East West University
Department of Computer Science and Engineering

CSE 303: Statistics for Data Science
LAB 02 (Handout)
Course Instructor: Dr. Mohammad Rezwanul Huq

Intermediate Python Programming

Lab Objective

Familiarize students with Python functions, lambda functions, list comprehension etc. Also introducing Pandas DataFrame basics.

Lab Outcome

After completing this lab successfully, students will be able to:

1. **Understand** the intermediate concepts of Python such as functions, lambda functions, list comprehension etc.
2. **Write** Python programs to solve generic problems with modest complexity.

Psychomotor Learning Levels

This lab involves activities that encompass the following learning levels in psychomotor domain.

Level	Category	Meaning	Keywords
P1	Imitation	Copy action of another; observe and replicate.	Relate, Repeat, Choose, Copy, Follow, Show, Identify, Isolate.
P2	Manipulation	Reproduce activity from instruction or memory	Copy, response, trace, Show, Start, Perform, Execute, Recreate.

Required Applications/Tools

- Anaconda Navigator (Anaconda3)
  - Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.
  - Popular Tools/IDEs: Spyder, Jupyter Notebook
- Google Colab: Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Lab Activities

1. Functions

Functions in python are defined using the block keyword "def", followed with the function's name as the block's name. For example:

```
def my_function():
  print("Hello from my function")
my_function()
```

Functions may also receive arguments (variables passed from the caller to the function). For example:

```
# reading input values from user
username = input('What is your name? ')
age = int(input('What is your age? '))
greeting = input('write your greetings: ')

def my_function_with_args(username, age, greeting):
        print("Hello, %s , Your age is %d, From My Function!, I wish you %s"%(username, age,
        greeting))

my_function_with_args(username, age, greeting)
```

Functions may return a value to the caller, using the keyword- 'return' . For example:

```
def sum_two_numbers(a, b):
   return a + b

print(sum_two_numbers(10, 5))
```

2. Lambda Functions

In Python, an anonymous function is a function that is defined without a name.

While normal functions are defined using the def keyword in Python, anonymous functions are defined using the lambda keyword.

Hence, anonymous functions are also called lambda functions.

Syntax of Lambda Function in python

```
lambda arguments: expression
```

Consider the following normal function.

```
def double(x):
   return x*2
print(double(5))
```

The corresponding lambda function is:

```
double = lambda x: x * 2
print(double(5))
```

3. User-defined Classes and Objects

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

A very basic class would look something like this:

```
class MyRectangle:
    length = 10.0 # any value will do
    width = 5.0

    def __init__(self, length, width):
        self.length = length
        self.width = width

    def printArea(self):
        print("Area is : " , self.length * self.width)

r1 = MyRectangle(4, 2)
r1.printArea()
```

4. List Comprehension

List Comprehensions is a very powerful tool, which creates a new list based on another list, in a single, readable line.

For example, let's say we need to create a list of integers which specify the length of each word in a certain sentence, but only if the word is not the word "the".

```
sentence = "the quick brown fox jumps over the lazy dog"
words = sentence.split()
word_lengths = []
for word in words:
if word != "the":
        word_lengths.append(len(word))
print(words)
print(word_lengths)
```

Using a list comprehension, we could simplify this process to this notation:

```
sentence = "the quick brown fox jumps over the lazy dog"
words = sentence.split()
word_lengths = [len(word) for word in words if word != "the"]
print(words)
print(word_lengths)
```

5. Sets

Sets are lists with no duplicate entries. Let's say you want to collect a list of words used in a paragraph:

```
print(set("my name is Eric and Eric is my name".split()))
```

The above statement will return the following:
{'name', 'Eric', 'is', 'my', 'and'}

Sets are a powerful tool in Python since they have the ability to calculate unions, differences and intersections between other sets.

```
a = set(["Jake", "John", "Eric"])
print(a)
b = set(["John", "Jill"])
print(b)
a = set(["Jake", "John", "Eric"])
print(a)
b = set(["John", "Jill"])
print(b)
print(a.intersection(b))
print(a.symmetric_difference(b))
print(a.difference(b))
print(b.difference(a))
print(a.union(b))
```

6. map(), filter(), reduce()

The map() function applies a given function to each item of an iterable (list, tuple etc.) and returns a list of the results.

The syntax of map() is:

```
map(function, iterable, ...)
```

A code snippet transforming each element of a list into its double and put them into another list.

```
def double(x):
    return x*2
list1 = [1, 2, 3, 4, 5, 6]
results = []
for i in list1:
    results.append(double(i))
```

The following shows the use of map() in the above case:

```
def double(x):
    return x*2
list1 = [1, 2, 3, 4, 5, 6]
results = [x for x in map(double, list1)] #lambda functions can also be used
print(results)
```

The filter() method constructs an iterator from elements of an iterable for which a function returns true.

The syntax of filter() method is:

```
filter(function, iterable)
```

```
def filterVowels(letter):
    vowels = ['a', 'e', 'i', 'o', 'u']

    if(letter in vowels):
        return True
    else:
        return False

filteredVowels = filter(filterVowels, letters)

print('The filtered vowels are:')
for vowel in filteredVowels:
    print(vowel)
```

The reduce(fun,seq) function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along. This function is defined in “functools” module.

The syntax of reduce() method is:

```
reduce(function, iterable)
```

```
def add(a, b):
   return a+b

# importing functools for reduce()
import functools

# initializing list
list1 = [1, 3, 5, 6, 2]

# using reduce to compute sum of list
print ("The sum of the list elements is : ", end="")
print (functools.reduce(add,list1))
```

7. Python String Methods

- Python String capitalize(): Converts first character to Capital Letter
- Python String casefold(): converts to case folded strings
- Python String center(): Pads string with specified character
- Python String count(): returns occurrences of substring in string
- Python String encode(): returns encoded string of given string
- Python String endswith(): Checks if String Ends with the Specified Suffix
- Python String expandtabs(): Replaces Tab character With Spaces
- Python String find(): Returns the index of first occurrence of substring
- Python String format(): formats string into nicer output
- Python String format_map(): Formats the String Using Dictionary
- Python String index(): Returns Index of Substring
- Python String isalnum(): Checks Alphanumeric Character
- Python String isalpha(): Checks if All Characters are Alphabets
- Python String isdecimal(): Checks Decimal Characters
- Python String isdigit(): Checks Digit Characters
- Python String isidentifier(): Checks for Valid Identifier
- Python String islower(): Checks if all Alphabets in a String are Lowercase
- Python String isnumeric(): Checks Numeric Characters
- Python String isprintable(): Checks Printable Character
- Python String isspace(): Checks Whitespace Characters
- Python String istitle(): Checks for Titlecased String
- Python String isupper(): returns if all characters are uppercase characters
- Python String join(): Returns a Concatenated String
- Python String ljust(): returns left-justified string of given width
- Python String lower(): returns lowercased string
- Python String lstrip(): Removes Leading Characters
- Python String maketrans(): returns a translation table
- Python String partition(): Returns a Tuple
- Python String replace(): Replaces Substring Inside
- Python String rfind(): Returns the Highest Index of Substring

- Python String rindex(): Returns Highest Index of Substring
- Python String rjust(): returns right-justified string of given width
- Python String rpartition(): Returns a Tuple
- Python String rsplit(): Splits String From Right
- Python String rstrip(): Removes Trailing Characters
- Python String split(): Splits String from Left
- Python String splitlines(): Splits String at Line Boundaries
- Python String startswith(): Checks if String Starts with the Specified String
- Python String strip(): Removes Both Leading and Trailing Characters
- Python String swapcase(): swap uppercase characters to lowercase; vice versa
- Python String title(): Returns a Title Cased String
- Python String translate(): returns mapped charactered string
- Python String upper(): returns uppercased string
- Python String zfill(): Returns a Copy of The String Padded With Zeros

Useful Links:

- https://www.learnpython.org/
- https://realpython.com/
- https://www.programiz.com/python-programming/anonymous-function
- https://www.programiz.com/python-programming/methods/built-in/map
- https://www.programiz.com/python-programming/methods/built-in/filter
- https://www.geeksforgeeks.org/reduce-in-python/
- https://www.programiz.com/python-programming/methods/string

East West University
Department of Computer Science and Engineering

CSE 303: Statistics for Data Science
LAB 02 (Exercise)
Course Instructor: Dr. Mohammad Rezwanul Huq

List Comprehension Problems

```
# Use for the questions below:
nums = [i for i in range(1,1001)]
string = "Practice Problems to Drill List
Comprehension in Your Head."
```

1. Find all of the numbers from 1–1000 that are divisible by 8
2. Find all of the numbers from 1–1000 that have a 6 in them
3. Count the number of spaces in a string (use string above)
4. Remove all of the vowels in a string (use string above)
5. Find all of the words in a string that are less than 5 letters (use string above)
6. Use a dictionary comprehension to count the length of each word in a sentence (use string above)
7. Use a nested list comprehension to find all of the numbers from 1–1000 that are divisible by any single digit besides 1 (2–9)
8. For all the numbers 1–1000, use a nested list/dictionary comprehension to find the highest single digit any of the numbers is divisible by

Submission Instruction:

Create a zip file containing your python (.py) files along with the report. Name of the file should be: <your-student-id>_Lab02.zip

Submit in the link given in the classroom.