LAB # 13

MODULES AND PACKAGES

OBJECTIVE

Getting familiar with the envoirnment for using modules and packages.

THEORY

Modules

- ✓ A module allows you to logically organize your Python code.
- ✓ A module can define functions, classes and variables. Modules help organize code into separate files so that programs become easier to maintain and reuse.
- ✓ Instead of writing everything in one place, related functionality can be grouped into its own module and imported whenever needed.
- ✓ There are three main types of modules in Python**: built-in modules** (provided by Python), external (third-party) modules, and **user-defined modules** (created by programmers).

1. User-Defined Modules

Create a Module
To create a module just save the code you want in a file with the file extension **.py:**

Example

Step # 01 Save this code in a file named **math_utils*.py***

```
# A utility module for basic mathematical operations
def calculate_percentage(obtained, total):
    return (obtained / total) * 100
```

Step # 02 Use the Module in Another File

```
import math_utils

marks = 420
total_marks = 500

percent = math_utils.calculate_percentage(marks, total_marks)

print("Percentage:", percent)
```

LAB # 13

Output

```
>>> %Run Lab_13.py
Percentage: 84.0
>>>
```

Types: Importing a Module

1. import module1[, module2[,... moduleN]
2. import module as m
3. from module import functionName
4. from module import *

Now use different ways to call the module we just created, by using the ***import*** statement.

Example

Step # 01 Creating a module in seprate file **: math_utils.py**

```
# math_utils.py
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
```

Step # 02 Let's import a module in separate file for results. **1st Type**

```
import math_utils  # 1st Type

print("Addition: ",   math_utils.add(5, 3))
print("Subtraction: ",math_utils.subtract(10, 4))
```

Output

```
>>> %Run Lab_13.py
Addition:  8
Subtraction:  6
>>>
```

Step # 02 Let's import a module in separate file for results. **2nd Type**

```
import math_utils as mu     # 2nd Type

print(mu.add(7, 2))
print(mu.subtract(15, 5))
```

Output

```
>>> %Run Lab_13.py
9
```

```
10
>>>
```

Step # 02 Let's import a module in separate file for results. **3rd Type**

```
from math_utils import add        # 3rd Type

print(add(6, 4))

Output

>>> %Run Lab_13.py
10
>>>
```

Step # 02 Let's import a module in separate file for results. **4rth Type**

```
from math_utils import *       # 4rth Type

print(add(9, 1))
print(subtract(20, 8))

Output

>>> %Run Lab_13.py
10
12
>>>
```

2. Built-in Modules

There are several built-in modules in Python, which you can import, here is the example of math and sys module; **Python - math Module**.

Example:

```
from math import pi

r=int(input("Enter:"))
print("Area:" , pi*(r**2))
```

Output:

```
>>> %Run task1.py
enter:3
Area: 28.27
```

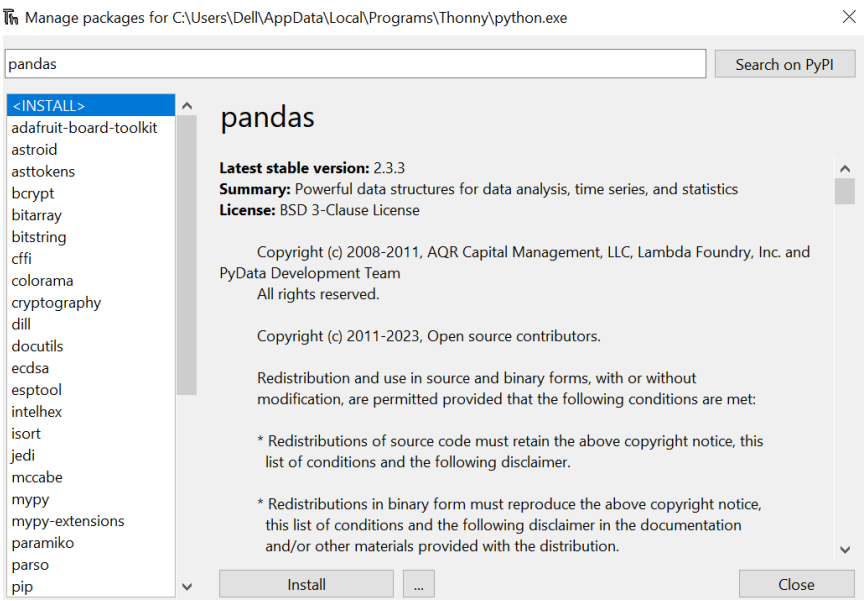
LAB # 13

Functions in Python Math Module

Functions	Description
ceil(x)	Returns the smallest integer greater than or equal to x.
factorial(x)	Returns the factorial of x
floor(x)	Returns the largest integer less than or equal to x
exp(x)	Returns e**x
cosh(x)	Returns the hyperbolic cosine of x
sinh(x)	Returns the hyperbolic cosine of x
tanh(x)	Returns the hyperbolic tangent of x
pow(x, y)	Returns x raised to the power y
sqrt(x)	Returns the square root of x
Pi	Mathematical constant, the ratio of circumference of a circle to it's diameter (3.14159...)
E	mathematical constant e (2.71828...)

3. External (third-party) Modules

External or third-party modules are libraries developed by the Python community or organizations and are not included by default with Python. These modules extend Python's functionality and are commonly used for data analysis, machine learning, web development, networking, and cybersecurity. They must be installed separately using a package manager such **as pip or install from manage packages - Thony**



LAB # 13

Once install then check:

Example:

```
import pandas
print(pandas.__version__)

Output
>>> %Run Lab_13.py
2.3.3
>>>
```

Python - sys Module

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment

Example:

```
import sys
print(sys.version)  #version number of the current Python interpreter
```

Output:

```
>>> %Run Lab_13.py
3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929 64 bit (AMD64)]
>>>
```

Difference Between Module and Function

Aspect	Function	Module
Definition	A function is a block of code that performs a specific task	A module is a file that contains related functions, classes, and variables
Scope	Works at a small level (single task)	Works at a larger level (group of related tasks)
Location	Defined inside a program or module	Saved as a separate .py file
Reusability	Reused by calling the function	Reused by importing the module
Example	def add(a, b): return a+b	math, os, or mymodule.py
Purpose	Code reuse and task abstraction	Code organization and maintainability

Packages

- ✓ A **package** is a collection of related modules organized inside a folder.
- ✓ A package is a collection of python modules, i.e., a package is a directory of python modules containing an additional __init__.py file.
- ✓ Each package in Python is a directory which must contain a special file called __init__.py.

LAB # 13

- ✓ This file can be empty, and it indicates that the directory it contains is a Python package, so it can be imported the same way a module can be imported.
- ✓ There are 130k + packages and still growing , numpy is one of the most running and useful python's package.
- ✓ It helps manage large projects by grouping multiple modules together.

Example Structure

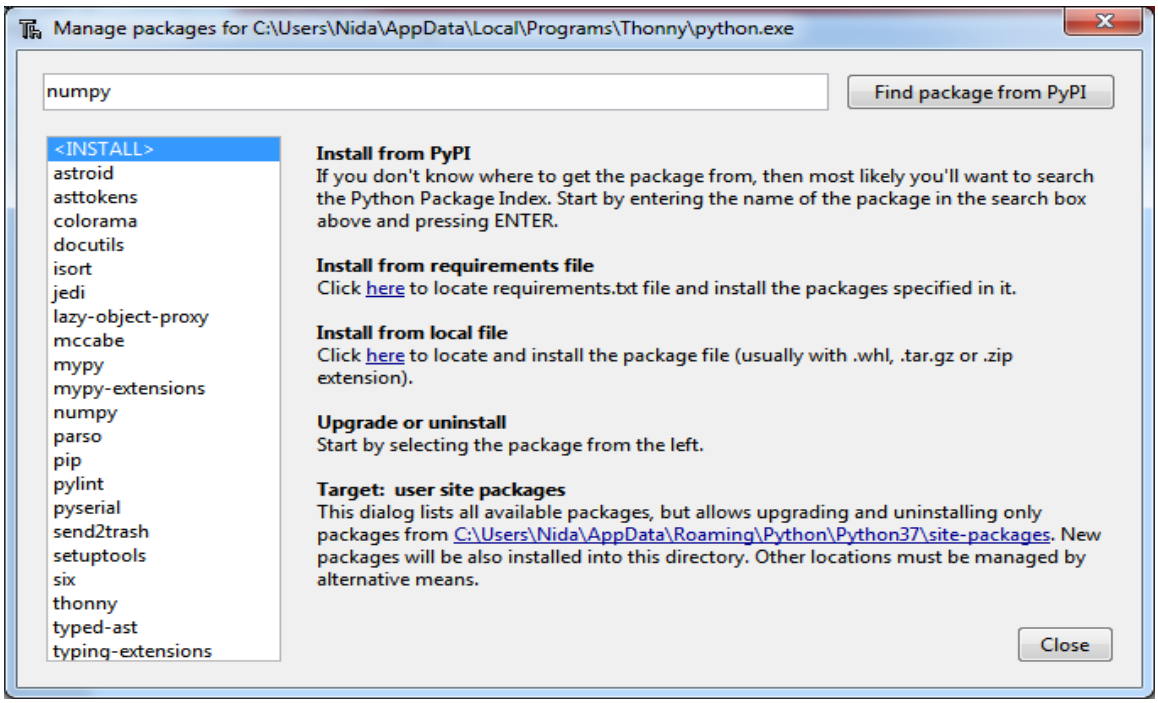
```
mypackage/
|— __init__.py
|— math_utils.py
|— string_utils.py
```

NumPy

- ✓ NumPy is a purposely an array-processing package.
- ✓ It provides a high-performance multidimensional array object, and tools for working with these arrays.
- ✓ It is the fundamental package for scientific computing with python.

Installation:

In Thonny, there is a menu and select ***Tools*** option and then select ***Manage packages,*** search numpy and install it.



LAB # 13

Example: Create 1D,2D,3D array

```
import numpy as np
#1D array: 1-dimensional array containing the values 2, 3, 4, and 5.
print("1D:",np.arange(2,6).reshape(4))
#2D array: 2-dimensional array with 2 rows and 4 columns.
print("2D:",np.arange(2,10).reshape(2,4))
#3D array: 3-dimensional array with 4 blocks, each containing 3 rows and 2 columns.
print("3D:",np.arange(24).reshape(4,3,2))
```

Output

```
>>> %Run Lab_13.py
1D: [2 3 4 5]

2D: [[2 3 4 5]
     [6 7 8 9]]

3D: [[[ 0  1]
      [ 2  3]
      [ 4  5]]

     [[ 6  7]
      [ 8  9]
      [10 11]]

     [[12 13]
      [14 15]
      [16 17]]

      [[18 19]
       [20 21]
       [22 23]]]
>>>
```

LAB # 13

EXERCISE

A. Point out the errors, if any, in the following Python programs.

1. Code:

```
import sys as s
print(sys.executable)
print(sys.getwindowsversion())
```

Output:

2. Code:

```
import datetime
from datetime import date
import times
# Returns the number of seconds
print(time.time())
# Converts a number of seconds to a date object
print(datetime.datetime.now())
```

Output:

3. Code:

```
From math import math
# using square root(sqrt) function contained
print(Math.sqrt(25) )
print(Math.pi)
# 2 radians = 114.59 degreees
print(Math.degrees(2))
```

Output:

LAB # 13

B. What would be the output of the following programs:

1. Code:

```
import calendar
yy = 2017
mm = 11
# display the calendar
print(calendar.month(yy, mm))
```

Output:

2. Code:

```
import sys
print(sys.argv)
for i in range(len(sys.argv)):
   if i==0:
      print("The function is",sys.argv[0])
   else:   print("Argument:",sys.argv[i])
```

Output:

3. Code:

```
import numpy as np
# Creating array object
arr = np.array( [[ 1, 2, 3],
                 [ 4, 2, 5]] )

# Printing array dimensions (axes)
print("No. of dimensions: ", arr.ndim)

# Printing shape of array
print("Shape of array: ", arr.shape)

# Printing size (total number of elements) of array
print("Size of array: ", arr.size)
```

Output:

LAB # 13

C. Write Python programs for the following:

1. Write a NumPy program to create an 1D array of 10 zeros, 10 ones, 10 fives

2. Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10.