East West University
Dept. of CSE
CSE 303: Statistics for Data Science
LAB 01 (Handout)

Introduction to Python Programming

Lab Objective

Familiarize students with the fundamental concepts of Python Programming such as data types, control flow statements, functions, lambda functions and list comprehension.

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: 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. Introducing Python

- General purpose programming language, you can build anything!
- Open Source. Free to use.
- Lots of Python packages
- Current Version: Python 3.x
- **Indentation is important using tabs or spaces!**

2. Printing Statements 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');
```

3. Objects (Variables) and Types

- Python is purely object oriented. Not “statically typed”.
- Integer, Floating points, Strings – int, float, str, bool and so on.
- Strings are defined either with a single quote of a double quotes.
- Useful functions: id(object_name), type(object_name), isinstance(object_name, type_name)

4. Arithmetic Operators

- 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)

5. Comparison Operators

· Same set: >, <, >=, <=, ==, !=

6. Logical Operators

· **and, or, not**

7. Conditional Statements

```
if condition1:
        action1
elif condition2:
        action2
else:
        action3
```

Special Operators: is, is not, in, not in

```
name = "John"
if name in ["John", "Rick"]:
        print("Your name is either John or Rick.")
```

8. Loops

· There are two types of loops in Python, for and while.

```
primes = [2, 3, 5, 7] # A list
for prime in primes:
   print(prime)

for x in range(5):
   print(x)

for i in range(1, 100, 2):
   print(i)
```

```
count = 0 while
count < 5:
print(count)
   count += 1  # This is the same as count = count + 1

count=0
while(count<5):
print(count)
count +=1 else:
   print("count value reached %d" %(count))
```

9. Python Functions

Defining a function first

```
def my_function(name, age):
      print(f'{name} is {age} years old')
```

Invoking a function

```
my_function('John', 23)
```

10. Mutable Objects: Lists

- Lists are very similar to arrays. They can contain any type of variable, and they can contain as many variables as you wish.
- Elements can be accessed using indexing: mylist = [1, 2, 3]; print (mylist[0]); print(mylist[-1]); print(mylist[1:3]);
- Python List append(): Add a single element to the end of the list
- Python List clear(): Removes all Items from the List
- Python List copy(): returns a shallow copy of the list
- Python List count(): returns count of the element in the list
- Python List extend(): adds iterable elements to the end of the list
- Python List index(): returns the index of the element in the list
- Python List insert(): insert an element to the list
- Python List pop(): Removes element at the given index
- Python List remove(): Removes item from the list
- Python List reverse(): reverses the list
- Python List sort(): sorts elements of a list

11. Immutable Objects: Tuple

- A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.
- A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).

```
# Tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)
```

- Python Tuple count(): returns count of the element in the list
- Python Tuple index(): returns the index of the element in the list

12. Mutuable Objects: Dictionaries

- A dictionary is a data type similar to arrays, but works with keys and values instead of indexes. Each value stored in a dictionary can be accessed using a key, which is any type of object (a string, a number, a list, etc.) instead of using its index to address it.

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

Looping over Dictionaries: phonebook = {"John" : 938477566,"Jack" : 938377264,"Jill" : 947662781} for name, number in phonebook.items():
   print("Phone number of %s is %d" % (name, number))

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

Useful Links:

- https://www.learnpython.org/
- https://www.programiz.com/python-programming/operators
- https://www.programiz.com/python-programming/methods/list
- https://www.programiz.com/python-programming/methods/dictionary
- https://www.programiz.com/python-programming/tuple

CSE 303: Statistics for Data Science
LAB 01 (Exercise)

---

1. Given two integer numbers, write a Python program to return their product. If the product is greater than 1000, then return their sum. Read inputs from the user.
2. Write a Python program to find the area and perimeter of a circle. Read inputs from the user.
3. Write a Python program to calculate the compound interest based on the given formula. Read inputs from the user.
   $A = P * (1 + R/100)^T$ where P is the principle amount, R is the interest rate and T is time (in years).
   Define a function named as compound_interest_<your-student-id> in your program.
4. Given a positive integer N (read from the user), write a Python program to calculate the value of the following series. $1^2 + 2^2 + 3^2 + 4^2 \ldots.. + N^2$
5. Given a positive integer N (read from the user), write a Python program to check if the number is prime or not. Define a function named as prime_find_<your-student-id> in your program.
6. Given a positive integer n (read from the user), write a Python program to find the n-th Fibonacci number based on the following assumptions. $F_n = F_{n-1} + F_{n-2}$ where $F_0 = 0$ and $F_1 = 1$
7. Given a list of numbers (hardcoded in the program), write a Python program to calculate the sum of the list. Do not use any built-in function.
8. Given a list of numbers (hardcoded in the program), write a Python program to calculate the sum of the even-indexed elements in the list.
9. Given a list of numbers (hardcoded in the program), write a Python program to find the largest and smallest element of the list. Define two functions largest_number_<your-student-id> and smallest_number_<your-student-id> in your program. Do not use any built-in function.
10. Given a list of numbers (hardcoded in the program), write a Python program to find the second largest element of the list.
11. Given a string, display only those characters which are present at an even index number. Read inputs from the user.
12. Given a string and an integer number n, remove characters from a string starting from zero up to n and return a new string. N must be less than the length of the string. Read inputs from the user. Do not use any built-in function.
13. Given a string, find the count of the substring “CSE303” appeared in the given string. Do not use any built-in function.
14. Given a string, write a python program to check if it is palindrome or not. Define a function named palindrome_checker_<your-student-id> in your program.
15. Given a two list of numbers (hardcoded in the program), create a new list such that new list should contain only odd numbers from the first list and even numbers from the second list.

Submission Instruction:

Create a zip file containing your python (.py) files along with the report. Name of the file should be:

<your-student-id>_Lab01.zip