Python Lab Manuel – Computer Science 10

Instructions

Only do the labs when directed in the Module course content and worksheets. Follow the instruction in the lab and make sure you download and save your code with the proper name into your Python folder. You will need to show your work to Mr. Murtha and hand it in for marks

Python Sandbox URL: https://edube.org/sandbox

Print Labs

Lab – Print1

Estimated Time: 10 minutes

Objectives

- becoming familiar with the print() function and its formatting capabilities
- experimenting with Python code.

Scenario

The print() command simply prints out a line to the screen.

Instructions:

- Use the print() function to print the line **Hello, Python!** to the screen. Use double quotes around the string
- Having done that, use the print() function again, but this time print **your first name**
- Remove the double quotes and run your code. Watch Python's reaction. What kind of error is thrown?
  Place your answer in the Python Module 2 Worksheet
- Now remove the parentheses, put back the double quotes, and run your code again. What kind of error is thrown this time?
- **Place your answer in the Python Module 2 Worksheet**
- Experiment as much as you can. Change double quotes to single quotes, use multiple print() functions on the same line, and then on different lines. See what happens.

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Print1**

Lab – Print2

Estimated Time: 10 minutes

Objectives

- becoming familiar with the print() function and its formatting capabilities;
- experimenting with Python code.

Scenario

Modify the first line of code below, using the sep and end keywords, to match the expected output. Don't change anything in the second print() invocation.

Use these two print() functions.

Code

```
print("Programming","Essentials","in")

print("Python")
```

Expected output

```
Programming***Essentials***in...Python
```

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Print2**

Lab – Print3

Estimated Time: 10 minutes

Objectives

- experimenting with existing Python code;
- discovering and fixing basic syntax errors;
- becoming familiar with the print() function and its formatting capabilities.

Scenario

Play with the code below and make some amendments. Also feel free to modify any part of the code, but there is one condition. Learn from your mistakes and draw your own conclusions.

Code

```
print("    *")
print("   * *")
print("  *   *")
print(" *     *")
print("***   ***")
print("  *   *")
print("  *   *")
print("  *****")
```

Output

```
    *
   * *
  *   *
 *     *
***   ***
  *   *
  *   *
  *****
```

Complete each of the following and save then with the names listed:

Original code save as: **YourName_Print3a**

- minimize the number of print() function invocations by inserting the \n sequence into the strings

Save as: **YourName_Print3b**

- make the arrow twice as large (but keep the proportions)

Save as: **YourName_Print3c**

- duplicate the arrow, placing both arrows side by side; note: a string may be multiplied by using the following trick: "string" * 2 will produce "stringstring"

Save as: **YourName_Print3d**

Now play a bit to make sure you fully understand the following. Do not save any of these changes.

- remove any of the quotes, and look carefully at Python's response; pay attention to where Python sees an error - is this the place where the error really exists?
- do the same with some of the parentheses;
- change any of the print words into something else, differing only in case (e.g., Print)
- replace some of the quotes with apostrophes; watch what happens carefully.

Save work for marks

When done download and save the code to your Python folder.

Literals Lab

Lab – Literal1

Estimated Time: 10 minutes

Objectives

- Becoming familiar with the print() function formatting capabilities
- Practicing coding strings
- Experimenting with Python code

Scenario

Write a **one-line** piece of code, using the print() function, as well as the newline and escape characters, to match the expected result outputted on three lines.

Expected output

```
"I'm"
""learning""
"""Python"""
```

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Literal1**

Variable Labs

Lab – Variable1

Estimated Time: 10 minutes

Objectives

- Becoming familiar with the concept of storing and working with different data types
- Experimenting with Python code.

Scenario

Here is a short story:

Once upon a time in Appleland, John had three apples, Mary had five apples, and Adam had six apples. They were all very happy and lived for a long time. End of story.

Your task is to:

- Create the variables: john, mary, and adam
- Assign values to the variables. The numbers of fruit possessed by each person
- Print the variables on one line, and separate each of them with a comma
- Now create a new variable named total_apples equal to addition of the three other variables
- Print the value stored in total_apples
- Print a string and an integer together on one line. Total number of apples: = total_apples

Expected output

```
3, 5, 6

14

Total number of apples: = 14
```

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Variable1**

Lab – Variable2

Estimated Time: 10 minutes

Objectives

- Becoming familiar with the concept of, and working with, variables
- Performing basic computations and conversions
- Experimenting with Python code

Scenario

Part 1:

Miles and kilometers are units of length or distance.

Bearing in mind that 1 mile is equal to approximately 1.61 kilometers, complete the program so that it converts:

miles to kilometers;

kilometers to miles.

Enter the following code:

```
kilometers = 12.25

miles = 7.38

miles_to_kilometers = ###  (### = miles * 1.61)

kilometers_to_miles = ###  (### = kilometers / 1.61)

print(miles, "miles is", round(miles_to_kilometers, 2), "kilometers")

print(kilometers, "kilometers is", round(kilometers_to_miles, 2), "miles")
```

Do not change anything in the existing code. Write your code in the places indicated by ###. Test your program with the data we've provided in the source code. Pay particular attention to what is going on inside the print() function. Analyze how we provide multiple arguments to the function, and how we output the expected data.

Note that some of the arguments inside the print() function are strings (e.g., "miles is", whereas some other are variables (e.g., miles).

TIP

There's one more interesting thing happening there. Can you see another function inside the print() function? It's the **round()** function. Its job is to round the outputted result to the number of decimal places specified in the parentheses, and return a float (inside the round() function). We're going to talk about functions very soon, so don't worry that everything may not be fully clear yet.

Expected output

```
7.38 miles is 11.88 kilometers

12.25 kilometers is 7.61 miles
```

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Variable2a**

Part 2:

Write a temperature converter for Celsius to Fahrenheit and Fahrenheit to Celsius. Use the round() function to round your results to three decimal places. Remember to test your programs.

Expected output

```
### Celsius is ### Fahrenheit

### Fahrenheit is ### Celsius
```

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Variable2b**

Lab – Variable3

Estimated Time: 15 minutes

Objectives

- Becoming familiar with the concept of numbers, operators, and arithmetic operations
- Performing basic calculations

Scenario

Enter the following code:

```
x =  # hardcode your test data here

x = float(x)

# write your code here

print("y =", y)
```

It reads a float value, puts it into a variable named x, and prints the value of a variable named y. Your task is to complete the code to evaluate the following expression:

3x3 - 2x2 + 3x - 1

The result should be assigned to y. Keep your code clean and readable, and test it using the data provided. Each time assigning it to the x variable (by hardcoding it).

Test Data

Sample input

x = 0

x = 1

x = -1

Expected output

```
y = -1.0

y = 3.0

y = -9.0
```

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Variable3**

Comment Labs

Lab – Comment1

Estimated Time: 5 minutes

Objectives

- Using and not using comments
- Replacing comments with code

Scenario

Comments are used not only to make programs easier to understand, but also to disable pieces of code that are currently not needed (e.g., when you need to test some parts of your code only and ignore other). Good programmers describe each important piece of code, and give self-commenting names to variables, as sometimes it is simply much better to leave information in the code.

Your task: Try to improve the code/comments below. Add or remove comments where you find it appropriate (yes, sometimes removing a comment can make the code more readable) and change variable names where you think this will improve code comprehension.

Enter your changes and code into Python and test it before saving the work. More code on next page

```
#this program computes the number of seconds in each number of hours

# this program has been written two days ago


a = 2 # number of hours

seconds = 3600 # number of seconds in 1 hour


print("Hours: ", a) #printing the number of hours

# print("Seconds in Hours: ", a * seconds) # printing number of seconds for number of hours


#here we should also print "Goodbye", but a programmer didn't have time to write any code

#this is the end of the program that computes the number of seconds in 3 hour
```

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Comment1**

input() Labs

Lab – input1

Estimated Time: 10 minutes

Objectives

- Becoming familiar with the inputting and outputting of data
- Evaluating simple expressions

Scenario

Your task is to complete the code to evaluate the results of four basic arithmetic operations. The results must be printed to the console. You may not be able to protect the code from a user who wants to divide by zero. That's okay don't worry about it for now.

Test your code - does it produce the results you expect?

# input a float value for variable a here

# input a float value for variable b here

# output the result of addition here

# output the result of subtraction here

# output the result of multiplication here

# output the result of division here

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Iput1**

Lab – input2

Estimated Time: 20 minutes

Objectives

- Becoming familiar with the concept of numbers, operators, and arithmetic operations
- Understanding the precedence and associativity operators, as well as the proper use of parentheses

Scenario

Your task is to complete the code to evaluate the following expression:

$$\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x}}}}$$

The result should be assigned to y. Be careful - watch the operators and keep their priorities in mind. Don't hesitate to use as many parentheses as you need. You can use additional variables to shorten the expression (but it's not necessary). Test your code carefully.

Use the following code as a starting point:

x = float(input("Enter value for x: "))

# Write your code here.

print("y =", y)

Test Data

Sample input: 1

Expected output: y = 0.6000000000000001

Sample input: 10

Expected output: y = 0.09901951266867294

Sample input: 100

Expected output: y = 0.009999000199950014

Sample input: -5

Expected output: y = -0.19258202567760344

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Input2**

Lab – input3

Estimated Time: 20 minutes

Objectives

- Improving the ability to use numbers, operators, and arithmetic operations
- Using the print() function's formatting capabilities

Scenario

Your task is to prepare a simple code able to evaluate the **end time** of a period, given as several minutes. The start time is given as a pair of hours (0 to 23) and minutes (0 to 59). The result must be printed to the console.

For example, if an event starts at **12:17** and lasts **59 minutes**, it will end at **13:16**.

The most important thing is that the code produces valid results for valid input data. Test your code carefully. Hint: using the % operator may be the key to success.

Use the following code as a starting point:

```
hour = int(input("Starting time (hours): "))

mins = int(input("Starting time (minutes): "))

dura = int(input("Event duration (minutes): "))

# Write your code here.
```

Test Data

Sample input:

12

17

59

Expected output: 13:16

More sample output on next page.

Sample input:

23

58

642

Expected output: 10:40

Sample input:

0

1

2939

Expected output: 1:0

Save work for marks

When done download and save the code to your Python folder. Name the file **YourName_Input3**