Assignment 2 – Writing Programs

Python Intro

We need to cover some basic Python skills before getting started.

Indenting

Unlike most other programming, indenting (typically 4 spaces) matters in python.

1. Code the following
    a. You've seen the print() statement and can probably logic through most of the code but if not, don't worry about it, just note the error due to improper indenting

```python
#! /usr/bin/env python3

import random

randomNumber = random.randint(1,9)
yourGuess = 0
count = 0

while yourGuess != randomNumber and yourGuess != "exit":
    yourGuess = input("Enter a guess between 1 to 9 or exit to end the game ")

    if yourGuess == "exit":
        break

    yourGuess = int(yourGuess)
    count += 1

    if yourGuess < randomNumber:
        print("Too low")
        elif yourGuess > randomNumber:
        print("Too high")
    else:
        print("Right!")
        print("You took only", count, "tries!")
input()
```

2. Fix the indenting and run the program to prove it works.
   Screen Capture #1 (2 point)

```
count = 0

while yourGuess != randomNumber and yourGuess != "exit":
    yourGuess = input("Enter a guess between 1 to 9 or exit to end the game ")

    if yourGuess == "exit":
        break

    yourGuess = int(yourGuess)
    count += 1

    if yourGuess < randomNumber:
        print("Too low")
    elif yourGuess > randomNumber:
        print("Too high")
    else:
        print("Right!")
        print("You took only", count, "tries!")
input()
```

```
Run: test  test  test
C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\python.exe C:/Users/Sa
Enter a guess between 1 to 9 or exit to end the game 5
Too high
Enter a guess between 1 to 9 or exit to end the game 3
Too high
Enter a guess between 1 to 9 or exit to end the game 1
Too low
Enter a guess between 1 to 9 or exit to end the game 2
Right!
You took only 4 tries!
```

While it is not imperative you understand all the syntax that made up this app, there are a few things you should be aware of:

- Each line of our app is called a statement and each statement performs a task.
- The first line of the app is called the shebang line and it starts with the hash (#) and bang (!) and is used to identify which interpreter to use.
    - It is not required in Windows apps but is considered good practice.

Comments

- Comments start with the # symbol and are ignored by the compiler.
    a. The # can be used to comment out a full line or inline to comment out everything after the # symbol.
- Comments are used to document what a portion of the code does.

3. Add the following comments:
   Screen Capture #2 (2 point)

```python
#! /usr/bin/env python3

import random

# Get a random number between 1 and 9
randomNumber = random.randint(1, 9)
yourGuess = 0
count = 0

# Continue guessing until the guess is correct or the user types 'exit'
while yourGuess != randomNumber and yourGuess != "exit":
    yourGuess = input("Enter a guess between 1 to 9 or 'exit' to end the game: ")

    # If the user types 'exit', end the application
    if yourGuess == "exit":
        break

    # Cast the user's guess to an integer so it
    # can be used in the comparison below
    yourGuess = int(yourGuess)
    count += 1

    if yourGuess < randomNumber:
        print("Too low")
    elif yourGuess > randomNumber:
        print("Too high")
    else:
        print("Right!")
        print("You took only", count, "tries!")

input()  # This will pause the app from exiting when running from command line
```

Functions

Functions are pieces of reusable code that performs a specific task. There are built-in functions, like the *print()* and *input()* and you can also create user defined functions.

We'll be creating our own later but for now we'll just work with some of the built-in functions.

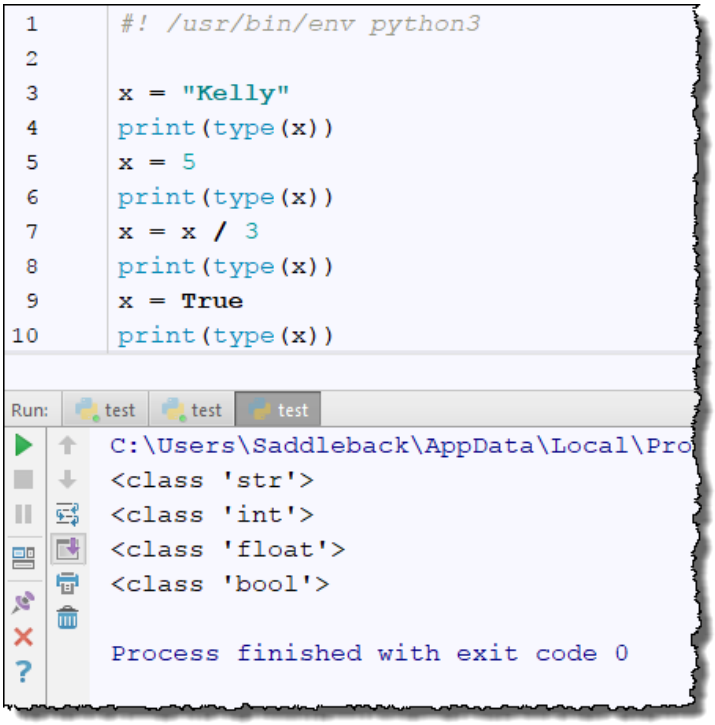
Variables and Data Types

We use variables to store data and you might not have realized but we've already been using variables throughout this assignment. In our example, random_number, your_guess and count are all variables that we use to hold values we'll later use in our application.

Unlike most popular languages today, Python is known as "Dynamically Typed". What that means is in languages like C#, Java, etc., when variables are created, they are created with a data type (i.e. string, whole number, decimal number, boolean, etc.). But Python is different as in our previous example, we just created the variables and the data type gets set when we assign a value to it. On top of that, it's possible for the variable to change data type based on the assignment.

4. Code as follows:
    a. type() is a Python function that returns the data type if a variable

Screen Capture #3 (1 point)



Naming Variables

Python has rules for naming variables:

- Must begin with a letter or underscore
- Can't contain spaces, punctuation, or special characters other than the underscore
- Can't begin with a number, but can have numbers in the name
- Can't be the same as a Python reserved keyword.

Additionally, there are recommendations for naming Python variables.

- Start with lowercase letters
- Use underscore notations or camel case. (i.e. camel_case or camelCase)
- Use meaningful names
- Don't use Python built-in names (i.e. type(), print(), etc.)

Arithmetic Expressions

5. Code as follows:

   Screen Capture #4 (2 point) – Include both code and running image.

```python
#! /usr/bin/env python3

# Basic arithmetic operators
print(7 + 3)
print(7 - 3)
print(7 * 3)
print(7 / 3)

# Two operands
print(7 // 3)    # Truncates decimal
print(7 ** 3)     # Raises to the power of

# Modulo operator
print(7 % 3)     # Returns the remainder of

# Order of precedence (Default)
# Multiplication and division will happen first
print(7 + 3 * 5)

# User specified order
print((7 + 3) * 5)   # User () to specify order
```

```
Run: Assignment 2
C:\Users\Saddleback\AppData\Local\Programs\Python\Pyth
10
4
21
2.3333333333333335
2
343
1
22
50
```

Assignment Operators

6. Code as follows:

Screen Capture #5 (1 point)

```
#! /usr/bin/env python

# Assignment Operators
diameter = 5    # Integer
pi = 3.141592   # float
circumference = diameter * pi

print("The circumference is " + str(circumference))
```

```
Run: circumference
C:\Users\Saddleback\AppData\Local\Programs\Python\Pytho
The circumference is 15.70796

Process finished with exit code 0
```

Incrementing and Decrementing a Number

7. Code as follows:

Screen Capture #6 (1 point)

```
#! /usr/bin/env python3

# Incrementing
counter = 0
print("Incrementing starting point: " + str(counter))
counter = counter + 1
print("counter = " + str(counter))
counter += 1
print("counter = " + str(counter))

# Decrementing
counter = 10
print("Decrementing starting point: " + str(counter))
counter = counter - 1
print("counter = " + str(counter))
counter -= 1
print("counter = " + str(counter))
```

```
Run: test  test  test
C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\p
Incrementing starting point: 0
counter = 1
counter = 2
Decrementing starting point: 10
counter = 9
counter = 8

Process finished with exit code 0
```

Strings

Notice in the previous example, we used the built-in python *str()* function to convert a numeric argument to string.

String Assignment

8. Code as follows:
    a. Use your own name and occupation here:

```
#! /usr/bin/env python3

# Can use single or double quotes
name = "Kelly"
occupation = 'Teacher'

```

Concatenating (joining) Stings

9. Add the results line and display the output.

Screen Capture #7 (1 point)

```
#! /usr/bin/env python3

# Can use single or double quotes
name = "Kelly"
occupation = 'Teacher'

results = name + " is a " + occupation

print(results)
```

```
Run: test  test  test
C:\Users\Saddleback\AppData\Local\Programs\
Kelly is a Teacher

Process finished with exit code 0
```

10. Modify as follows:
    a. Notice the "\" which is performing a line continuation.

Screen Capture #8 (1 point)

```
#! /usr/bin/env python3

# Can use single or double quotes
name = "Kelly"
occupation = 'Teacher'
tenure = 6

results = name + " is a " + occupation + \
    " and has been for " + str(tenure) + " years"

print(results)
```

```
Run: test  test  test
C:\Users\Saddleback\AppData\Local\Programs\Python\Py
Kelly is a Teacher and has been for 6 years

Process finished with exit code 0
```

Special Characters in Strings

11. Modify as follows:
    b. \n for new line
    c. \' for single quote
    d. \" for double quote
    e. \\ for slash (not shown)

Screen Capture #9 (1 point)

```
#! /usr/bin/env python3

# Can use single or double quotes
name = "Kelly"
occupation = 'Teacher'
tenure = 6
location = "Saddleback College"

results = "\"" + name + "\" is a " + occupation + \
    " at " + location + \
    "\nand has been for over \'" + \
    str(tenure) + "\' years"

print(results)
```

```
Run: test  test  test
C:\Users\Saddleback\AppData\Local\Programs\Python\Pyth
"Kelly" is a Teacher at Saddleback College
and has been for over '6' years

Process finished with exit code 0
```

Python Built-In Functions

print()

We've been using print() from the beginning.

12. Modify as follows:
    a. The value being passed in is called an argument.

```
#! /usr/bin/env python3

# print()
name = "Homer"
print(name)   # name is an argument

```

13. Once again, modify as follows:

Screen Capture #10 (1 point)

```
#! /usr/bin/env python3

# print()
name = "Homer"
name2 = "Marge"
name3 = "Bart"
name4 = "Lisa"
name5 = "Maggie"
#  print(name)   # name is an argument

print(name, name2, name3, name4, name5)

Run: test  test  test
C:\Users\Saddleback\AppData\Local\Programs\P
Homer Marge Bart Lisa Maggie

Process finished with exit code 0
```

14. Modify to add a separator character

Screen Capture #11 (1 point)

```
#! /usr/bin/env python3

# print()
name = "Homer"
name2 = "Marge"
name3 = "Bart"
name4 = "Lisa"
name5 = "Maggie"
#  print(name)   # name is an argument

print(name, name2, name3, name4, name5, sep = ' | ')
```

```
Run: test  test  test
C:\Users\Saddleback\AppData\Local\Programs\Python\Pythor
Homer | Marge | Bart | Lisa | Maggie

Process finished with exit code 0
```

input()

The input causes the application to pause and wait for user input.

15. Code as follows:
    a. Use your name here.

Screen Capture #12 (1 point)

```
#! /usr/bin/env python3

# input()
name = input("Enter your name: ")
print("Hello " + name)
```

```
Run: test  test  test
C:\Users\Saddleback\AppData\Local\Program
Enter your name: Kelly
Hello Kelly

Process finished with exit code 0
```

16. Modify as follows:
    b. Use your name here.

Screen Capture #13 (1 point)

```
#! /usr/bin/env python3


# input()
print("Enter your name: ")
name = input()
print("Hello " + name)
```

```
Run: Hello_World
C:\Users\Saddleback\AppData\Local\Pr
Enter your name:
Kelly
Hello Kelly

Process finished with exit code 0
```

Conversions

We've already seen *str()* which converts numeric data to a string, these convert (typically string) data to numeric, int() to whole numbers and float() to decimal numbers

int() / float()

17. Code as follows:
    a. Probably not the results we wanted or were expecting.

Screen Capture #14 (1 point)

```
#! /usr/bin/env python3

# int()

x = "5"
y = 3
product = x * y

print(x + " * " + str(y) + " = " + str(product))
```

```
Run: test test test
C:\Users\Saddleback\AppData\Local\Programs\Python\Pyt
5 * 3 = 555

Process finished with exit code 0
```

18. Modify to convert x to an integer using the int() function.

Screen Capture #15 (1 point)

```
#! /usr/bin/env python3

# int()

x = "5"
y = 3
product = int(x) * y

print(str(x) + " * " + str(y) + " = " + str(product))
```

```
Run: test test test
C:\Users\Saddleback\AppData\Local\Programs\Python\Pytho
5 * 3 = 15

Process finished with exit code 0
```

Test Scores Application

Create an application the excepts test scores from a user. Allow the user to enter scores until an exit condition is met. (In this case, the user types 'Exit'). Then display the number of scores, total of all the scores and average of all the scores.

Start by creating a python file named **test_scores.py** and add code to complete the application. Try to complete this on you own but I have included the code on the next page if you get stumped.

Make sure you attach the test_scores.py file when submitting your assignment.

Code Validation #1 (1 point)
Screen Capture #16 (1 point)

```
C:\Users\Saddleback\assignment_02\Sc
The Test Scores Program

Enter 3 test scores
==========================
Enter test score: 90
Enter test score: 100
Enter test score: 80
==========================
Total Score:   270
Average Score:  90

Bye

Process finished with exit code 0
```

Test Scores Program code:

test_scores.py

```python
#! /usr/bin/usr python3

# display a welcome message
print("The Test Scores Program\n")
print("Enter 3 test scores")
print("=" * 25)

# get scores from the user
total_score = 0
total_score += int(input("Enter test score: "))
total_score += int(input("Enter test score: "))
total_score += int(input("Enter test score: "))

# calculate the average
average_score = round(total_score / 3)

# format and display the results
print("=" * 25)
print("Total Score:  ", total_score,
      "\nAverage Score: ", average_score)

# end the application
print("\nBye")
```

Extra Credit (1 point)

Add the ability to add scores until the user enters ‘exit’

```
The Test Scores Program

Enter Test scores
Enter 'Exit' to Quit
========================
Enter test score: 70
Enter test score: 95
Enter test score: 100
Enter test score: 80
Enter test score: 90
Enter test score: Exit
========================
# of Score:  5
Total Score:  435
Average Score:  87

Process finished with exit code 0
```

Extra Credit

To get full points for each extra credit, you must include screen captures of the running output as well as the python (.py) code files.

Extra Credit #1 – Student Registration (+1 Extra Credit)

Complete a program that allows a student to complete a registration form and display a completion message that includes the user's full name and temporary password.

```
Registration Form

First name:     Eric
Last name:      Idle
Birth year:     1934

Welcome Eric Idle!
Your registration is complete.
Your temporary password is: Eric*1934
```

Specifications:

- The user's full name consists of the user's first name, a space, and the user's last name.
- The temporary password consists of the user's first name, an asterisk (*), and the user's birth year.
- Assume the user will enter valid data.

Extra Credit #2 – Travel Time Calculator (+1 Extra Credit)

Create a program that calculates the estimated hours and minutes for a trip

```
Travel Time Calculator

Enter miles: 200
Enter miles per hour: 65

Estimated travel time
Hours: 3
Minutes: 5
```

Specifications:

- The program should only accept integer entries like 200 and 65.
- Assume the user will enter valid data.

Hint

- Use integers with the integer division and modulus operators to get hours and minutes.