Simon Fraser University
School of Computing Science

Lab Week 3

Instructor: Liaqat Al1

Objectives

In this Lab we shall:

1. Using Python IDLE Interpreter shell (Python 3) ...
    - play around with statements, variables, literal values and data types,
    - get familiar with the Python IDLE interpreter's error messages.
2. Using Python IDLE Program editor (Python 3):
    - create and execute Python programs.

Notes

1. If we do not have enough time to finish these lab exercises within our lab session, no problem! We can finish the lab on our own outside the lab session either

   a. by coming back to CSIL and using a CSIL workstation in a lab room where there is no scheduled lab taking place, or
   b. by using our own computer at home or on campus or elsewhere.
2. Let's make sure we show our work to one of the TA's in the lab to ensure we understand the concepts (see Objectives above) exercised in this lab.
3. In this lab, you ***do not*** have to submit anything.

Lab Prep

1. Once we have logged into a CSIL Windows workstation, let's go into your **SFU Home/CMPT120** directory and let's create a directory called **Lab2**. Let's store all the files we create in this lab into this directory.
2. Let's find ourselves a partner to do Lab 2 with.

Exercise 1 – Exploring Python using the Python IDLE Interpreter shell

Research shows that exploring new material on our own is very powerful way of learning. The following exercises give us the opportunity to do exactly that: aside from practicing what we have seen so far in lectures, we shall explore aspects of Python that we have not yet seen in our lectures, but will soon. I hope this exercise trigger your curiosity. Enjoy!

Let's start by …

- Executing the Python IDLE as shown in class i.e., using the Start button at the bottom, on the left of Window's taskbar.
- Using the Python Interpreter shell, let's do the following exercises:

1. Type `1 + 2` and then hit *Enter*. Python *evaluates* this *expression*, displays the result, and then shows the prompt **>>>** once again. Because we have taken math courses in the past, we are quite familiar with the symbol `+` which represents the *addition operator*. Here are two other operators: `*` is the *multiplication operator*, and ** is the *exponentiation operator*. Note that their symbols are different from the symbols we have used so far in math. Experiment with these 3 mathematical operators by typing each of the expressions below (and any others that come to our mind) one at a time at the Python interpreter shell prompt, then press *Enter* and observe what happens:

```
2 ** 4
2 + 3 * 4  # What happens? Is the result what you expected? Why?
2 + 3 * 4
(2 + 3) * 4
```

   What about the *subtraction operator*? Let's check it out too!

2. Let's play around with Python's two division operators by typing each of these statements, one at a time (then press *Enter*), at the Python interpreter shell prompt and observe what happens:

```
23/7
23.0/7
23.0//7
23//7
type(23/7)
type(23.0/7)
type(23.0//7)
type(23//7)
```

```
round(23.0/3)
```
Question: what does the round( ) built-in function accomplish?

3. Let's play around the modulus % operator. Type each of these statements, one at a time (then press *Enter*), at the Python IDLE Interpreter shell prompt and observe what happens:

```
23 % 3
21 % 3
3 % 23
```
Question: can we figure out what the % operator does?

4. Let's play around with variables, values, data types such as integers, float and strings, type( ) and conversion functions by typing each of these statements, one at a time (then press *Enter*), at the Python IDLE Interpreter shell prompt and observe what happens:

```
type(123)
type("Hello")
type("123")
type(str(123))
type(int("Hello"))
```
Question: Why does this cause an error?
```
type(int("123"))
pi = 3.14
type(pi)
str(pi)
type("3.14")
type("three point fourteen")
type(str(pi))
type(float("3.14"))
type(float("three point fourteen"))
```
Question: Does this statement produce an error?

5. Do Exercise 1.1 of Section 1.9 Exercises of our online textbook.

   In most of the problems of this Exercise 1.1, Python will try to evaluate the expressions we were told to type (by the problems instructions), but

it won't be able to do so because the expressions will not be syntactically correct (they will be breaking some syntax rules). For example, in the last problem (5), the operator between the operands will be missing, hence breaking the syntax rule dictating the use of operators: <operand> <operator> <operand>. So, the Python interpreter will show an error message. In many cases, the Python interpreter will indicate where the syntax error occurred, but it may not always be correct, and the message itself may not give us much information about what went wrong.

As software developers, it is very helpful for us to become familiar with the type of error messages the Python interpreter displays, what these messages mean (or try to mean) and learn how to fix the error(s) that produced these messages.

6. Type `cheese` without quotation marks. What happens? This is a runtime error; specifically, it is a *`NameError`* because the name `cheese` is not defined.

7. Let's play around with the `+` operator as we apply it to different types of values. Type each of these statements, one at a time (then press *Enter*), at the Python IDLE Interpreter shell prompt and observe what happens:

```
"Hello" + "    " + "everyone"
"Hello! " + 3
"Hello! " + "3"
1 + 1
"1 + 1"
"1" + "1"
"Hello! " + str(3)
print( "Hello! " + str(1+2+3) )
```

Question: What happens?

```
"Hello! " + str(1) + str(2) + str(3)
```

8. Let's play around with the `*` operator as we apply it to different types of values. Type each of these statements, one at a time (then press *Enter*), at the Python IDLE Interpreter shell prompt and observe what happens:

```
"Hello! "  *  3
"Hello! "*3
3 * "Hello! "
"2" * "3"     Question: What may be the problem?
int("2") * "3"
int("2") * int("3")
```

9. Do Exercise 1.2 of Section 1.9 Exercises of our online textbook.

Exercise 2 – Exploring Python programs using the Program editor

1. Create a Python program by typing the following Python code fragment in a window ("New File" option of the "File" menu) of the Python IDLE Program editor. We can also copy and paste the Python code fragment. Save it, execute it and observe (understand) the output it creates:

Python code fragment:

```
grade = 48      # Explanation: here we are assigning the value 48 to the variable named "grade".
print( grade )      # Question: what do we print here?
print( grade * 10 + 1000 )
print( grade )     # Question: why has the content of the variable " grade" remained unchanged?
```

```
print( "grade" )     # Question: what do we print here?
grade = grade * 2    # Question: why is this statement not producing an output on the monitor screen?
print( "grade is %i" %grade )   # Question: as this point in the Python code fragment, the contents of the variable "grade" changed. Why?
print( anotherGrade )     # Question: what happens here?
print( GRADE )      # Question: and here?
```

2. Create a Python program by typing the following Python code fragment in a window of the Python IDLE Program editor. We can also copy and paste the Python code fragment. Save it, execute it and observe (understand) the output it creates:

   Python code fragment:

```
age = 10
print( age + age )
age = "120"
print( age + age )
```

3. Create a Python program by typing the following Python code fragment in a window of the Python IDLE Program editor. We can also copy and paste the Python code fragment. Save it, execute it and observe (understand) the output it creates:

   Python code fragment: please, read the instructions below.

```
print( """She said "Ohlala! You get
all this Python stuff so quickly!
You'll do well in the midterm!" """ )
```

   In the Python code fragment above, we are playing around with quotes by using all kinds of quotes (triple: """, single: ' and the normal type: "). We know that we can use normal quotes when we are using a string as in `print( "grade" )`. Can we discover why we are

using the triple quotes in this Python code fragment? Also, could we use single quotes with strings?

Try the following Python code fragment. It may help us understand a little more how Python strings and Python's `print( )` function work.

```
print("She said \n")
print("Ohlala! \nYou get all this Python stuff
\nso quickly! """ )
aString = "You'll do " + "well " + "in the
midterm!"
print(aString)
print("I got %d%% on my final." %(95))
print()
print("Youpi!")
print(" ")
print("")
print("Youpi!")
```

Have fun!
