Course Title (Course Code): Statistics for Data Science (CSE303)
Lab 01 Exercises
Course Instructor: Md Al-Imran
Lab Title: Introduction to Python Programming

Lab Objective

Familiarize students with the fundamental concepts of Python Programming such as data structures, control flow statements, functions, lambda functions, and object-oriented programming.

Lab Outcome

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

1. Understand the fundamental concepts of Python.
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: IDLE, 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

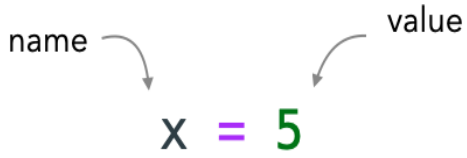
1. Installation and writing basic codes

Python is an interpreted high level general-purpose programming language. The latest version of Python 3.x can be found on https://www.python.org/. The language can be used to build anything you want! It is free and open source. So, let's start Python with printing and reading inputs.

```
print ("This line will be printed.")
using Format specifier:
name = "John"
age = 23
print ("%s is %d years old." % (name, age))
Reading Inputs
name = input("Enter your name: ")
```

```
age = int(input("Enter your age: "))
print(f'{name} is {age} years old')
```

Variables: Variables are containers for holding data and they're defined by a name and value.




```
# Integer variable
x = 5
print (x)
print (type(x))
```

```
# String variable
x = "hello"
print (x)
print (type(x))
```

```
# Floating point numbers
import math
print(format(math.pi, '.12g'))
print(format(math.pi, '0.6f'))
```

2. Operators

Arithmetic	Comparison	Logical
• Same set: **+, -, *, /, %** • // Floor division - division that results into whole number adjusted to the left in the number line x // y • ** Exponent - left operand raised to the power of right x**y (x to the power y)	Same set: > < = <= >= == ! !=	and or not

2**52 <= 2**56 // 10 < 2**53 (Guess Output?)

3. Conditional Statements

```
if case1:
        perform action
elif case2:
        perform action2
else:
        perform action 3
```

```
# If statement
x = 4
if x < 1:
    score = "low"
elif x <= 4: # elif = else if
    score = "medium"
else:
    score = "high"
print (score)
```

```
# If statement with a boolean
x = True
if x:
    print ("it worked")
```

4. Loops

```
# For loop
veggies = ["carrots", "broccoli", "beans"]
for veggie in veggies:
      print (veggie)
```

```
if veggie == "broccoli":
      break

print(veggie)

Try continue as well
```

Course Title (Course Code): Statistics for Data Science (CSE303)
Lab 01 Exercises
Course Instructor: Md Al-Imran
Lab Title: Introduction to Python Programming

5. Data Structures

```
# Creating a list
x = [3, "hello", 1.2
]
print (x)
```

```
# Creating a tuple
x = (3.0, "hello")
# tuples start and end with (
)
print (x)
```

```
# Sets
text = "SDS IN PYTHON"
print (set(text))
print (set(text.split(" "))
)
```

List Comprehension

```
# Grab every letter in string
lst = [x for x in 'word']
lst

# Check for even numbers in a range
lst = [x for x in range(11) if x % 2 == 0]
lst

# Convert Celsius to Fahrenheit
celsius = [0,10,20.1,34.5]

fahrenheit = [ ((float(9)/5)*temp + 32) for temp in Celsius ]

fahrenheit

#nested list comprehension
lst = [ x**2 for x in [x**2 for x in range(11)]]
lst
```

List Methods

- append(): Add a single element to the end of the list
- clear(): Removes all Items from the List
- copy(): returns a shallow copy of the list
- count(): returns count of the element in the lis
- extend(): adds iterable elements to the end of the list
- index(): returns the index of the element in the list
- insert(): insert an element to the list
- pop(): Removes element at the given index
- remove(): Removes item from the list
- reverse(): reverses the list
- sort(): sorts elements of a list

Dictionaries

phonebook = { "John" : 938477566, "Jack" : 938377264, "Jill" : 947662781 }
print(phonebook)

```
for name, number in phonebook.items():
        print("Phone number of %s is %d" % (name, number))
```

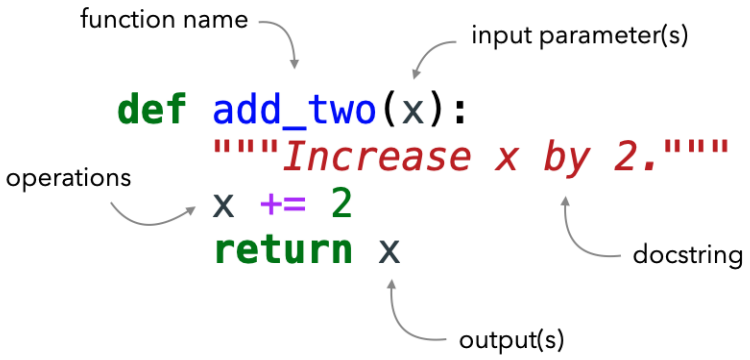
Dictionary methods

- clear(): Removes all Items
- copy(): Returns Shallow Copy of a Dictionary
- fromkeys(): creates dictionary from given sequence
- get(): Returns Value of The Key
- items(): returns view of dictionary's (key, value) pair
- keys(): Returns View Object of All Keys
- pop(): removes and returns element having given key
- popitem(): Returns & Removes Latest Element From Dictionary
- setdefault(): Inserts Key With a Value if Key is not Present
- update(): Updates the Dictionary
- values(): returns view of all values in dictionary

The differences and similarities we have seen so far:

	Mutable	Ordered	Indexable	Unique
List	✓	✓	✓	✗
Tuple	✗	✓	✓	✗
Set	✓	✗	✗	✓
Dictionary	✓	✗	✗	✓ keys ✗ values

6. Functions




```
# Define the function
def add_two(x):
    """Increase x by 2.""" # explains what this function will do
    x += 2
    return x
```

Notes: It's good practice to always use keyword argument when using a function so that it is very clear what input variable belongs to what function input parameter. On a related note, you will often see the terms *args and **kwargs which stand for arguments and keyword arguments. You can extract them when they are passed into a function. The significance of the * is that any number of arguments and keyword arguments can be passed into the function.

```
def f(*args, **kwargs):
    x = args[0]
    y = kwargs.get("y")
    print (f"x: {x}, y: {y}")
f(5, y=2)
```

Lambda

```
def identity(x):
    return x
```

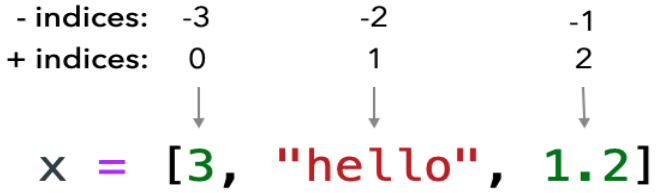
Can be rewrite as

```
lambda x: x          #Keyword: lambda, bound variable: x, body: x
```

Another example

```
(lambda x: x + 1)(2)
```

7. Indexing and Slicing




```
# Indexing
x = [3, "hello", 1.2]
print ("x[0]: ", x[0])
print ("x[1]: ", x[1])
print ("x[-1]: ", x[-1]) # the last item
print ("x[-2]: ", x[-2]) # the second to last item
```

```
# Indexing beyond length
print (x[:100])
print (len(x[:100]))
```

```
# Slicing
print ("x[:]: ", x[:]) # all indices
print ("x[1:]: ", x[1:]) # index 1 to the end of the list
print ("x[1:2]: ", x[1:2]) # index 1 to index 2 (not including index 2)
print ("x[:-1]: ", x[:-1]) # index 0 to last index (not including last index)
```

Course Title (Course Code): Statistics for Data Science (CSE303)
Lab 01 Exercises
Course Instructor: Md Al-Imran
Lab Title: Introduction to Python Programming

---

8. **Object Oriented Programming**

   Classes

   - _init_ function
   - _str_ function
   - Object methods
   - Inheritance

   Follow Lab instructions properly to learn the Python OOP.

Useful Links:

- Book: Practical Statistics for Data Science by O'Reilly Publications
- https://www.learnpython.org/
- https://realpython.com/

Note: You are given a set of problems in different pdf file. And you are requested to commit your solution to your Github repository. Please take a serious note that your profile will be visited per week.