Lab 1 – Basic I/O, Strings, & Formatting

Topics Covered:

- Introduction to PyCharm
- Printing and simple arithmetic
- Assigning Values to Variables, Arithmetic Expressions, Operator Precedence
- Boolean Expressions
- Strings & Concatenation
- Printing, Formatting Strings, Escape Sequences
- Conversion Types, Casting

Perquisites

Before attending the lab, please answer the pre-lab questions and bring your solutions to the prelab questions (the TA will check these). If you will be using a laptop during the lab session, please make sure you have Python and an IDE installed. Any Python IDE is acceptable, but PyCharm is recommend.

How Labs are Run

You are expected to complete part 1 (the prelab questions) at home before attending the lab section and to work on the questions in part 2 **during** the lab session. After you have completed the lab during the session or if the lab is coming close to the end time, your TA will review your work and assign a pass/fail grade. To pass a lab you must:

1) Attend the lab session you are registered in (you can not go to any other lab section)
2) Arrive to the lab session on time.
3) Have the pre-lab questions completed.
4) Make a good effort towards working on the lab (part 2) during the lab session.
5) Remain in the lab until the TA reviews your work and assigns you a grade (pass/fail).

Arriving late, completing the full lab at home, or not making a good effort to complete the lab during the lab session, may result in the TA entering a failing lab grade.

Make sure you record your answers to all parts and have your code ready to demonstrate to the TA once you have completed the lab. When you are done put up your hand and the TA will review your work. You are NOT required to submit your work, you just need to have the TA record your pass/fail grade.

Part 1: Pre-Lab Questions

Answer the following questions before the Lab. Test your solutions out in Python to ensure they are correct!

Question 1: What function is used to print/output a message to the screen?

a) output()
b) sysprint()
c) print()
d) systemprint()

Question 2: Which of the following will correctly print "Hello World!"?

a) print("Hello World!")
b) print(Hello World!)
c) sysprint("Hello World!")
d) sysprint(Hello World!)

Question 3: Take a look at the following line of code. Here we are creating a variable *answer* that will store the value of *4+8.* What is the correct way to print this variable?

```
answer = 4+8
```

a) print("answer")
b) print(answer)

Question 4: Take a look at the following line of code. Here we are creating a variable *x* that will store the value of *6.* Which one of the following print statements will add 4 to *x*, and multiply the result by 8?

```
x = 6
```

a) print(x+4*8)
b) print(x*4+8)
c) print((x+4)*8)

Question 5: For the following print statement, determine whether the output will be True, False, or an error.

```
print(3>6)
```

a) True
b) False
c) Error

Question 6: For the following code segment, determine whether the output will be True, False, or an error.

```
x = 3
print(x=3)
```

a) True
b) False
c) Error

Question 7: For the following print statement, determine whether the output will be True, False, or an error.

```
print(7==8)
```

a) True
b) False
c) Error

Question 8: For the following code segment, determine whether the output will be True, False, or an error.

```
x = 5
print(x+3*0>2)
```

a) True
b) False
c) Error

Question 9: Which of the following lines of code will produce this output?

```
The car went,
"vroom!"
```

a) print("The car went, \n"vroom!")
b) print("The car went, \n\vroom!\")
c) print("The car went, \n\"vroom!\")
d) print("The car went, \n\"vroom!\"")

Question 10: Look at the following variables. We want to format a string to output these values. Which of the following correctly formats the string using the **f-string method** so that the output is *Becky 32 $10.32*?

```
name = "Becky"
age = 32
balance = 10.32
```

a) print(f"{name} {age} ${balance}")
b) print(name + " " + age + "$" + balance)
c) print(f"{Becky} {32} ${10.32}")
d) print(name, age, balance)

Question 11: Look at the following variables. We want to format a string to output each value with **two decimal places**. Which of the following correctly formats the output such that it is *42.00 3.14 3.50*

```
x = 42
y = 3.141592653589
z = 3.50
```

a) print(f"%2f %2f %2f")
b) print(f"{x} {y} {z}")
c) print(f"{x:.0f} {y:.0f} {z:.0f}")
d) print(f"{x:.2f} {y:.2f} {z:.2f}")

Question 12: What is the output of the following print statement concatenation?

```
print("6" + "8")
```

a) 68
b) 86
c) 14
d) 48

Lab 1 – PART 2

The following exercises require some code to be entered and executed. Make sure to save the code to a file and also record the output (e.g. take a screenshot or past the output into a word document). If a question is asked in the document, make sure to record your answer (e.g. in a word document). Your TA may review these documents when checking your work.

If you have any questions, run into any issues, or have completed the lab, raise your hand and the TA will provide assistance.

1. **Write the following code and see what is printed:**

```
answer1 = 5+3
answer2 = 7*4
print("answer1")
print(answer1)
```

Why did you get the output you did? How would you correctly output the value of answer2?

2. **Write the following code and run it to see the many ways we can format strings in Python:**

```
x=7
y=x+5

# Python has many ways to format strings, lets try some.

# The following prints using the format function
print("x is {} and y is {}".format(x, y))

# The following prints using the interpolation method
print("x is %d and y is %d" % (x, y))

# The following prints using a f-string
print(f"x is {x} and y is {y}")

# The following prints using concatenation
print("x is " + str(x) + " and y is " + str(y))

# The following prints by giving multiple arguments to print()
print("x is", x, "and y is", y)
```

Why did we need to use str() when using concatenation? How could we make the interpolation method work if y was a float?

3. Follow the instructions below to complete and run a program.

Let's say we had a shop, *MyShoppe*, and wanted to prompt for the shop name. We would use:

```
shopName = input("Please enter the shop name: ")
```

This will store the user-entered shop name into the variable shopName. Let's prompt for the quantity of 2 items in the shop, a ring and glasses.

```
ringQTY = input("Please enter ring QTY: ")
glassesQTY = input("Please enter glasses QTY: ")
```

Run the program as it is; it prompts for user input, but doesn't output anything. Let's add print statements for our current variables:

```
print(f"Shop name is {shopName}")
print(f"Ring QTY is {ringQTY}")
print(f"Glasses QTY is {glassesQTY}")
```

Test the program. We can see that the program works and prints the correct quantities that we entered.

Let's say we wanted to add up all of our inventory in our shop and print the total. Try:

```
total = ringQTY+glassesQTY
print(f"Inventory Total: {total}")
```

What do you think the inventory total will be if we have 4 rings and 5 glasses? Test it (run the program).

Did you get the correct output? Notice how the output is 45 and not 9. Why is that? Well, when you take a value from input it is a string value by default. So when we ask for the shop name, it is a string value. When we ask for the QTY, it is also a string value by default. What is happening when we add these 2 quantity string values is they are being concatenated. This means we need to convert the user entered values for QTY into integers, and we do this using *casting*.

Modify the prompts to include casting:

```
ringQTY = int(input("Please enter ring QTY: "))
glassesQTY = int(input("Please enter glasses QTY: "))
```

Now run the program -- we get the value 9! Make sure to save your code and output to show the TA you have run this program after you have completed the lab (there are two more questions on the next pages).

4. Write the following Python program:

You are tasked with writing a simple Unit Converter program that performs conversions between Celsius and Fahrenheit. The program should allow the user to:

1) Input the Celsius value to convert to Fahrenheit as a floating-point value (a number with decimal places).

2) Convert the value using the formula given below and store the result in a new variable:

$$Fahrenheit = \left(Celsius \times \frac{9}{5}\right) + 32$$

3) Display the converted value with using formatted strings (you may use any string formatting method you like but display the result with one (1) decimal places).

Example I/O *(user input shown in green)*:

For 32.5 Celsius:

```
Enter the temperature in Celsius: 32.5
The temperature in Fahrenheit is 90.5.
```

For -15.2 Celsius:

```
Enter the temperature in Celsius: -15.2
The temperature in Fahrenheit is 4.6.
```

What happens if the user inputs an invalid value like “Duck”? Why does this error occur (you don’t need to correct it, just explain why)?