Lab-03
To Get familiar with Advance concepts in Python

Objectives:

- Python Lambda , python Arrays
- File I/O In Python(Python File Handling,Read Files, Write/Create Files,Delete File
- Python Modules: Built-in Modules In Python,OS,Math,Random,Datetime

Lab tasks

Exercise 1

Perform the given operations

I. a Python program to square and cube every number in a given list of integers using Lambda.

```
nums = [1, 2, 3, 4]
square_nums = list(map(lambda x: x**2, nums))
cube_nums = list(map(lambda x: x**3, nums))
print(square_nums)
print(cube_nums)
```

II. a Python program to find if a given string starts with a given character using Lambda.

```
check_char = lambda full_word,char: True if full_word.startswith(char) else False;
print(check_char("Qambar","Q"))
```

III. a Python program to extract year, month, date and time using Lambda.

Code:
```
now = datetime.datetime.now()
year = lambda x: x.year
month = lambda x: x.month
day = lambda x: x.day
time = lambda x: x.time
print(year(now))
print(month(now))
print(day(now))
print(time(now))
```

Exercise 2

I. You have collected information about cities in your province. You decide to store each city's name, population, and mayor in a file. Write a python program to accept the data for a number of
cities from the keyboard and store the data in a file in the order in which they're entered.

Code:
```
cities_data =  open('cities.txt','x')
cities_data =  open('cities.txt','a')
cities_data.write("City Population Mayor")
def addCity():
    cityName = input("Enter City Name: ")
    population = input("Enter Population: ")
    mayorName = input("Enter Mayor Name: ")
    cities_data.write(f'\n{cityName} {population} {mayorName}')
addCity()
```

II. Write a python program to create a data file student.txt and append the message "Now we are AI students"s

Code:
```
student_data =  open('student.txt','x')
student_data =  open('student.txt','a')
student_data.write("Now we are AI students")
```

Exercise 3:

Plan on python modules.
(hands on experience)

Notes: these exercises will focus on importing, exploring and using some of the more useful modules available in the Python standard library. Unlike previous exercises there are no questions, the goal is to inform you of modules in the Python standard library that you might find useful in your final projects.

1. In previous exercise sets we used specific modules, for example, random to generate random numbers, math to access mathematical functions and sys for system variables. To access the contents of a

module we use the import keyword. For example:

>>> import math

Modules in the standard library have a top-level help page to give you an overview of the contents of the module (remember, press 'q' to exit the help pages):

>>> help(math)

Or you can access the help page of a specific variable or function:

>>> help(math.sqrt)

In Python3 you can also see what is in a module by typing the name of the module, a dot and pressing tab twice.

>>> math.

2. Instead of importing an entire module, we can import a single function or variable using the from keyword, for example:

>>> from math import sqrt

>>>

sqrt(9) 3.0

This can be useful if we know we are going to use the function alot and want our code to look neater. The downside is that we cannot have a function called, for example, sqrt() in our own code.

3. Sometimes modules change names from one version to another or have long names that we want to avoid typing all the time. Instead of using from we can rename the module using as.

For example, the English do not use the word math, but use the far more pleasing, maths. If I wanted to change that in my code I could do this:

>>> import math as maths

>>> maths.sqrt(9)
3.0
This only works with modules however, not the functions inside a module (because they are not modules, they are functions):
>>> import math.sqrt as squareroot Traceback (most recent call last):
File "<frozen importlib._bootstrap>", line 2218, in _find_and_load_unlocked
AttributeError: 'module' object has no attribute '_path_' During handling of the above exception, another exception occurred:
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ImportError: No module named 'math.sqrt'; 'math' is not a package
Instead we use a combination of the from keyword and the as keyword to rename functions or variables:
>>> from math import sqrt as squareroot >>> squareroot(9)
3.0

4. time module: if you want to time something, get the current date or pause the execution of your script, then you can use the time module.
   time.time() returns the current time in seconds since the epoch (the epoch is defined as Thursday 1st January 1970 https://en.wikipedia.org/wiki/Unix_time):
   >>> import time
   >>> time.time()
   1427791906.13485
   time.ctime(s) converts a time, s, measured in seconds past the epoch into a human-readable string. If you do not provide an argument, then it returns the current date and time:
   >>> current_time = time.time()
   >>>
   time.ctime(current_time)
   'Tue Mar 31 11:55:39 2015'
   >>> time.ctime()
   'Tue Mar 31 11:55:51 2015'
   time.sleep(s) waits for at least s seconds and then returns:
   >>> time.sleep(3)
   >>> time.sleep(0.5)
5. glob module: despite the strange name, the glob module is immensely useful for processing large numbers of files. It takes a single string argument that you use to specify a pattern to match files. Patterns are strings that map to file names. For example, in a useless example we can find all files called "file.py" in the current directory:
   >>> import glob
   >>> glob.glob("file.py")
   We never want to get a list of all files with a specific file name (such a list would be zero or one items long, so we may as well just use an if statement). To match all files in the current directory, we can use the asterisks character to mean "match any thing".
   >>> glob.glob("*")

To match a subset of files we can use the asterisks to match parts of file names. For example, to match all .py files:

```
>>> glob.glob("*.py")
```

Play around with this to ensure you understand how the asterisks works.

6. random module: in a previous set of exercises we used random.randint(a,b) to generate random integers between a and b, inclusive:

```
>>> import random
>>> random.randint(1, 10)
```

random.random() generates a real-valued random number in the interval [0,1):

```
>>> random.random()
```

random.shuffle(list) permutes the list into a random ordering in place, i.e. it does not return a permuted copy of the original list:

```
>>> x = [1,2,3,4,5]
>>> random.shuffle(x)
>>> print(x)
```

random.sample(list,k) randomly selects k elements from list (k cannot exceed len(list)) and returns a new list containing those elements:

```
>>> x = [1,2,3,4,5]
>>> random.sample(x, 2)
[5, 1]
```