Simon Fraser University
School of Computing Science

Lab Week 5

(No submission required for this lab.)

Objective

In this week's lab, our objective is to practise:

1. Creating effective User Interaction (Interface)
2. Another way of formatting strings
3. Playing around strings and lists
4. Manipulating string and using conditional statements
5. Creating Boolean expressions
6. Exploring iterative statements

Notes

1. For this lab, you are free to work on our own or with a partner.
2. This is a long lab. If you do not have enough time to finish these lab exercises within our lab session, no problem! You 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.
3. 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.

Lab Prep

Once you log into a CSIL Windows workstation, access our **U: drive/CMPT120** and create a directory called **Lab3**. Let's store any files we create as part of this lab into this directory.

Exercise 1 – Effective User Interaction (Interface)

1. Your task is to create a Python program that asks a user for her/his last name, then for her/his first name, then for her/his address (number and street name, city and postal code – all in one line using one input statement) and finally, for her/his age, and **echoes** all this information back to the user.

   "Echoing" signifies that whatever the user enters, the program prints it onto the monitor screen right after reading it, as it was entered by the user. This "echoing" technique allows the user to verify that the data s/he has entered was read by the program correctly (as intended).

   Back to our program: Let's not forget that our goal is to create an effective user interaction by giving the user enough details when

   - we are asking her/him for information (input) and
   - when echoing her/his entered information back on the monitor screen (output).

   Before creating our program, let's have a look at our Good Programming Style (GPS) web page on our course web site for more details regarding User Interaction (or interface).
2. In this second problem, our task is to create a Python program which asks a user for the same information as in the problem above and echoes all this information back to the user as well as telling the user how old s/he will be in 8 years from now.

Exercise 2 – Another way of formatting strings – the string method format( )

So far, to format a string, we have used the *old* string format **%** operator. Let's now introducing another way of formatting strings (a *newer* way): the string method called `format( )`. Let's create a Python program using the Python Code Fragment below and see what it produces. Here **{ }** act as placeholder instead of %.

Python Code Fragment:

```
# Does it round or truncate the number?
print("{:0.3f}".format(3.1415926))
```

```
# This statement shows what to do in order to include "{" and "}" in our output
print("The {} set is often represented as
{{0}}.".format("empty"))

favourite = "{}, {}, {}".format("ice", "cream", "vanilla")
print("favourite: ", favourite)      # Hummm …

# You can specify an index of the argument – of format()!
# Index 0 refers to the first argument.
favourite = "{2} {0} {1}".format("ice", "cream", "vanilla")
print("favourite: ", favourite)      # Better!

# The symbol * unpacks (takes apart) argument sequence '+-*/%'
arithmeticOps = '{}, {}, {}, {}, {}'.format(*'+-*/%')
print("arithmetic Operators: ", arithmeticOps)

# Argument's index can be repeated
print('{0}{1}{0}{1}{0}{1}{0}'.format('X', 'O'))
print('{:>26}'.format('right aligned'))

# Use '*' as a fill character
print('{:*^26}'.format('centred'))
mark = 27
outOf = 38
print("Midterm average: {:.2%}.".format(mark/outOf))

# Here, d -> decimal
print("Here is a decimal number: {0:d}".format(21))
```

Exercise 3 – Strings and Lists

1. What is the result of the following Python Code Fragments?

Python Code Fragment 1:

```
# With a string

word = 'Greetings'
print("length of the string stored in the variable word
is: ", len(word))

# get a character at a specified position from a string
print("First letter in the string variable word is:) : ",
word[0])
# Concatenate the two strings using + operator
word2 = word + '!'
print("word2 : ", word2)
```

```
# What is this in operator?
print("'e' in word? ", 'e' in word)
print("'a' in word? ", 'a' in word)
print("word*3: ", word*3)
```

Python Code Fragment 2:

```
# With a list

aList = [1,'b',300]    # list: values in square brackets[]
print("Number of values in aList are: ", len(aList))

# Value at specified postion in the list.
print("First value in the aList is : ", aList[0])

# create a new list from exiting list and add more values
bList = aList + ['D',5.5]
print("bList now contains: ", bList)

print("300 in aList? ", 300 in aList)
# (300 in aList) is a Boolean expression

print("1 in aList? ", 1 in aList)
print("'a' in aList? ", 'a' in aList)
print("'b' in aList? ", 'b' in aList)

# You can use repeat operator * with a list as well
print("aList*3: ", aList*3)
```

Let's add the following Python statements to our program:

- a statement that would create an "index out of range" error using a string,
- a statement that would create an "index out of range" error using a list,
- a statement that would successfully create and print a slice from a string,
- a statement that would successfully create and print a slice from a list,
- a statement that would successfully step through a string, printing every 3rd character starting from the second character at index 1,
- a statement that would successfully step through a list, printing every 4th element starting from the first element at index 0.

2. What is the result of the following Python Code Fragments?

Python Code Fragment 1:

```
word1 = 'abcd'
word2 = word1.upper()
word3 = word1.replace('a', 'X')
print("3 words: ", word1, word2, word3)
print("A list from a string: ", "3 cones of Vanilla ice
cream, please!".split())
```

Python Code Fragment 2:

```
aList = [1,'b',300]
bList = aList + ['D',5.5]
print("bList : ", bList)
bList[0] = 'pos 0 was changed'
print("bList : ", bList)
bList.append(66)
print("bList : ", bList)
bList.insert(5, 'five')
print("bList : ", bList)
bList.reverse()
print("bList : ", bList)
del bList[5]
print("bList : ", bList)
bList.remove(300)
print("bList : ", bList)
```

Exercise 4 – String Manipulation and Conditional Statements

1. **String Manipulation**

Hint: All the calculations below should be doable using string and arithmetic operations, string built-in functions (such as **len( )**) and string methods (such as**upper( )**, **find( )** or **index( )**). Explore which string built-in functions or methods will be helpful for each of the following step. Links to useful resources on the Internet can be found on the **Textbooks and Links** web page of our course web site and in our lecture notes.

Problem statement: Write a Python program that asks the user for her/his first name, her/his last name, her/his age and her/his address (including number and

street name). Sounds familiar? Yes, you can recycle the code you have written previously in this lab. Then, informs the user of the following:

a) the number of characters in the first name and the last name,
b) the first index at which there is an “e” (lower case only) in the address, or inform the user that there is no lower case “e” at all in the address,
c) the number of times the first letter of her/his first name appears in the user’s address (here, we need to consider the case of this first letter, so if the first letter is ‘A’, then we search for an ‘A’, not ‘A’ and ‘a’) ,
d) the square of the age.

See the 2 **sample runs** below. These sample runs clarify how the program works. We can use the data the user entered in these sample runs (bolded and underlined), as test data and others.

Sample run 1: (what the user has typed is bolded and underlined)

```
Please, enter your first name: Mary
Please, enter your last name: VeryHappy
Please, enter your age: 21
Please, enter your address: 123 Nice Street

Dear Mary VeryHappy:

Your first name has 4 characters,
your last name has 9 characters,
the first "e" in your address is at index 7,
there are 0 letters M in your address,
and the square of your age is 441.
Bye!
```

Sample run 2: (what the user has typed is bolded and underlined)

```
Please, enter your first name: John
Please, enter your last name: VeryCalm
Please, enter your age: 22
Please, enter your address: 0123 Enjoyable Road

Dear John VeryCalm:

Your first name has 4 characters,
your last name has 8 characters,
the first "e" in your address is at index 13,
```

```
there are 0 letters J in your address,
```
<- Remember: case sensitive!!!
```
and the square of your age is 484.
Bye!
```

Exercise 5 – Boolean Expressions

1. Copy and paste each of the Python Code Fragments below in a Python program and observe the results printed on the monitor screen.
   Also, we may want to rewrite the output statements such that they abide to our Good Programming Style (GPS) guidelines (see Exercise 1 – User Interface above). ☺

Python Code Fragment 1:

```
weekDay = 6
print(weekDay == 3)
print(type(weekDay))
print(type(weekDay == 3))
print(weekDay == 6 )
print(weekDay != 2)
print(weekDay < 4)
print(weekDay >= 0)
```

Python Code Fragment 2:

```
print(0 == "0")
print(0 != "0")
print(0 == 0.0)
print(str(0) == "0")
print(type(0))
print(type("0"))
print(type(0 == 0))
```

Python Code Fragment 3:

```
# a variable may contain a Boolean value!
threeLess = 3 < 4
threeEqual = (3 == 4)
print(type(threeLess))
print(threeLess)
print(threeEqual)
```

Python Code Fragment 4:

```
x = "78"
print(x.isnumeric())
print(x.isdecimal())
print(x.isdigit())
print("78".isalpha())

x = "78.0"
print(x.isnumeric())
print(x.isdigit())
print(x.isdecimal())
print(x.isalpha())

x = 78
# What kind of error is this statement producing?
print(x.isnumeric())

# What kind of error is this statement producing?
print(x.isdigit())

# What kind of error is this statement producing?
print(x.isalpha())

# What kind of error is this statement producing?
print(78.isdigit())
```

Exercise 6 – Exploring Iterative Statements

1. Let's explore the iterative statement called the `for` loop. We shall look at the `for` loops in our next lectures.
   a) First, investigate what the built-in function `range( )` produces in Python by typing `range(10)` on the Python IDLE Interpreter shell. Then type `list(range(10))`, i.e., create a list from this range and see each of its elements.

   See what we get when we change this **end** argument `10` to another number.

   Just like when we slice a string or a list, `range(10)` uses `0` as a default **start** value: `range(0, 10)`. Notice that range does not use ":" to separate the start, end and step value. These values are arguments to the built-

in function `range(...)` so these arguments are separated by a coma. See what we get when we explicitly add a **start** value to `range(10)`.

Just like when we slice a string or a list, `range(10)` also uses `1` as a default **step** value.

b) Type or copy and paste the following Python Code Fragment 1 in a Python program. Execute it and see what happens:

Python Code Fragment 1:

```
for index in range(10):
    print("Loop iteration %d" %index)
```

c) Let's modify our Python program such that it prints our name 5 times.

d) Type or copy and paste the following Python Code Fragment 2 in a Python program. Execute it and see what happens:

Python Code Fragment 2:

```
sum = 0
for index in range(10):
    sum += index
print("sum = %d" %sum)
```

What does the above Python Code Fragment 2 do?

Have fun!
