be given a large list of IP addresses and will answer some basic questions based on its contents. Afterwards, you will add and remove IP addresses to and from the list.

Instructions sent via Slack

Suggested Time:
15 minutes

Your Turn: Messy Lists

Instructions:

1. **Determine** the length of the list and print the length out to the terminal.

2. **Determine** the indexes for the IPs "82.82.0.22" and "207.209.106.220" and then print the indexes out to the terminal.

3. **Add** the following IP addresses to the list:

➢ "220.66.146.40"
➢ "245.201.208.161"
➢ "208.222.148.199"
➢ "104.216.140.187"
➢ "73.57.167.115"

4. **Remove** the following IP addresses from the list:

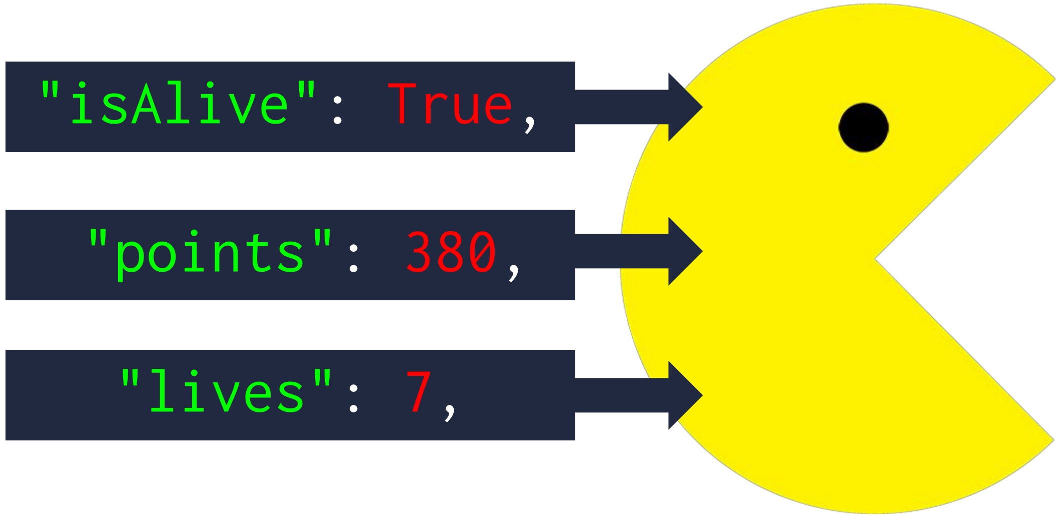
➢ "53.239.114.76"
➢ "65.136.121.223"

Questions?

Dictionaries

Dictionaries

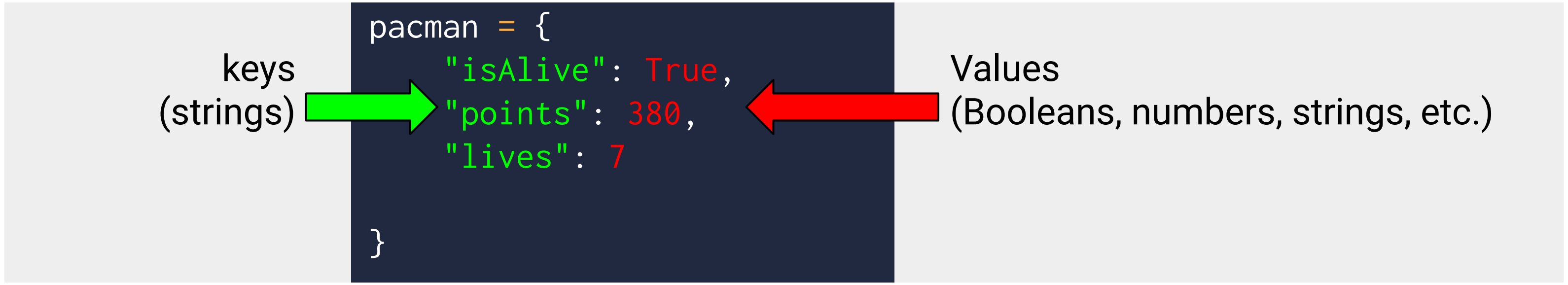
Dictionaries store data in key-value pairings in which the key is a string that can be referenced in order to collect the value that is associated with it.



```
pacman = {
      "isAlive": True,
      "points": 380,
      "lives": 7,

}
```

Dictionaries: Creation

- A dictionary is a mapping of keys to values.
- You use curly braces, { }, to construct the dictionary.
- Values can be any Python value.
- Keys are more limited and are typically strings.

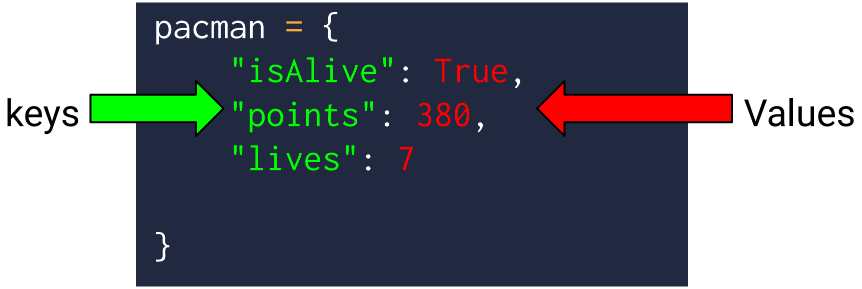


```
pacman = {
    "isAlive": True,
    "points": 380,
    "lives": 7

}
```

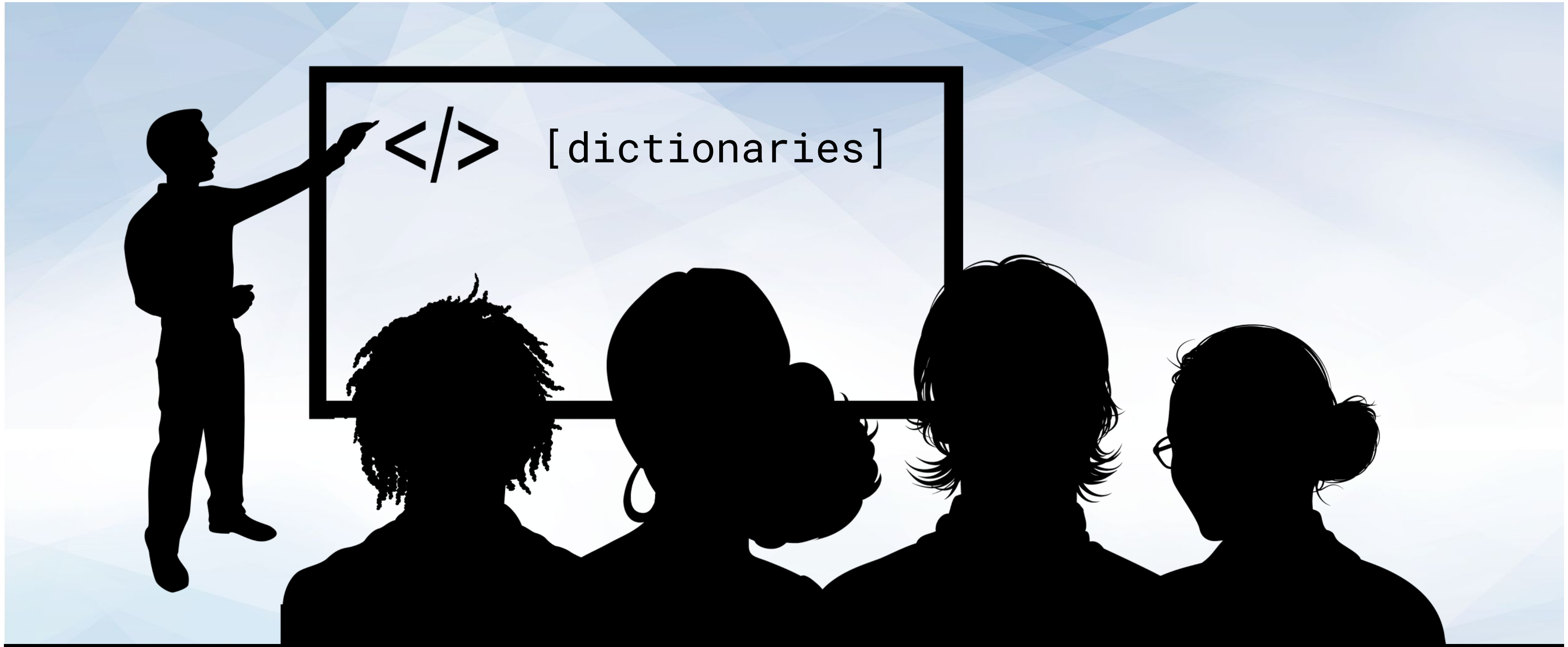
Dictionaries: Referencing

Referencing values in a dictionary is similar to lists. Instead of using a number index, you use the key value. Both use square brackets:

`pacman ["isAlive"]`	evaluates to	`True`
`pacman ["points"]`	evaluates to	`380`



```
pacman = {
    "isAlive": True,
    "points": 380,
    "lives": 7

}
```



Instructor Demonstration

Dictionaries

Activity: Hobby Book

In this activity, you will practice creating and accessing your own dictionaries based on your hobbies.

Instructions sent via Slack

Suggested Time:
15 minutes

Your Turn: Hobby Book

Instructions

Create a dictionary that will store the following:

```
-   Your name
-   Your age
-   A list of a few of your hobbies
-   A dictionary of the times you wake up during the week
```

Print out three statements:

```
-   Hello I am (name) and I am a (occupation)
-   I have (number of) hobbies!
-   On the weekend I get up at (time)
```

Use the file provided to help you get started.

We covered *a lot* today!

Operators
(Arithmetic, Comparison, Logical)

Variable
Assignment
and Reference

Data Types and Conversions
(Numbers, String, Boolean)

Lists

User Input

Dictionaries

Python Learning Tips

01 **Review Immediately.** We'll be building on these concepts quickly. The firmer your grasp now, the better off you'll be.

02 **Redo the exercises from class.** Don't just reread! Spend some time redoing the activities from scratch on your own.

03 **Come to office hours.** Ask conceptual questions. Ask specific questions. Just keep asking questions!

04 **Don't Be Afraid.** You will get this. It will take time, but you *will* get this. Keep at it. Patience will pay off.

05 **Practice Practice Practice.** Only by doing will you learn how to code—reading code and reviewing helps, but writing code is the best way to succeed.

Programming Fundamentals We Covered Today:

Data	Logic
1. Numbers	1. Operators
2. Strings	2. Conditionals
3. Booleans	3. Loops
4. Lists	4. Functions
5. Dictionaries	5. Modules

Class Objectives

By the end of class today, you will be able to:

- ✅ Explain how and why Python is used in cybersecurity.
- ✅ Create and run Python files via the terminal using VS Code.
- ✅ Use the `print()` function to print lines to the console.
- ✅ Use basic Python elements like variables and operators.
- ✅ Employ the Python `input()` function to retrieve, store, and utilize user inputs.
- ✅ Reference and store collections of data using lists.
- ✅ Create and reference data in dictionaries.
- ✅ Use `listVariable.append(Value)`, `listVariable.index(Value)`, and `listVariable.remove(Value)` to add, return, and delete values from a list.
- ✅ Use `len(listVariable)` function to return the length of a list.

Any Questions?

See you next class!