## Lab 1 – Expressions, statements, numbers and strings

Welcome to your first (or maybe second) Python lab!

*Rather than treating you (i.e., the learners) as passive containers into which 'knowledge' is poured, these laboratories will engage you as equal partner in the process of learning. Most of the programming problems are designed so that you can perform to discover the answer on your own. This will give you a greater sense of ownership of the knowledge learned. Hint: https://docs.python.org/3/index.html

The primary goal of the first half (Lab 1) is to let you warm up with Python programming. The problems are not intended to be algorithmically challenging, instead just ways to flex your Python muscles.

In the second half (Lab 2), we will focus more on data structures (i.e., list, tuples, and dictionary) to solve some interesting problems.

Have fun and enjoy coding!

1. Comment
   Typically, you just use the # (pound) sign to comment out everything that follows it on that line.

   Then how about when you want to comment multiple lines?
   Simply use 3 single/double quotes before and after the part you would like to comment.

   Try out:

| | Code | Output |
|---|---|---|
| Single line | `print("This is not a comment")`<br>`# print("This is a comment")` | `This is not a comment` |
| Multiple Line | `"""`<br>`print("This is a comment")`<br>`print("This is still a comment")`<br>`"""`<br>`print("This is not a comment")` | `This is not a comment` |

   Do note that `print("This is not a comment")` and `print('This is not a comment')` are essentially the same. What you use is your personal preference. However, good practice is to use single quotes for shorter strings and double quotes for more complicated text or text that contains single quotes (such as print("Don't eat that!")).

2. Variables
   Just like any other good programming languages, variable is the most important concept in programming. When it comes to variables, Python is smart in handling it. Python would interpret and declare variables for you when you set them equal to a value.

Try out:

| | Code | Output |
|---|---|---|
| numbers | `x = 5`<br>`y = 10`<br>`print(x + y)` | 15 |
| strings | `x = "5"`<br>`y = "10"`<br>`print(x + y)` | 510<br>*String concatenation |

Did you notice that when we put the numbers in quotations, Python thinks that they are strings. Isn't it awesome? But there is a flip side, suppose we started off with "2" being a string, but changed our mind and want it to be a number. Can we still add the string "2" to a random number (let's say 10)?

Try out:

| Code | Output |
|---|---|
| `x = "5"`<br>`y = 10`<br>`print(x + y)` | What output would you get? |

To fix this, we have to cast the variable into a certain type.

| Code | Output |
|---|---|
| `x = "5"`<br>`y = 10`<br>`print(int(x) + y)` | 15 |

What we have essentially done is that we **cast** the string 5 to an integer 5.
Some important variables type in Python to take note of:

- int (variable) – casts variable to integer
- float(variable) – casts variable to float
- str(variable) – casts variable to string

3. Hello World
   If you don't already write a Hello World program, this might be a good time to start. The program, when run, should get Python to print 'Hello World!' to the interactive/console. Experiment with both editor and interactive interpreter.

4. Let's go more in depth into Strings

Strings contain characters. It can contain almost anything (recall item 2 above) and the number of character is flexible.

Type

To know that a string is indeed a string, duh!

```
>>> myStr = ""
>>> print(type(myStr))
```

`type()` would return the variable type of whatever is inside the parentheses. Now try using `type()` to determine other types of variables (e.g. int and float).

String Indexes

We will go more in depth into indexes (Lab 2), but for now, let's looks at string indexes.
The index of string is actually the character. You can grab just one character or a range of character.

```
>>> a = "String"
>>> print(a[1:4])
>>> print (a[:-1])
```

Take note of the output. You will realize that the indexes of Python start at 0.
For `a[1:4], tri` is printed because it prints everything up to our range of 4, but not 4 itself.
For `a[:-1],` what Python does is it calculate the index from the end toward the front. By specifying `-1`, we tell Python that we want everything from the first character to the second to last character.

Now try out some of the string functions and methods yourself

- stringVar.count('x')
- stringVar.find('x')
- stringVar.lower()
- stringVar.upper()
- stringVar.replace('a', 'b')
- stringVar.strip()

5. Printing a pattern

Now that you've learned to print text in Python, write another program that, when run, prints out the pattern below:

```
*|*|*|*|*
*********
*|*|*|*|*
*********
*|*|*|*|*
```

*Hint: you may find the optional arguments for 'print' useful for this program (e.g. .join, .format). Read about them here.

6. Printing a pattern 2 (challenge)
   Write a program that, when run, prints out a super pattern (from Q2).

```
*|*|*|*|* H *|*|*|*|* H *|*|*|*|*
********* H ********* H *********
*|*|*|*|* H *|*|*|*|* H *|*|*|*|*
********* H ********* H *********
*|*|*|*|* H *|*|*|*|* H *|*|*|*|*
==========+==========+==========
*|*|*|*|* H *|*|*|*|* H *|*|*|*|*
********* H ********* H *********
*|*|*|*|* H *|*|*|*|* H *|*|*|*|*
********* H ********* H *********
*|*|*|*|* H *|*|*|*|* H *|*|*|*|*
```

   There are many ways to solve this problem. Do explore and discuss with your peers about your approach.

7. Operators/Order of operations
   Operators are used to vaguely describe 5 different areas: arithmetic, assignment, increment/decrement, comparison and logical.

   Try out the following arithmetic operators

```
print (5 + 9)
print (5 - 9)
print (5 * 9)
print (5 / 9)
print (5 % 9)
print (5 ** 9)
print (5 // 9)

num1 = 100
num1+=10
print(num1)
num2 = 10
num2*=20
print(num2)
```

   Understand the arithmetic operators above and transcribe the following equations into Python (without simplifying).

   - $\frac{37 \times 72}{9+3}$, example equation in Python: `(37*72)/(9+3)`
   - $\frac{-25+\left(\sqrt{7+9}\right)^2}{3 \times 3}$
   - $\sqrt[4]{(-35+273)}$
   - 345 mod 2 (modular arithmetic)

8. User input

Write a program to ask the user for his/her first and last name and print them out.
Here is an example of what the program should do: (user input in blue, output in red)

```
Enter your first name:
Jane
Enter your last name:
Doe
Hi, Jane Doe.

Enter your date of birth:
Day?
3
Month?
March
Year?
2000
Jane Doe was born on March 3, 2000
```

9. Reading and Writing to Files

All the data that we have used with variables lives in the 'memory' of a computer when our Python code is running. Unfortunately, when the program stops and power is off, the data is gone.

So, to make all these data available even when the program stops, we have to open (or create) and write the data to a text file. When needed, we can open the text file to read the content.

Writing

Create a text file: `testfile.txt`

```
>>> textfile = open('testfile.txt', 'w')  # Open the file in write mode
```

Write something to the file

```
>>> textfile.write("I am a test file.\n")
>>> textfile.write("This is a new line of text.\n")
>>> textfile.close()  # Remember to always close the file once you are done
```

What if after you've closed the file and then only realize that you need to add 1 more line of string?

Reading

Reading a text file: testfile.txt

```
>>> f = open('testfile.txt', 'r') # Open the file in read mode
>>> string = f.read(1)            # Read n numbers of characters from file
```

What happens if n is empty (i.e. `f.read()`)?
Do explore the `.readline()` method as well.

Alright, that's all for this week lab. Make sure you understand everything thoroughly. We might ask you to explain your understanding/solution during the next lab session.

**Homework**

1. Create a new text file using Python code. Save the file as `python_file.txt`.
   The Python filename should follow the following standard: `<StudentMatrixNumber>.py`.

2. Using python code, write in the text file
   i) your full name in a single line
   ii) your track (e.g., Data Science) in a new line.

   Make sure the formatting is correct.

3. Write a program that reads the text file and prints what you wrote three times.

Once you are done, submit only **ONE Python file** containing your codes.