**Lab 1: Variables**

Learning Outcomes:

* Solve very simple arithmetic problems by using variables referring values of different type
* Be able to evaluate python expressions and statements without running the code
* Get to know that there a str type in Python and to get familiar with string concatenation
* Get familiar with using print and input built-in functions

Instructions:

* Suggest that you create a working folder **is111\lab1** in your **C** or **D** drive. Store all your solutions in this working folder.
* Challenging questions are marked with \*.

To submit:

* Please submit your working solutions via your assignment Dropbox in eLearn **within 1 week**.
* We encourage you to include a comment with your name at the beginning of each source file you are submitting.
* Zip up all your source files into a single zip file called **<your email ID>\_lab1.zip** (e.g. **ahlian.lim.2011\_lab1.zip**). You should only submit a single zip file for each lab.
* How to submit: Log into eLearn, click on the “Assignments” link at the top menu. Click on the appropriate folder to submit your lab solution. Click on “Add a File”, and select the zip file that you have created with all your solutions. Insert any comments (optional), and click on “Submit”.

1. Write a program in a file named **lab1\_1.py** that computes and prints out the area and circumference of a circle of a given radius. The radius could be of set to an int or float value. You can set the value of π to **3.14**

**Note**: Declare π using the variable name PI with all capitalized letters to indicate that the value is not to be changed in the code after the initial assignment as it is supposed to be a constant. In Python, you are expected to be a responsible adult. In other words, you should not modify a constant when you see that the variable is declared with capitalized letters with underscore to separate each word. For example, NUM\_STUDENT.

Formulas:

* Area of circle = π X radius2
* Circumference of circle = π X 2 X radius

Here is a sample output when **lab1\_1** is run (if radius is set to **4.0**):

|  |
| --- |
| D:\is111\lab1>**python lab1\_1.py**  Area of circle of radius 4.0 cm is 50.24 sq cm  Circumference of circle of radius 4.0 cm is 25.12 cm |

To submit**: lab1\_1.py**

1. Your weight is actually the amount of gravitational attraction exerted on you by the Earth. On different planets, your weighing scales will show different figures depending on the gravitational force of that planet.

Write a program in a file named **lab1\_2.py** that defines a variable to store your weight on Earth. The program then computes and outputs your weight on Mercury, Venus, Jupiter and Saturn. Use this conversion table:

|  |  |
| --- | --- |
| **Planet** | **Multiply the Earth Weight by** |
| Mercury | 0.4 |
| Venus | 0.9 |
| Jupiter | 2.5 |

Here is a sample output of the program (if your weight on Earth is set to **60**):

|  |
| --- |
| D:\is111\lab1>**python lab1\_2.py**  Your weight on Earth is 60 kg  Your weight on Mercury is 24.0 kg  Your weight on Venus is 54.0 kg  Your weight on Jupiter is 150.0 kg |

To submit: **lab1\_2.py**

1. Trace the following code and write the output.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20 | a = 5  b = 3  c = 6  d = "25"  e = 2.0  print(2 + a \* b)  print(c % a - b)  result1 = b + c / 2  result2 = int(d) - result1  print(result1, result2)  print(a // e)  result3 = d \* (c // b)  print(result3)  result4 = a // b  print("Result4 = " + str(result4)) |

To submit: **-**

1. In the code given below, complete code that circularly shifts the values of 4 variables a, b, c and d. You are NOT supposed to make use of more than one new variable to accomplish the task.   
   For example, if the variable values are initially 11, 21, 31 and 41 respectively, then the final values (of a, b, c and d) are 41, 11, 21, and 31 respectively.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10 | a = 11;  b = 21;  c = 31;  d = 41;  print('before rotation: a = ', a, 'b = ', b, 'c = ', c, 'd = ',d);  **# write your code here**  print('after rotation: a = ', a, 'b = ', b, 'c = ', c, 'd = ',d); |

Here is the output when the program placed in a file named **lab1\_4.py** is executed:

|  |
| --- |
| D:\is111\lab1>**python lab1\_4.py**  before rotation: a = 11, b = 21, c = 31, d = 41  after rotation: a = 41, b = 11, c = 21, d = 31 |

To submit: **lab1\_4.py**

1. Write a program in a file named **lab1\_5.py** that converts a temperature reading in Fahrenheit (tempInF) to Celsius (tempInC). The formula for conversion is:
2. Get a value from the user for temperature in Fahrenheit.
3. Store the user input in tempInF, and print out the corresponding temperature in Celsius.

**Hint:** The built-in function input() helps to get the user input. The return value of the function is a string (type:str). Hence, remember to convert the value to float before converting to Celsius.

Here is a sample output when **lab1\_5** is executed (with tempInF set to 100.2):

|  |
| --- |
| D:\is111\lab1>**python lab1\_5.py**  Enter a Fahrenheit temperature:**100.2**  100.2 F = 37.88888888888889 C |

To submit: **lab1\_5.py**

1. Interest on credit card outstanding amount can be quite high. Some credit card companies compute interest on an *average daily balance*.  The credit card issuer determines your average daily balance for the month by multiplying the balance you owe by the number of days you carried it, and dividing by the total number of days in the month.

Here is an algorithm for computing the average daily balance and the monthly interest charge on a credit card.

Step 1: Multiply the net balance shown on the credit card statement by the number of days in the billing cycle (i.e. number of days in the month). This is what needs to be paid.

Step 2: Multiply the net payment received by the number of days the payment was received before the statement date. The statement date is always the last day of the billing cycle. Therefore, if payment was received on day 5 of the billing cycle, this converts to 31-5, or 26 days before the statement date. This is what has been paid.

Step 3: Subtract the result of the calculation in step 2 from the result of the calculation in step 1.

Step 4: Divide the result of step 3 by the number of days in the billing cycle. This value is the average daily balance.

Step 5: Compute the interest charge for the billing period by multiplying the average daily balance by the monthly interest rate.

Here is an example. Assume the billing cycle for the month is 31 days, and the monthly interest rate is 1.33%. The credit card statement shows a previous balance of $1,100.00. A payment of $650.00 was made on day 19 of the billing cycle (i.e. 12 days before the statement date). The calculation of the interest charge goes like this:

Step 1: $1,100 x 31 = $34,100

Step 2: $650 x 12 = $7,800

Step 3: $34,100 - $7,800=$26,300

Step 4: Average daily balance: $26,300 ÷ 31 = $848.39

Step 5: Interest charge: $848.39 x 0.0133 = $11.28

Write a python program called **lab1\_5** that computes the monthly interest charged on a credit card account. Your program may have to define variables to store the following values:

* previous balance
* number of days in the billing cycle (for simplicity, you can set this value to **31** regardless of month
* payment amount
* day of the billing cycle on which payment was made
* monthly interest rate

Choose suitable variable names and assign values to them to test your program.

Here is a sample output given the data used in the example above:

|  |
| --- |
| D:\is111\lab1>**python lab1\_6.py**  Previous balance is $1100  Payment of $650.0 was made on day 19 of the billing cycle  Interest on outstanding amount is $11.283548387096774 |

To submit: **lab1\_6.py**