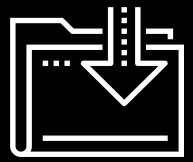
Introduction to Python

Data Boot Camp

Lesson 3.1



Class Objectives

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

Perform Python 3 installation.

Navigate through folders and files via the terminal.

Create Python scripts and run them in the terminal.

Understand basic programming concepts in Python.

Let’s explore Python!

Python

A few things to note before we start:

Python is a traditional programming language.

The fundamental difference between VBA and Python is the syntax.

Instructor Demonstration

Terminal

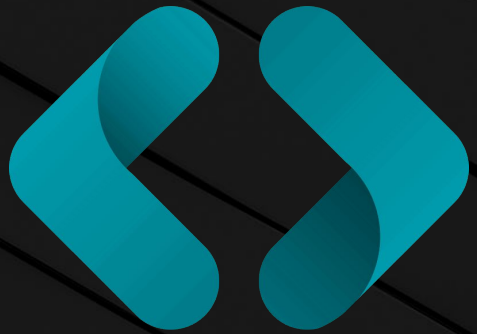
Some basic commands

`cd`	changes the directory.
`cd ~`	changes to the home directory.
`cd ..`	moves up one directory.
`ls`	lists files in the folder.
`pwd`	shows the current directory.
`Mkdir <FOLDERNAME>`	creates a new directory with the FOLDERNAME
`touch <FILENAME>`	creates a new file with the FILENAME.
`rm <FILENAME>`	deletes a file.
`rm -r <FOLDERNAME>`	deletes a folder; note the -r.
`open .`	opens the current folder on Macs.
`explorer .`	opens the current folder on Windows.
`open <FILENAME>`	opens a specific file on Macs.
`explorer <FILENAME>`	opens a specific file on Windows.

Common Commands

```
bash-3.2$ mkdir PythonStuff
bash-3.2$ cd PythonStuff
bash-3.2$ touch first_file.py
bash-3.2$ open first_file.py
```

```
bash-3.2$ python first_file.py
bash-3.2$ This is my first_file.py
```



Time to <code>

Activity: Terminal

In this activity, you will create three folders and a pair of Python files to print strings of your own creation to the console.

Suggested Time:

10 minutes

Activity: Terminal

Write and execute the following commands:

- Create a folder called `LearnPython`.
- Navigate into the folder.
- Inside `LearnPython`, create another folder called `Assignment1`.
- Inside `Assignment1`, create a file called `quick_python.py`.
- Add a print statement to `quick_python.py`.
- Run `quick_python.py`.
- Return to the `LearnPython` folder.
- Inside `LearnPython`, create another folder called `Assignment2`.
- Inside `Assignment2`, create a file called `quick_python2.py`.
- Add a different print statement to `quick_python2.py`.
- Run `quick_python2.py`.

Time’s Up! Let’s Review.

Time to Code

Check Your Anaconda Installation

Suggested Time:

5 minutes

Check Your Anaconda Installation

Check if Anaconda is properly installed by doing the following:

01 Open the terminal.

02 Run `conda --version` and press Enter.

03 The terminal output should return `conda 4.x.x`

Time to Code

Create a Virtual Environment

Suggested Time:

10 minutes

Create a Virtual Environment

What is a virtual environment?

Virtual environments create an isolated environment for Python projects.

You may work on different projects that have different dependencies.

Different projects might also use different types and versions of libraries.

This virtual environment ensures you have the required dependencies for future class activities.

Instructor Demonstration

Variables

Variables

- Variables are similar to values stored in VBA cells.
- In Python, a value is being stored and named.
- Variables can store different data types, like strings, integers, and an entirely new data type called Booleans, which hold true or false values.

```
# Creates a variable with a string "Frankfurter"
title = "Frankfurter"

# Creates a variable with an integer 80
years = 80

# Creates a variable with the boolean value of True
years = True
```

Print Statements

We can print statements that include variables, but traditional Python formatting won't concatenate strings with other data types. This means integers and Booleans must be cast as strings by using the `str()` function.

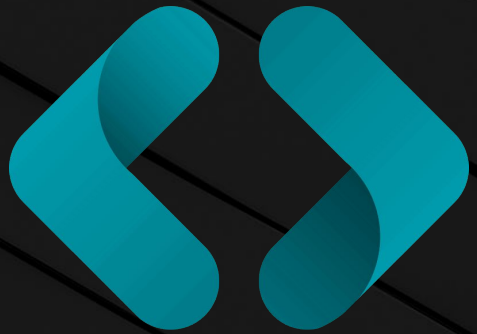
```
# Prints a statement adding the variable
print("Nick is a professional" + title)

# Convert the integer years into a string and prints
print("He has been coding for " + str(years) + " years")

# Convert a boolean into a string and prints
print("Expert status:" + str(expert_status))
```

Alternatively, the f-string method of string interpolation allows strings to be formatted with different data types.

```
# An f-string accepts all data types without conversion
print(f"Expert status: {expert_status}")
```



Time to <code>

Activity: Hello, Variable World!

In this activity, you will create a simple Python application that uses variables to run calculations on integers and print strings out to the console.

Suggested Time:

10 minutes

Activity: Hello, Variable World!

Instructions:

- Create two variables, called `name` and `country`, that will hold strings.
- Create two variables, called `age` and `hourly_wage`, that will hold integers.
- Create a variable called `satisfied`, which will hold a Boolean.
- Create a variable called `daily_wage`, which will hold the value of `hourly_wage` multiplied by 8.
- With an f-string, print the `daily_wage` and `satisfied` variables.

```
HelloVariableWorld.py
You live in the United States
You are 25 years old
You make 120 per day
Are you satisfied with your current wage? True
```

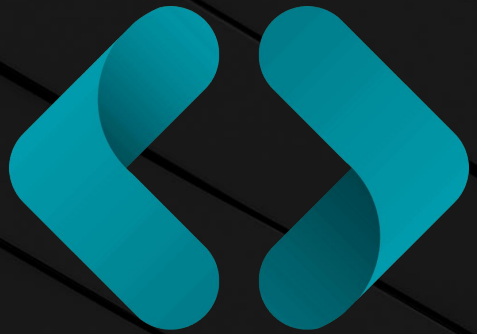
Time’s Up! Let’s Review.

Instructor Demonstration

Inputs and Prompts

Inputs and Prompts

```
(PytonData) $ python inputs.py
What is your name? Gary
How old are you? 33
Is this statement true? yes
My name is Gary
I am 33 years old.
The statement was True
```



Time to <code>

Activity: Down to Input

In this activity, you will store inputs from the command line and run code based on the values entered.

Suggested Time:

10 minutes

Activity: Down to Input

Instructions

Create two different variables, one to take the input of your first name and one for your neighbor's first name.

Create two more inputs to ask how many months you and your neighbor have been coding.

Finally, display a result with both your names and the total amount of months you've been coding.

Time’s Up! Let’s Review.

Instructor Demonstration

Conditionals

Conditionals

A few things to keep in mind!

- Conditionals in Python feature nearly the same logic as VBA. The primary differences are the syntax and indentation.
- Python uses `if`, `elif`, and `else` to create contionals.
- Conditional statements are concluded with a colon. Because Python reads blocks of code based on indentation, all lines after the colon **must** be indented to be considered a part of that code block.
- All sorts of operators, like greater than, less than, and equal to, can be used to create logic tests for conditionals.
- The condition `is equal to` uses `==`, while variable assignment uses one equal sign.
- Multiple logic tests can be checked within a single conditional statement. If we use the term `and`, both tests must return `True`, while `or` requires that only one test return as `True`.
- Conditionals can even be nested, allowing programmers to run logic tests based on whether or not the original logic test returned as `True`.

Conditionals

Indentation matters in Python!

```
>>> x = 1
>>> y = 10
>>>
>>> # Look what happens w/o indentation
... if x == 1:
... print('x is equal to 1')
 File "<stdin>", line 3
    print('x is equal to 1')
        ^
IndentationError: expected an indented block
```

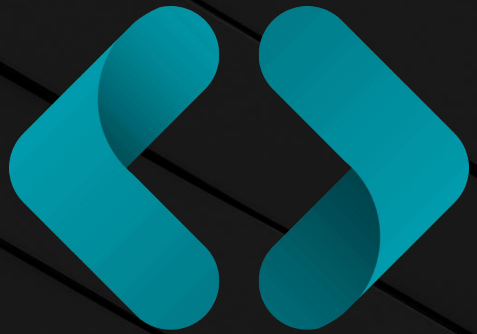
Hint: Count four space strokes on your keyboard, or press the Tab key once.

```
>>> if x == 1:
```

Conditionals

```
>>> # Checks if one value is equal to another
...  if x == 1:
...    print(“x is equal to 1”)
x is equal to 1
>>> # Checks if one value is NOT equal to another
...  if y != 1:
...    print(“y is not equal to 1”)
y is not equal to 1
>>> # Checks if one value is less than another
...  if x < y:
...    print(“x is less than y”)
x is less than y
>>> # Checks if one value is greater than another
...  if y > x:
...    print(“y is greater than x”)
y is greater than x
```

```
>>> # Checks if one value is greater than or equal to another
... if x >= 1:
...      print(“x is greater than or equal to 1”)
x is greater than or equal to 1
>>> # Checks for two conditions to be met using “and”
... if x == 1 and y == 10:
...      print(“Both values returned true”)
Both values returned true
>>> # Checks if either of two conditions is met
... if x < 45 or y <5:
...      print(“One or more of the statements were true”)
One or more of the statements were true
>>> # Nested if statements
... if x < 10:
...      if y < 5:
...           print(“x is less than 10 and y is less than 5”)
...      elif y == 5:
...           print(“x is less than 10 and y is equal to 5”)
...      else:
...           print(“x is less than 10 and y is greater than 5”)
```



Time to <code>

Activity: Conditional Conundrum

In this activity, you will review some prewritten conditionals and predict which lines will be printed to the console.

Suggested Time:

10 minutes

Activity: Conditional Conundrum

Instructions	Go through the conditionals in the provided code, and predict which lines will be printed to the console.
	Do not run the application at first. Try to follow the thought process for each code chunk and then make a guess. You should only run the application after coming up with a guess for each section.
Bonus	After figuring out the output for all of the code chunks, create your own series of conditionals to test your fellow classmates. Once you have completed your puzzle, slack it out to everyone so they can test it.

Time’s Up! Let’s Review.

Countdown timer

15:00

(with alarm)

Instructor Demonstration

Lists

Lists

A few points to keep in mind:

Lists are the Python equivalent of arrays in VBA. Like arrays, lists hold multiple pieces of data within one variable.

Lists can hold multiple types of data, such as strings, integers, and Boolean values, all within a single list!

List Methods in Python

Python has a set of built-in methods that you can use on lists:

`append`	method adds elements to the end of a list.
`index`	method returns the numeric location of a given value within a list.
`len`	function returns the length of a list.
`remove`	method deletes a given value from a list.
`pop`	method can remove a value by index.

```
# Create a variable and set it as an List
myList = ["Jacob", 25, "Ahmed", 80]
print(myList)

# Adds an element onto the end of a List
myList.append("Matt")
print(myList)

# Changes a specified element within an List at the given index
myList[3] = 85
print(myList)

# Returns the index of the first object with a matching value
print(myList.index("Matt"))

# Returns the length of the List
print(len(myList))

# Removes a specified object from an List
myList.remove("Matt")
print(myList)

# Removes the object at the index specified
myList.pop(0)
myList.pop(0)
print(myList)
```

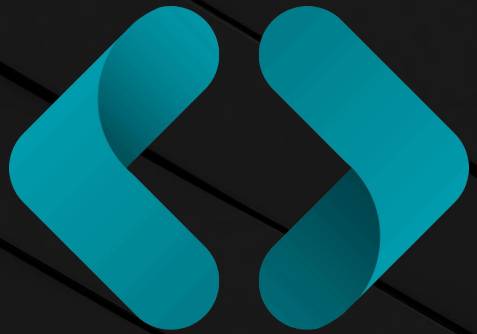
Tuples

Tuples are functionally similar to lists in what they can store, but they are immutable.

While lists in Python can be modified after their creation, tuples can never be modified after their declaration.

Tuples tend to be more efficient to navigate through than lists and also protect the data stored within from being changed.

```
# Creates a tuple, a sequence of immutable Python objects that cannot be changed
myTuple = ('Python', 100, 'VBA', False)
print(myTuple)
```



Time to <code>

Activity: Rock, Paper, Scissors

In this activity, you will create a simple game of Rock, Paper, Scissors that will run within the console.

Suggested Time:

15 minutes

Activity: Rock, Paper, Scissors

Instructions

- Using the terminal, take an input of `r`, `p`, or `s` for rock, paper, or scissors.
- Have the computer randomly pick one of these three choices.
- Compare the user's input to the computer's choice to determine if the user won, lost, or tied.

```
(PythonData) $ python RPS_Solved.py
Let’s Play Rock Paper Scissors!
Make your Choice: (r)ock, (p)aper, (s)cissors? P
You choose paper. The computer choose rock.
Yay! You won.
```

Time’s Up! Let’s Review.

Instructor Demonstration

Loops

Loops

Loops are another concept that we learned with VBA!

- The variable `x` is created within the loop statement and could theoretically take on any name as long as it is unique.
- When looping through a range of numbers, Python will halt the loop one number before the final number. For example, when looping from 0 to 5, the code will run 5 times, but `x` will only ever be printed as 0 through 4.
- When provided with a single number, `range()` will always start the loop at 0. However, when provided with two numbers, the code will loop from the first number until it reaches one fewer than the second number.

```
# Loop through a range of numbers (0 through 4)
for x in range(5):
    print(x)

print("----------------------------------------")

# # Loop through a range of numbers (2 through 6)
for x in range(2, 7):
    print(x)

print("----------------------------------------")
```

Looping through strings

Python can also loop through all the letters within a string.

The syntax is for `<variable> in <string>`:

```
# Iterate through letters in a string
word = "Peace"
for letters in word:
    print(letters)

print("_____________________________________________")
```

Looping through lists

Python can also loop through all the values within a list.

The syntax is for `<variable> in <list>:`

```
# Iterate through a list
zoo = ['cow', 'dog', 'bee', 'zebra']
for animal in zoo:
    print(animal)

print("_____________________________________________")
```

`while` Loops

These are just like `for` loop but will continue looping for as long as a condition is met.

```
# Loop while a condition is being met
run = 'y'
white run == 'y':
    print('Hi!')
    run = input("To run again. Enter 'y'")
```

Activity: Number Chain

In this activity, you will take user input and print out a string of numbers.

Suggested Time:

15 minutes

Activity: Number Chain

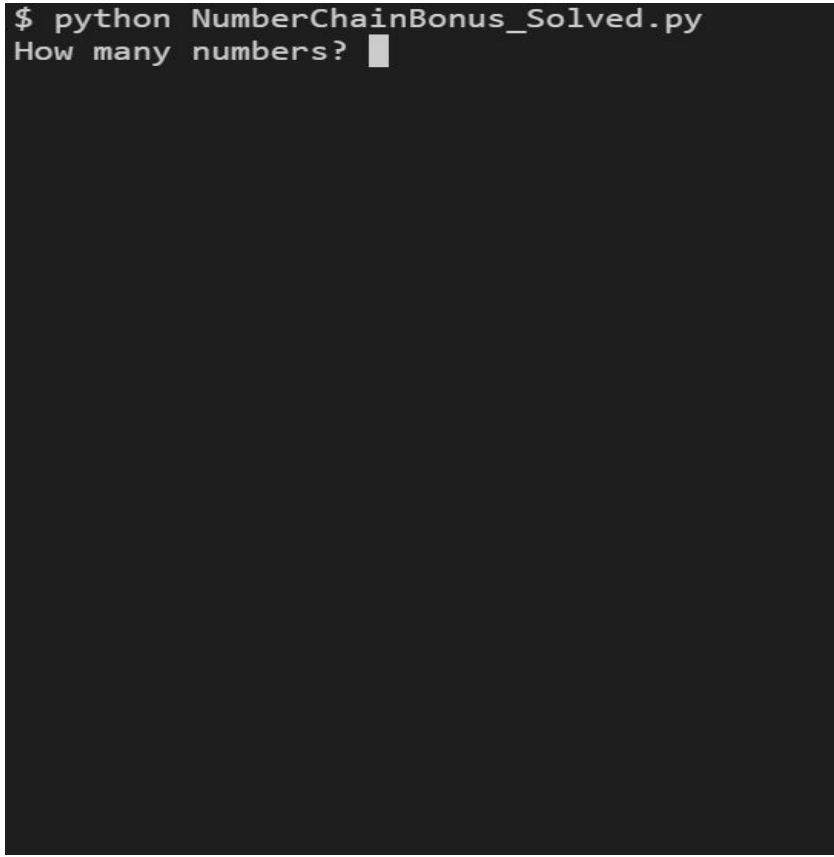
Instructions

Using a `while` loop, ask the user "How many numbers?", and then print out a chain of numbers in increasing order from 0 to the user-input number.

After the results have printed, ask the user if they would like to continue. If "y" is entered, keep the chain running by inputting a new number and starting a new count from 0 to the new user-input number. If "n" is entered, exit the application.

Bonus

Rather than just displaying numbers starting from 0, have the numbers begin at the end of the previous chain.



Time's Up! Let's Review.

Questions?