SeaGlide_Team	Functions And Methods	Lesson # 3 (60 minutes)

Description:
Your role as an assistant computer programmer means that you must learn to be proficient in many types of programming and coding. For this assignment, you will have to learn how to use a high-level programming tool known as Python. Because you are in training, you will be working on several modules and continue to build upon your skills. You will work on various skills in Python and submit them to the head computer programmer to make sure your codes are both sufficient and efficient. For this task, you will learn and utilize functions and modules in Python by designing your own program.

Students will be able to:

- Understand the difference between built-in and user defined functions.
- Utilize Python modules.
- Debug a provided Python code.
- Understand order of execution.
- Apply logic and problem solving skills to design a program.

Students will understand:
Python is a widely known, high-level coding language. Prior to this lesson, students were exposed to understanding how computer programming languages are logical, methodical and meticulous. They were also exposed to how to define and utilize variables by understanding the concepts of data types and expressions. In this lesson, students will learn the usefulness of defining and utilizing functions. Students will begin understanding how to create a Python program from start to finish.

Key Definitions & Concepts: [1]

- **Function**: a named sequence of statements that performs some useful operation. They may or may not produce a result
- **Function Definition**: a statement that creates a new function, specifying its name, parameters and the statements it executes
- **Header**: the first line of a function definition
- **Body**: the sequence of statements inside a function definition
- **Parameter**: a name used inside a function to refer to the value passed as an argument
- **Function Call**: a statement that executes a function; consists of the function name followed by an argument list
- **Argument**: a value provided to a function when the function is called; the value is assigned to the corresponding parameter in the function
- **Return Value**: the result of a function and/or the value of an expression if the function call is used as an expression
- **Module**: a file that contains a collection of related functions and other definitions
- **Import Statement**: a statement that reads a module file and creates a module object
- **Dot Notation**: the syntax for calling a function in another module by specifying the module name followed by a period and the function name
- **Flow of Execution**: the order in which statements are executed during a program run

Standards: [Copied from: 2]
2.AP.14: Create procedures with parameters to organize code and make it easier to use.
1B.AP.11: Decompose problems into smaller, manageable subproblems to facilitate the program development process.
1B.AP.15: Test and debug a program or algorithm to ensure it runs as intended.

Background Information

Prior Knowledge:
- Logical, step-by-step thinking
- Variable assignment
- Expression evaluation
- Debugging techniques

Science Practices: [Copied from: 3]	Core Ideas: [Copied from: 4]	Cross Cutting Concepts: [Copied from: 5]
• Planning and carrying out investigations • Constructing explanations and designing solutions	• Developing possible solutions • Optimizing the design solution	• Systems and system models • Structure and function

Possible Preconceptions/Misconceptions:
Students should be able to complete this lesson successfully, save for the Bonus section of the *Investigation and Programming* exploration worksheet. Since this lesson does not explicitly discuss the utilization of modules, expect the students to have partially correct answers for using the math module even though the proper statements are provided in the prompting worksheet.

Lesson Plan - 5E(+) Model

Engage:
The instructor will hand out the half sheets titled *Temperature Program* and give each pair of students access to tablets or computers that run IDLE and the latest update of Python. The students will upload or copy the *Temperature Program: Script* code into the script mode of IDLE and run the code. By interacting with the code, students will be able to answer the questions about what the built-in functions do. Students should be given up to 7 minutes to work with a partner to complete the half sheet. The purpose of this engage is to introduce the fact there are two distinct types of functions in Python - built-in, and user-defined. The half sheet forces students to engage with the built-in functions, which serves as a precursor to learning about and utilizing user-defined functions. The students will be engaged in a whole class discussion that is facilitated by the instructor after completing the half sheet to review students' responses and address any misconceptions as necessary. This section of the lesson should last up to 10 minutes.

Explore:

Part I: Introduction

Students will be engaged in a whole class discussion where the teacher will talk to the students about the two types of functions - built-in versus user defined. The students will understand that the functions they have encountered so far are built-in functions. They will hypothesize what a user defined function is and how they might be able to utilize it. Reference the "Teacher Discussion Guide" at the bottom of the *Temperature Program* answer key. This discussion should last about 5 minutes.

Part II: *Benchmark Lesson*: Debugging and Evaluating

The instructor will hand out the scripts titled *Tip and Tax Functions: Script* and give each pair of students access to tablets or computers that run IDLE and the latest update of Python. The students will upload or copy the code into the script mode of IDLE and run the code. By interacting with the code, students will be able to answer the questions on the associated worksheets. Students will work in pairs on the worksheets *Tip and Tax Functions: Part 01: Debugging* and *Tip* and *Tax Functions: Part 02: Evaluation*. These worksheets require that students work with a given code and debug it. The questions associated with Part 01 inquire which lines are throwing the errors and inquire how the students were able to debug the code for each of the errors. The questions in Part 02 ask students to discuss the functions of each line of the code once the errors are resolved and to review variable assignments through function calling. These parts of the exploration allow students to achieve the learning goals of this lesson by working collaboratively and recursively. The instructor should facilitate an open class discussion to review student's responses and address any misconceptions as necessary after completing each part of the worksheets. In total, this should take up to 25 minutes to complete.

Part III: *Investigation Lesson*: Programming

Students will work individually on the worksheet *Investigation and Programming*. This worksheet provides the students with the RAFT of the unit, which is written in the description section of this lesson plan. This worksheet contains a prompt that asks students to design their own program from scratch. This prompt has them utilize built-in functions, user defined functions, and variable assignments. The bonus activity asks the students use the math module. The purpose of this part of the exploration is to have students create a Python code from scratch so they can work through a real-world example by developing the skills they have learned from the beginning of this curriculum. The remaining class time should be allotted for the students to complete writing this program.

Explain:

Throughout the exploration, the students will engage in discussions that inquire their understanding and knowledge of the information at-hand. Thus, teachers will be informally asking students to explain all topics and relevant connections throughout the entirety of this lesson. The worksheets include "why" and "how" questions to provide the teacher with their individual explanations and to also give the students an opportunity to verbalize their understandings. The exploration also includes questions that prompt students to defend their responses and explain why they chose a particular solution.

Elaborate:
The elaboration of this lesson is the investigation lesson section. Computer programming is currently a prominent and relevant career path that requires developers to tackle problems by debugging current code and writing new codes with helpful functions. The *Investigation and Programming* worksheet introduces the students to these programming skills. This student-designed program forces the students to think like a programmer by engaging in real-world scenarios.

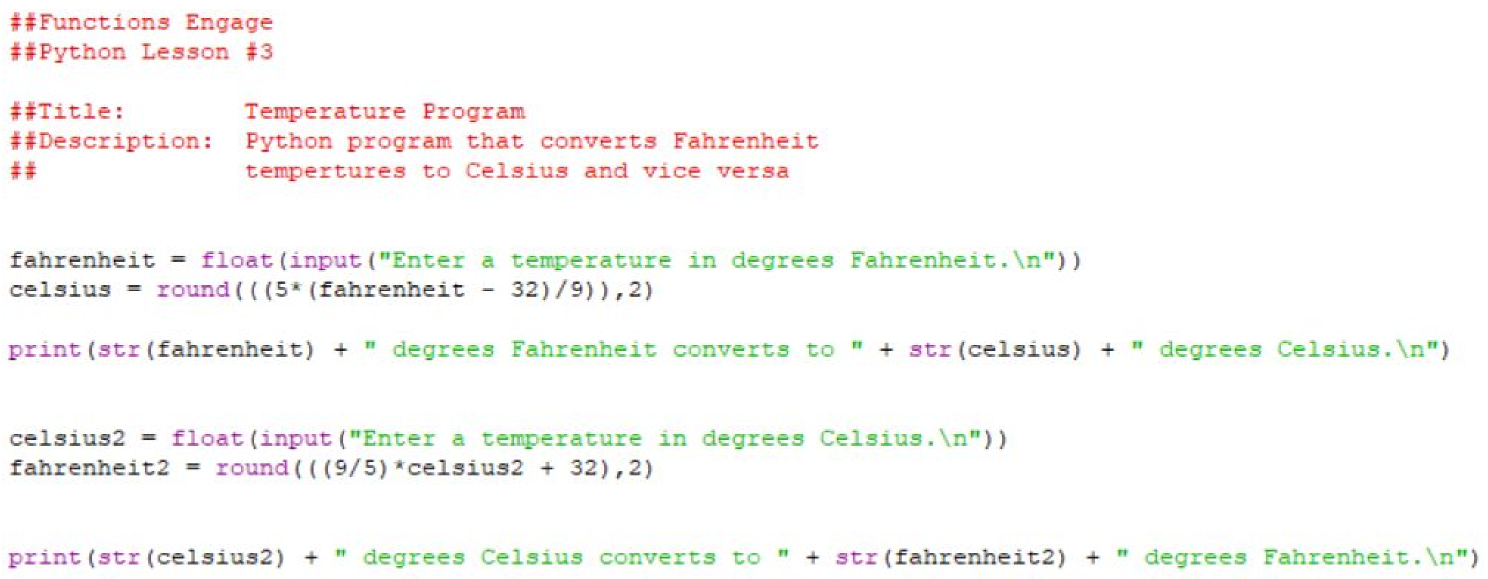
Evaluate:
This lesson is designed with having both informal and formal evaluations throughout its entirety. The informal evaluations occur throughout the exploration because of the leading and open-ended questions and class discussions. This allows teachers to gauge surface-level student understanding. By surveying the students during completion of the worksheets, teachers will be able to hear and address any misconceptions or misunderstandings as necessary. The formal evaluation of this lesson is the *Investigation and Programming* worksheet. This individual activity forces students to write their own code from beginning to end by pulling on information learned throughout this lesson and both prior Python lessons.

Enrich:
This lesson could be extended by introducing how to utilize conditional statements and loops. Since most functions contain complicated conditionals and nested loops, the next logical step is to have students interact and explore Python code that includes functions with conditional statements and loops. Further, functions are most commonly used in industry when programmers realize that certain parts of codes are utilized in a multitude of programs. Hence, programmers tend to write a code containing functions that can be called and utilized in other program codes. Having students to learn conditionals and loops primes them to developing necessary skills that programmers use daily.

All associated documents are attached below
Reference *Annotated Bibliography* on the very last page of this packet

Name: ____ Date: ________________

Temperature Program: Script



```
##Functions Engage
##Python Lesson #3

##Title:        Temperature Program
##Description:  Python program that converts Fahrenheit
##              tempertures to Celsius and vice versa


fahrenheit = float(input("Enter a temperature in degrees Fahrenheit.\n"))
celsius = round(((5*(fahrenheit - 32)/9)),2)

print(str(fahrenheit) + " degrees Fahrenheit converts to " + str(celsius) + " degrees Celsius.\n")


celsius2 = float(input("Enter a temperature in degrees Celsius.\n"))
fahrenheit2 = round(((9/5)*celsius2 + 32),2)


print(str(celsius2) + " degrees Celsius converts to " + str(fahrenheit2) + " degrees Fahrenheit.\n")
```

Name: ____ Date: ________________

Temperature Program

1.) What does the input( ) function do?

2.) What does the float( ) function do?

3.) What does the print( ) function do?

4.) What does the round(,2) function do?

Bonus! What does the str( ) function do? Why do we need it?

Name: ____ Date: ________________

Temperature Program

1.) What does the input( ) function do?

2.) What does the float( ) function do?

3.) What does the print( ) function do?

4.) What does the round(,2) function do?

Bonus! What does the str( ) function do? Why do we need it?

Name: ________________Teacher Copy ONLY________________ Date: _______________

Tip and Tax Functions: Script (Without Bugs)

```
##Functions Explore
##Python Lesson #3

##Title:        Tax and Tip
##Description:  Python program that provides the bill
##              total after both tax and tip are
##              calculated.


##Function to calculate the bill total with 8% tax

def tax(bill):
    bill *= 1.08
    print("The bill's total with 8" + '%' + " tax is: $ %.2f" %bill)
    return bill

##Function to calculate the bill total with 20% tip

def tip(bill):
    bill *= 1.20
    print("The bill's total with 20" + '%' + " tip is: $ %.2f" %bill)
    return bill

mealCost = float(input("How much did your meal cost?\n"))

mealWithTax = tax(mealCost)
mealWithTip = tip(mealWithTax)
```

Name: ____ Date: ________________

Tip and Tax Functions: Script

```
##Functions Explore
##Python Lesson #3

##Title:         Tax and Tip
##Description:   Python program that provides the bill
##               total after both tax and tip are
##               calculated.


##Function to calculate the bill total with 8% tax

def tax(bill):
    bill *= 1.08
    print("The bill's total with 8% tax is: $ %.2f") %bill
    return bill

##Function to calculate the bill total with 20% tip

def tip(bill):
    bill *= 1.20
    print("The bill's total with 20% tip is: $ %.2f") %bill
    return bill

mealCost = float(input("How much did your meal cost?\n"))

mealWithTax = tax(mealCost)
mealWithTip = tip(mealWithTax)
```

Name: ____ Date: ________________

Tip and Tax Functions: Part 01: Debugging

Directions: Copy and Paste or upload the “Tip and Tax Functions” Python file into the script mode of IDLE. Work with a partner and use the code to answer all the following questions.

1.) When prompted with the error:

    Type Error: unsupported operand type(s) for %: ‘None Type’ and ‘float’

    a.) What line(s) of code are throwing that error?

    b.) How did you resolve this error? Provide an explanation and the valid code changes.

2.) When prompted with the error:

    Value Error: unsupported format character ‘t’ (0x74) at index 25

    a.) What line(s) of code are throwing that error?

    b.) How did you resolve this error? Provide an explanation and the valid code changes.

Name: ____ Date: ________________

Tip and Tax Functions: Part 02: Evaluating

1.) What does each line of the following function do? Define and describe all key concepts.

```
def tax(bill):
    bill *= 1.08
    print("The bill's total with 8" + '%' + " tax is: $ %.2f" %bill)
    return bill
```

2.) Why do we use 1.08 for the tax but not 0.08?

3.) What does each line of the following variable assignments do?

```
mealCost = float(input("How much did your meal cost?\n"))
mealWithTax = tax(mealCost)
mealWithTip = tip(mealWithTax)
```

Bonus! What would happen if the “return bill” statements were not there?

Name: ____ Date: ________________

Investigation and Programming

Directions:

Your role as an assistant computer programmer means that you must learn to be proficient in many types of programming and coding. For this assignment you will have to learn how to use a high-level programming tool known as Python. Because you are in training, you will be working on several modules and continue to build upon your skills. You will work on various skills in Python and submit them to the head computer programmer to make sure your codes are both sufficient and efficient. For this task, you will learn and utilize functions and modules in Python by designing your own program.

Prompt:

JP Morgan and Chase is an investment banking company. The employers enjoy celebrating their employees' birthdays, but no one likes to sing "Happy Birthday!" JP Morgan and Chase hired you to write a Python code that can be programmed into their stereo system so the "Happy Birthday!" song can be played during the celebrations. They provided the following rules:

1. One user defined function must print the "Happy Birthday!" song that takes the employee's name as an argument.
2. One user defined function that asks the user for the name of the employee and that calls the birthday song function from number 1.

Bonus!

As an investment banking company, JP Morgan and Chase computes compounding interest for their clients. They hired you to design a Python program to handle these computations more easily. They provided the following rules:

1. One user defined function must take four arguments: principle, interest rate, compounding period, and number of years. This function must complete the appropriate computation with those variables.
2. One user defined function that asks the user for the necessary variables for number 1 and that calls the investment balance function.
3. To verify that your investment balance function is computing properly, use the following example:
    a. If a principle amount of $5,000 is deposited into a savings account at an interest rate of 5% and that is compounded monthly, what is the value of the investment after 10 years?
    Answer: A = 5000(1 + (0.05/12)) ^ (12*10) = $8,235.05

Here are some helpful tips for you:

1. Compounding Interest formula: $A = P(1 + \frac{r}{n})^{n \cdot t}$
2. Math Module into script mode: import math
3. Exponents: math.pow(x,y) (where x is the base and y is the power)

Name: ___________________ANSWER KEY____________________ Date: ________________

Temperature Program

1.) What does the input( ) function do?
The input function prompts the user with a string and takes the information that the user types into the keyboard. The keyboard information is taken into the computer program and utilized as part of the code.

2.) What does the float( ) function do?
The float function changes the variable type of the keyboard input from a string variable to a float variable. We need this variable change because of the necessary calculations in the program.

3.) What does the print( ) function do?
The print function presents a string onto the computer screen for the user to observe the output of the program.

4.) What does the round(,2) function do?
The round function takes a float variable and "cuts off" the fractional part, i.e. if a number is a decimal, then the whole number is the result. When the function is used as round(, n), then the round functions "cuts off" at the nth decimal place. Hence, round(,2) rounds the input to have two values after the decimal.

Bonus! What does the str( ) function do? Why do we need it?
The string function changes the variable type of the input to a string variable. We need this because a print statement can only present a string variable. Since the program needs numbers to be displayed, then these numbered variable types need to be converted into the string variable type.

Teacher Discussion Guide

All of the functions utilized in the temperature program script are built-in functions. These are functions that are provided by the Python library. These are not the only types of functions, however. There are also user defined functions in Python. In general, a function is a group of statements that perform a helpful task. Hence, a user defined function is a group of statements created by the programmer that performs a specific task within their program. One of the main reasons for defining functions is that they help break out programs into smaller, more sensible chunks. Utilizing functions is a great way to keep a code organized and manageable by avoiding repetition and making the code reusable.

Name: __________________ANSWER KEY___________________ Date: ________________

Tip and Tax Functions: Part 01: Debugging

Directions: Copy and Paste or upload the "Tip and Tax Functions" Python file into the script mode of IDLE. Work with a partner and use the code to answer all the following questions.

1.) When prompted with the error:

Type Error: unsupported operand type(s) for %: 'None Type' and 'float'

a.) What line(s) of code are throwing that error?
The print statements: print("The bill's total with 8% tax is: $ %.2f") %bill and print("The bill's total with 20% tip is: $ %.2f") %bill

b.) How did you resolve this error? Provide an explanation and the valid code changes.
Because the percent sign is a special symbol in Python that is responsible for formatting purposes, it cannot be inside the print statement this way. Hence, we have to be explicit to the computer that we want the percent sign to have the type of string and to be printed on the screen. The appropriate resolution is to use string concatenation within the print statement. The corrections are as follows: print("The bill's total with 8" + "%" + "tax is %.2f") %bill and print("The bill's total with 8" + "%" + "tax is %.2f") %bill.

2.) When prompted with the error:

Value Error: unsupported format character 't' (0x74) at index 25

a.) What line(s) of code are throwing that error?
The print statements: print("The bill's total with 8" + "%" + "tax is %.2f") %bill and print("The bill's total with 8" + "%" + "tax is %.2f") %bill

b.) How did you resolve this error? Provide an explanation and the valid code changes.
The percent sign acts as a string formatter in Python. Because it's a special character, using the percent sign means following the proper syntax. The corrections are as follows: print("The bill's total with 8" + "%" + "tax is %.2f" %bill) and print("The bill's total with 8" + "%" + "tax is %.2f" %bill).

Name: ___________________ANSWER KEY____________________ Date: ________________

Tip and Tax Functions: Part 02: Evaluating

1.) What does each line of the following function do? Define and describe all key concepts.

```python
def tax(bill):
    bill *= 1.08
    print("The bill's total with 8" + '%' + " tax is: $ %.2f" %bill)
    return bill
```

The keyword "def" marks the start of a function header. The title "tax" is the function name used to uniquely identify the function. The parameter of the function is "bill," which passes values into the function. Hence, the first line of the code defines the function and its parameter. The second line adds the 8% tax onto the total bill amount and reassigns that new total to the variable "bill". The third line prints a sentence as an output for the user to see the new bill total with the added tax. In this print statement, the percent sign is utilized as a string formatter. The "%.2f %bill" means that the new bill total is changed from a float with two values after the decimal into a string so that it can be used in the print statement (remember, print statement only function with variable type string). The last line returns the new bill amount from the function. The return statement is used to exit a function and go back to the place from where it was called. In this case, it returns into the mealWithTax variable (see description in number 3).

2.) Why do we use 1.08 for the tax but not 0.08?
When computing tax mathematically, we would say that the newBillPrice = billPrice + billPrice * 0.08. Hence, the statement can be rewritten as newBillPrice = billPrice * 1.08 by basic math principles. Recall the that the "bill *= 1.08" in Python means the same thing as "bill = bill * 1.08.

3.) What does each line of the following variable assignments do?

```python
mealCost = float(input("How much did your meal cost?\n"))
mealWithTax = tax(mealCost)
mealWithTip = tip(mealWithTax)
```

mealCost is a variable assignment that requires the user to input the total amount of the bill and it converts the input from a string variable into a float variable. mealWithTax and mealWithTip are variable assignments that are calling the tax and tip functions, respectively. In the statement mealWithTax = tax(mealCost), mealCost is the argument of the tax function call, which is put into the parameter of the tax function, "bill". Hence, the "return bill" statement assigns the calculated bill total with tax into the mealWithTax variable. The same logic and description applies to mealWithTip.

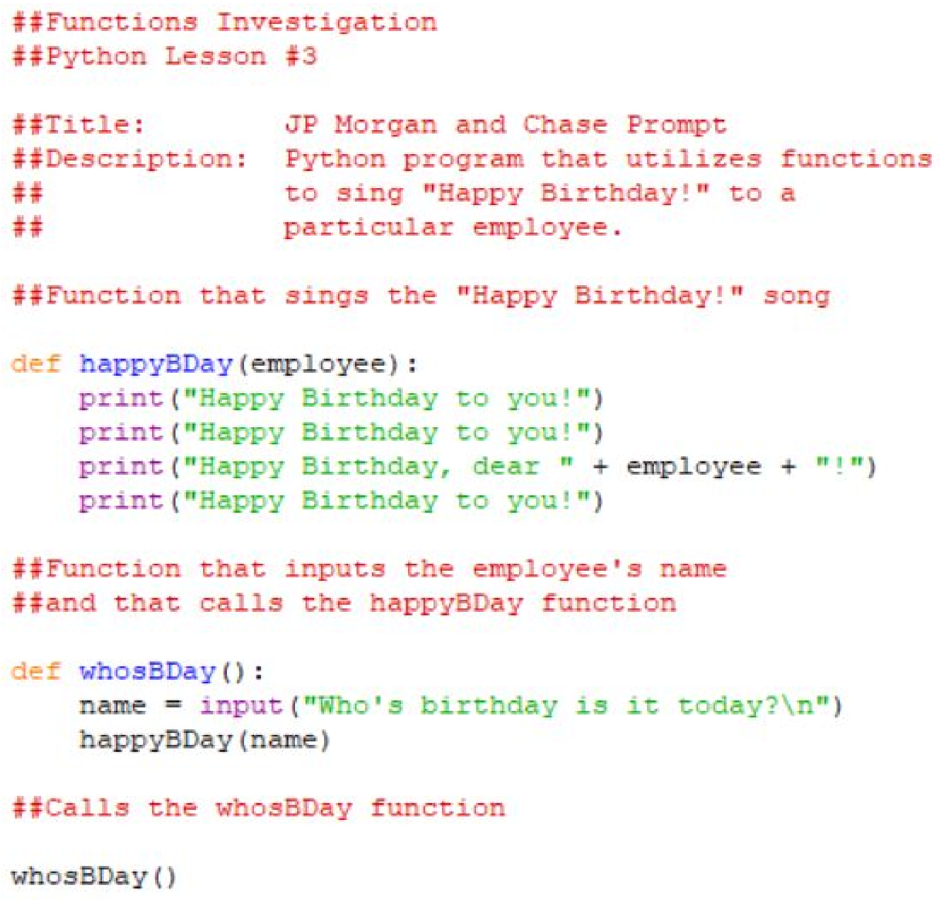
Bonus! What would happen if the “return bill” statements were not there?

If the “return bill” statements were not present, then the program would not output the proper values and an error will occur once the execution reaches the variable assignment lines. This is because nothing would be returned into the associated variable assignments.

Name: ____________________ANSWER KEY_____________________ Date: ________________

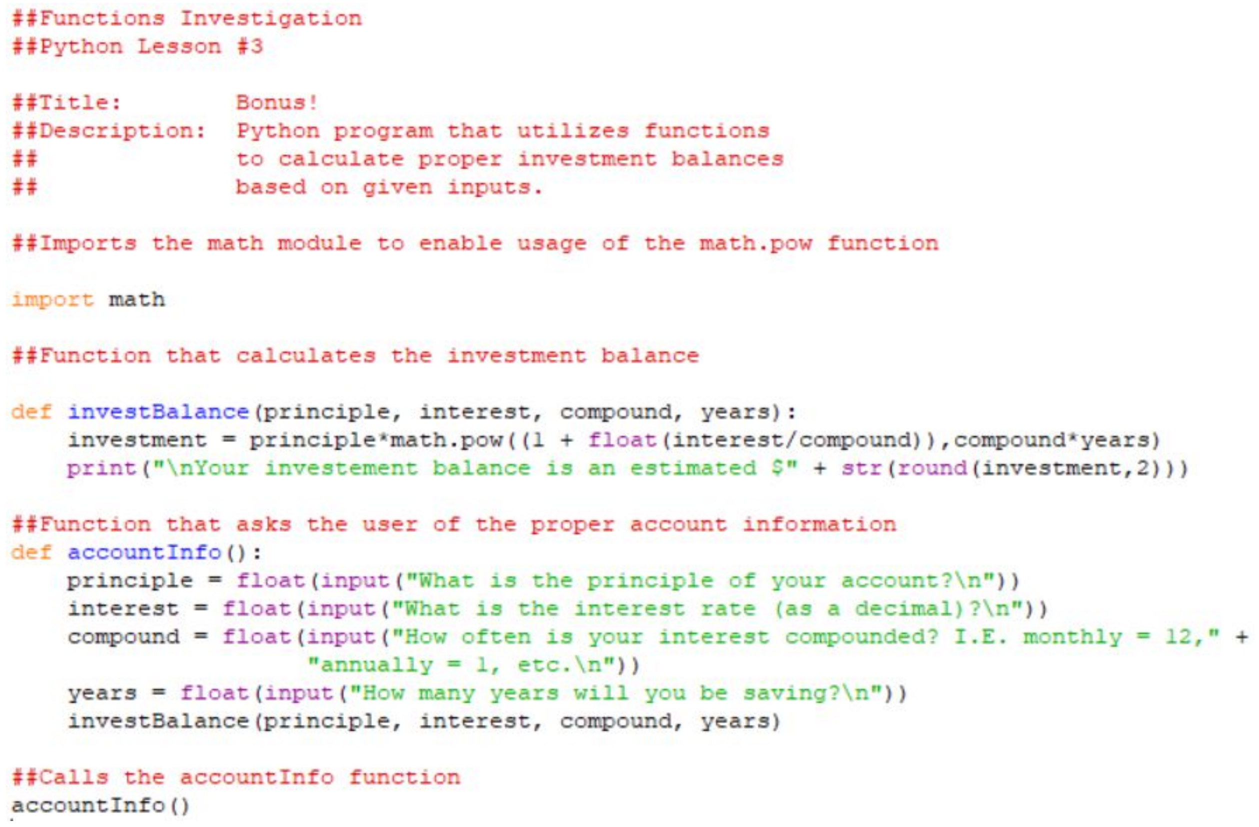
Investigation and Programming

JP Morgan and Chase Prompt



```python
##Functions Investigation
##Python Lesson #3

##Title:         JP Morgan and Chase Prompt
##Description:   Python program that utilizes functions
##               to sing "Happy Birthday!" to a
##               particular employee.

##Function that sings the "Happy Birthday!" song

def happyBDay(employee):
    print("Happy Birthday to you!")
    print("Happy Birthday to you!")
    print("Happy Birthday, dear " + employee + "!")
    print("Happy Birthday to you!")

##Function that inputs the employee's name
##and that calls the happyBDay function

def whosBDay():
    name = input("Who's birthday is it today?\n")
    happyBDay(name)

##Calls the whosBDay function

whosBDay()
```

Bonus! Code



```python
##Functions Investigation
##Python Lesson #3

##Title:        Bonus!
##Description:  Python program that utilizes functions
##              to calculate proper investment balances
##              based on given inputs.

##Imports the math module to enable usage of the math.pow function

import math

##Function that calculates the investment balance

def investBalance(principle, interest, compound, years):
    investment = principle*math.pow((1 + float(interest/compound)),compound*years)
    print("\nYour investement balance is an estimated $" + str(round(investment,2)))

##Function that asks the user of the proper account information
def accountInfo():
    principle = float(input("What is the principle of your account?\n"))
    interest = float(input("What is the interest rate (as a decimal)?\n"))
    compound = float(input("How often is your interest compounded? I.E. monthly = 12," +
                     "annually = 1, etc.\n"))
    years = float(input("How many years will you be saving?\n"))
    investBalance(principle, interest, compound, years)

##Calls the accountInfo function
accountInfo()
```

Annotated Bibliography

[1] Downey, A. (2012). *Think Python: How to Think Like a Computer Scientist* (Vol. 2.0.17). Needham, MA: Green Tea Press. From http://www.greenteapress.com/thinkpython/thinkpython.pdf

This online textbook was used for excerption within the Python Functions and Methods lesson plan as part of the Computer Science module. This reference aided in the completion of providing definitions for the key concepts and definitions sections and for the associated worksheets. This book was useful because of its layout and completeness. The lesson expands upon the material used form this book as it uses the material in the creation of worksheets and activities that are not provided in the textbook.

[2] Standards Aligned System. (n.d.). Retrieved from https://www.pdesas.org/

This website was used in each lesson in the Computer Science module to select proper Pennsylvania State standards, which are based in Common Core, that each lesson is centered around.

[3] Nsta. (n.d.). Science and Engineering Practices. Retrieved January 18, 2019, from https://ngss.nsta.org/PracticesFull.aspx

This website used in every lesson in the Computer Science module to find Standards for Scientific Practices that are applicable in each lesson.

[4] Nsta. (n.d.). Disciplinary Core Ideas. Retrieved from https://ngss.nsta.org/DisciplinaryCoreIdeasTop.aspx

This website was used in each lesson in the Computer Science module to select appropriate disciplinary core ideas set forth by the NSTA that are at the center of each lesson.

[5] Nsta. (n.d.). Crosscutting Concepts. Retrieved from https://ngss.nsta.org/CrosscuttingConceptsFull.aspx

This website was used in each lesson in the Computer Science module to selecting appropriate crosscutting concepts set forth by the NSTA that apply to each Python lesson.