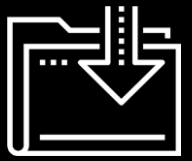
Python and Functions

"It's hard to beat a person who never gives up."

-Babe Ruth

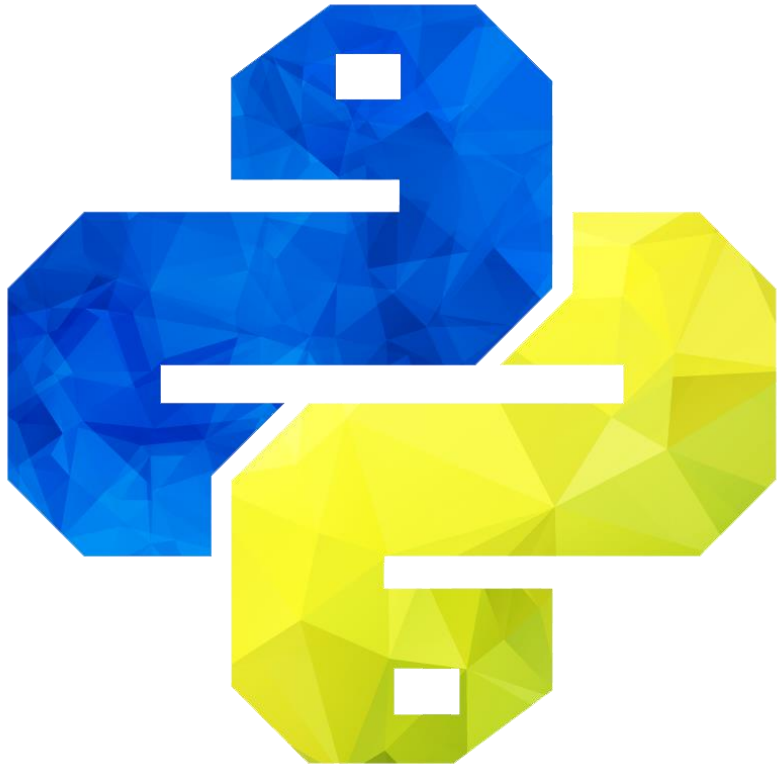
Cybersecurity Boot Camp

Python 1: Day 3



Python: Python and Functions

Today's class, like the previous one, will introduce new programming building blocks with Python. The focus today will be functions.



Class Objectives

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

- ❏ Define and call functions
- ❏ Create functions to print data from a dictionary
- ❏ Create functions with arguments to make them more modular
- ❏ Create functions with return values
- ❏ Build command-line applications that both return and print out values to the user

Last Class we Covered:

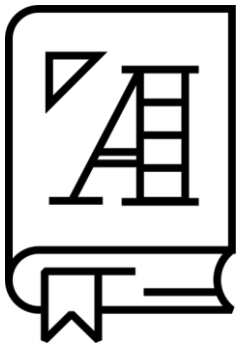
Data	Logic
1. Numbers	1. Operators
2. Strings	2. Conditionals
3. Booleans	3. Loops
4. Lists	4. Functions
5. Dictionaries	5. Modules

Today's Class:

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

Functions

Functions



Functions are block of organized, reusable code that perform a single action multiple times

Functions

We've already used some "built-in" functions like:

`print()`, `input()`, and `range()`

But programmers will also want to create their own "**`user-defined`**" functions for more specific use cases.

Functions

The first step is defining our function.

```
# Creating a new user-defined function called printHello()
def printHello():
    # This indented code will be run anytime printHello() is called
later in the application
    print("Hello User!")
    print("You have just defined and called your first function!")
    print("Great job!")



# A function must be executed in order for the code it contains to
be run
printHello()
printHello()
printHello()
```

To define a function, we use the **`def`** keyword followed by the function's name and a pair of parentheses.

The block of code contained within the function begins with a colon and must be indented in order to work properly.

Global Variables

Not all variables are accessible or available to all parts of our program.

```
# This variable is defined at the top level of the application
# This means that its scope is "global" and it can be referenced
anywhere
global_variable = "I CAN GO ANYWHERE!"

# Creating a function
def printPhrases():

 # This variable is defined within a function
 # This means that it can only be referenced within this function
 local_variable = "I ONLY EXIST IN THIS FUNCTION"
```

If you define a variable at the top of your program or in the main body of your program, it is considered a *global variable*.

Any part of your program, for example any functions, can access that variable and use it for example as a parameter in the function.

Local Variables

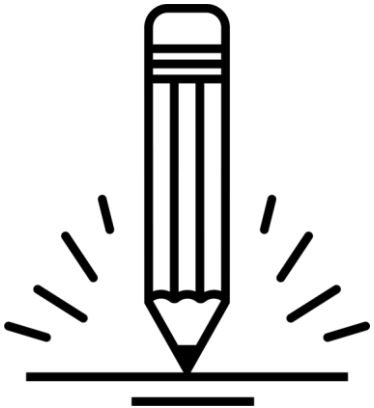
Not all variables are accessible or available to all parts of our program.

```
# This variable is defined at the top level of the application
# This means that its scope is "global" and it can be referenced
anywhere
global_variable = "I CAN GO ANYWHERE!"

# Creating a function
def printPhrases():

 # This variable is defined within a function
 # This means that it can only be referenced within this function
 local_variable = "I ONLY EXIST IN THIS FUNCTION"
```

If you define a function within a specific function, then that that variable is *local* and accessible only to that function.

Any other function outside of that one in your program cannot access it.

The picture can't be displayed.



Activity: My Very First Functions

In this activity, you will be given a dictionary and asked to create three functions to print valuable information to the terminal.

Activities/04-Stu_FirstFunction/Unsolved/Readme

Suggested Time:
7 minutes

Your Turn: Password Check

Instructions:

Using the provided dictionary of names and social security numbers (SSNs), complete the following tasks:

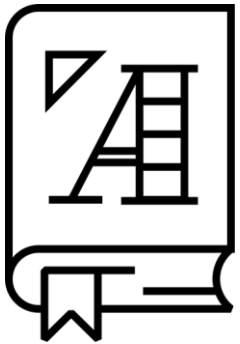
- Create a function that will loop through all of the keys in the dictionary and print them out one at a time.
- Create a function that will loop through all of the values in the dictionary and print them out one at a time.
- Create a function that will loop through all of the keys AND values in the dictionary and print them out one at a time.

Hints:

- You will need to use loops within your functions.
- The file we provided to you guides you step by step through the code you will have to create.
- Remember that you will need to call your functions to execute them.

Parameters

Parameters



Parameters are values used by functions to complete actions. A *parameter* is the variable. The *argument* is the specific value that is contained in that variable.

Parameter

Defining a function

```
# Functions can be given multiple parameters
# Parameters can also be provided with default values
def recordScore(name, score=0):
 # The score that is printed out will default to 0 if none is
provided
 # First value
 print(name + "'s score is " + str(score))


recordScore("Jacob")
recordScore("Ahmed", 20)
recordScore("Steven", 15)
```

The parameter is defined in the def statement of the function and is a variable you pass ito the function.

When a function is called, it passes a specific value or argument into the function.

Parameters

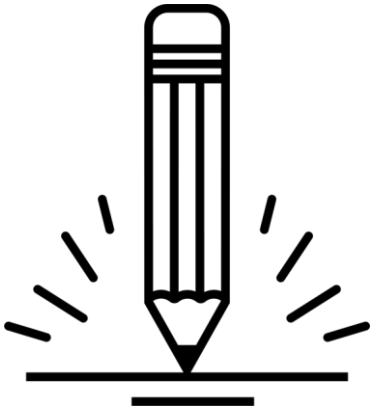
Multiple parameters and default values

```
# Functions can be given multiple parameters
# Parameters can also be provided with default values
def recordScore(name, score=0):
 # The score that is printed out will default to 0 if none is
provided
 # First value
 print(name + "'s score is " + str(score))


recordScore("Jacob")
recordScore("Ahmed", 20)
recordScore("Steven", 15)
```

If you call a function but do not include a specific argument, then the default value will be assigned.

All parameters with default values must come after those parameters without.

The picture can't be displayed.



Activity: Calculator Activity

In this activity, you will make a command-line calculator application. The application will ask the user what kind of arithmetic operation they would like to perform.

Activities/06-Stu_FunctionalCalculator/Unsolved/Readme

Suggested Time:
20 minutes

Your Turn: Calculator Activity (Activities/06-Stu_FunctionalCalculator/Unsolved/Readme)

Instructions:

In this activity you are creating a command-line calculator application. When completed, the app should perform the following:

- Ask the user what kind of arithmetic operation they would like to use.
- Add two integers together if the user selects addition.
- Subtract two integers from one another if the user selects subtraction.
- Multiply two integers together if the user selects multiplication.
- Divide two integers by one another if the user select division.

Each of the arithmetic operations should be contained within functions that take in two integers as parameters.

Hint: The file we provided to you guides you through the code you will use to create your calculator. We also added some of the code for you to get you started.

Take a Break!

Returning Values

Returning variables

Functions can return values using the “return” statement.

```
# use the 'return' keyword in a function
def sum(a, b):
    return a + b

# assign the function call to a variable to use the value later
result = sum(2, 3)
print("The first sum is " + str(result))

result2 = sum(4, 6)
print("The second sum is " + str(result2))

sumOfSums = sum(result, result2)
print("The sum of sums is " + str(sumOfSums))
```

Whatever value is placed after `return` will be passed back to the main script.

The return statement concludes whatever function it is placed inside of. Therefore, only one return can be reached within a function.

Returning variables

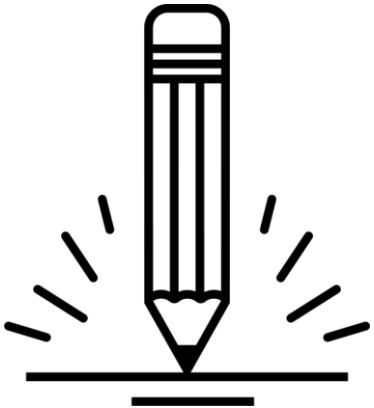
Storing values within variables:

```
# use the 'return' keyword in a function
def sum(a, b):
    return a + b
```

```
# assign the function call to a variable to use the value later
result = sum(2, 3)
print("The first sum is " + str(result))

result2 = sum(4, 6)
print("The second sum is " + str(result2))

sumOfSums = sum(result, result2)
print("The sum of sums is " + str(sumOfSums))
```

When a function containing a `return` statement is called, it is commonplace to store the results within a variable so that the value could be referred to multiple times throughout the remainder of the application.

The picture can't be displayed.



Activity: Validate Password

In this activity, you will create an application that checks whether or not a password entered by the user is of valid length.

Activities/08-Stu_ValidatePassword/Unsolved/Readme

Suggested Time:
15 minutes

Your Turn: Validate Password

Instructions:

Use the script file provided. We've added a lot of the code for you to get started.

Define a function called `validate_password` which accepts a `password` parameter that will be a string.

Inside the function, check if the password is longer than six characters long. Return `True` if it is and return `False` if it is not.

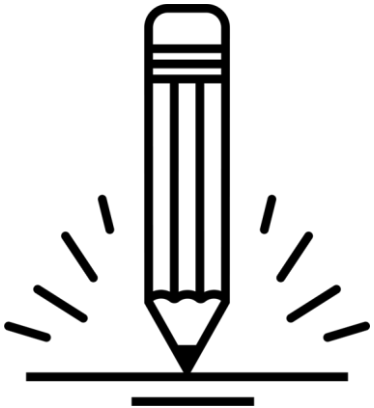
Prompt the user to enter a password and then send this password into the `validate_password` function. Save the result of the function to a variable called `result`.

Print out the value of `result` to the terminal.

Hints:

- Strings can be looped through just like lists can except, in this case, the application is looping through each character in the string instead.
- Since strings can be looped through like lists, the length of a string can also be collected using the `len()` function.

The picture can't be displayed.



Activity: User Creation

In this activity, you will create a command-line application that allows users to create new usernames, passwords, and email addresses for a fake account.

Activities/09-Stu_UserCreation/Unsolved/Readme

Suggested Time:
15 minutes

Your Turn: Validate Password

Instructions:

Use the script file provided. We've added a lot of the code to get you started.

Your application should have the following features:

- A function called `collect_user_information` that will prompt the user for their username, password, and email address. It should return this information in a list that contains those three values.
- The above returned list should be passed into a function called `create_user` that checks if the password entered is valid.
- If the password is valid, it will create a new dictionary for the user with their information. The dictionary should have keys for username, password, and email with the associated values that the user entered. It should then print a message to the screen with this information.
- If the password is not valid, it will print a message to the screen letting the user know that their password isn’t valid.

Python topics we've covered so far:

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:

- ☑ Define and call functions
- ☑ Create functions to print data from a dictionary
- ☑ Create functions with arguments to make them more modular
- ☑ Create functions with return values
- ☑ Build command-line applications that both return and print out values to the user

Questions?
Let’s Review