**Lab Report No: 05**

**Lab Report Name: Basic Python programming**

**Name: S.M.SANOWER HOSSAIN**

**ID: IT-16047**

**Python Programming:** There are several different programming paradigms that a programming language may allow developers to code in, these are:

• Procedural Programming in which a program is represented as a sequence of instructions that tell the computer what it should do explicitly. Procedures and/ or functions are used to provide structure to the program; with control structures such as if statements and loop constructs to manage which steps are executed and how many times. Languages typifying this approach include C and Pascal.

• Declarative Programming languages, such as Prolog, that allow developers to describe how a problem should be solved, with the language/environment determining how the solution should be implemented. SQL (a database query language) is one of the most common declarative languages that you are likely to encounter.

• Object Oriented Programming approaches that represent a system in terms of the objects that form that system. Each object can hold its own data (also known as state) as well as define behaviour that defines what the object can do. A computer program is formed from a set of these objects co-operating together. Languages such as Java and C# typify the object oriented approach.

• Functional Programming languages decompose a problem into a set of functions. Each function is independent of any external state, operating only on the inputs they received to generate their outputs. The programming language Haskell is an example of a functional programming language. Some programming languages are considered to be hybrid languages; that is they allow developers to utilize a combination of difference approaches within the same program. Python is an example of a hybrid programming language as it allows you to write very procedural code, to use objects in an object oriented manner and to write functional programs. Each of these approaches is covered in this book.

**Python Libraries:** As well as the core language, there are very many libraries available for Python. These libraries extend the functionality of the language and make it much easier to develop applications. These libraries cover

• Web frameworks such as Django/Flask,

• email clients such as smtplib (a SMTP email client) and imaplib (an IMAP4 email client),

• content management operations such as the Zope library,

• lightweight concurrency (running multiple operations at the same time) using the Stackless library,

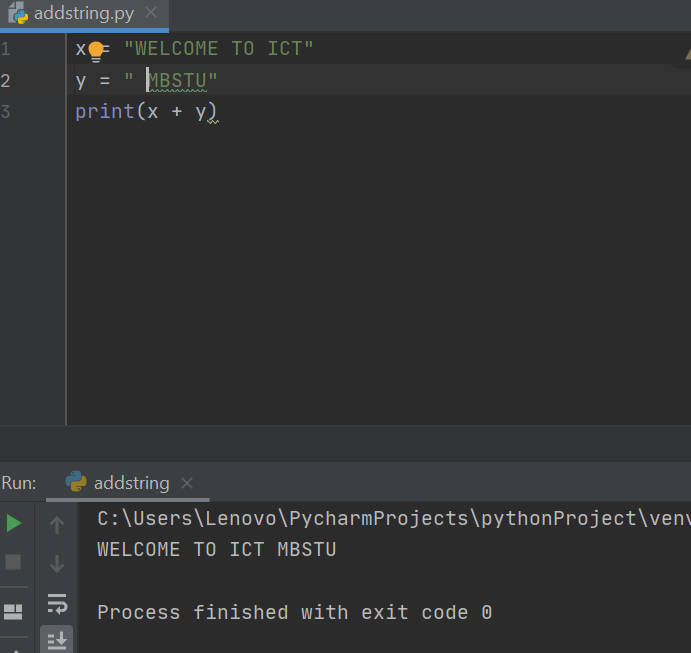
• the Generation of Microsoft Excel files using the Python Excel library,

• graphics libraries such as Matplotlib and PyOpenGL,

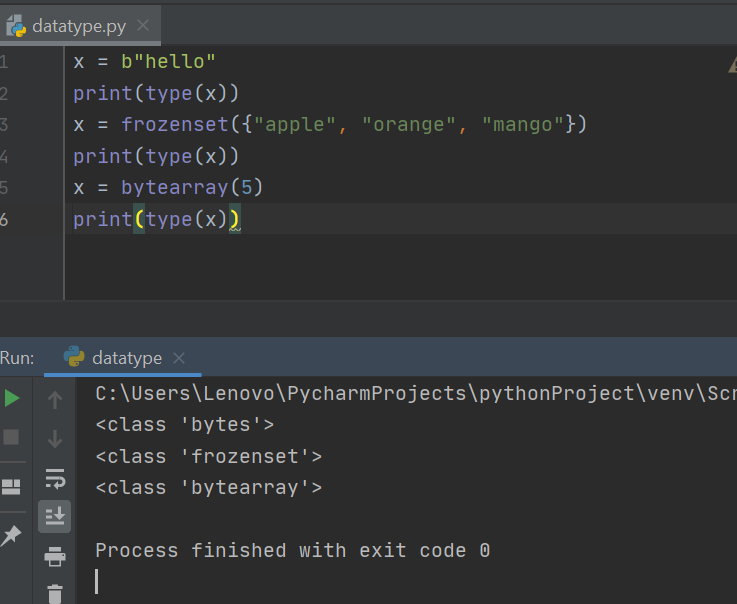
• machine learning using libraries such as SKLearn and TensorFlow.

Some code with Python programming Language:

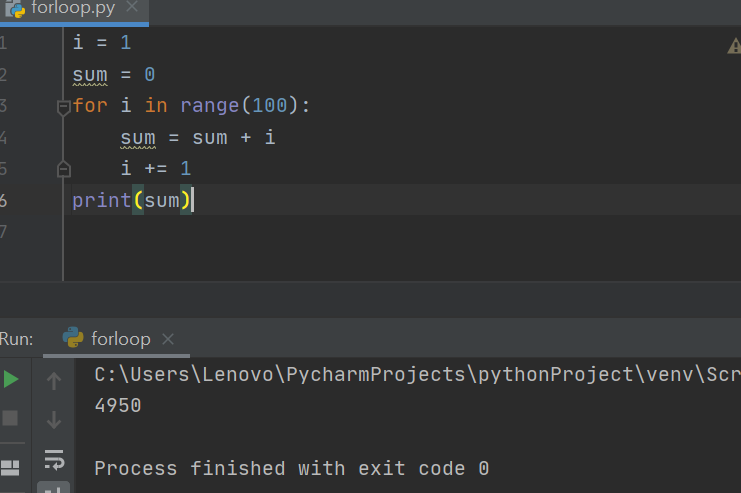
1. Adding two string



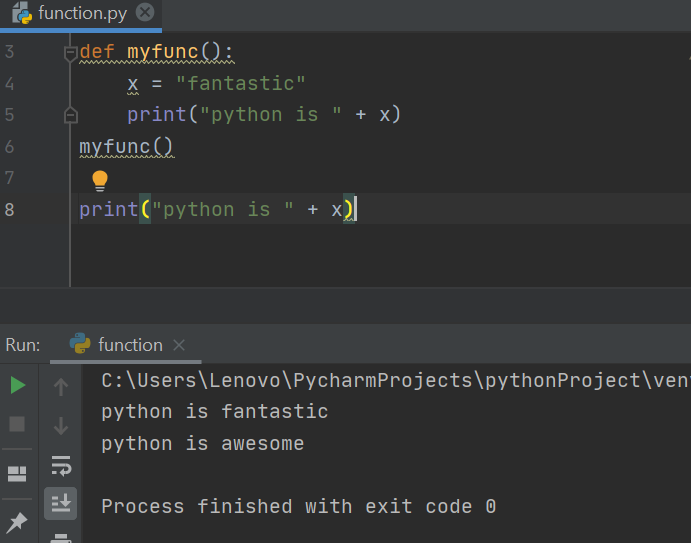
2.Data types:



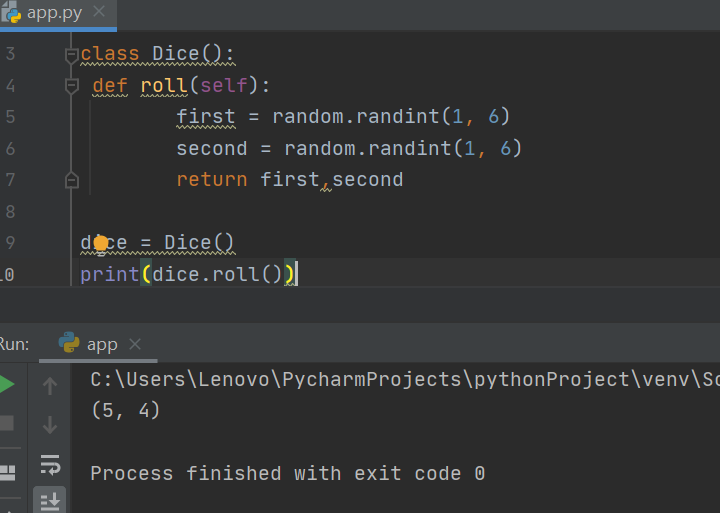
2.For Loop in python:



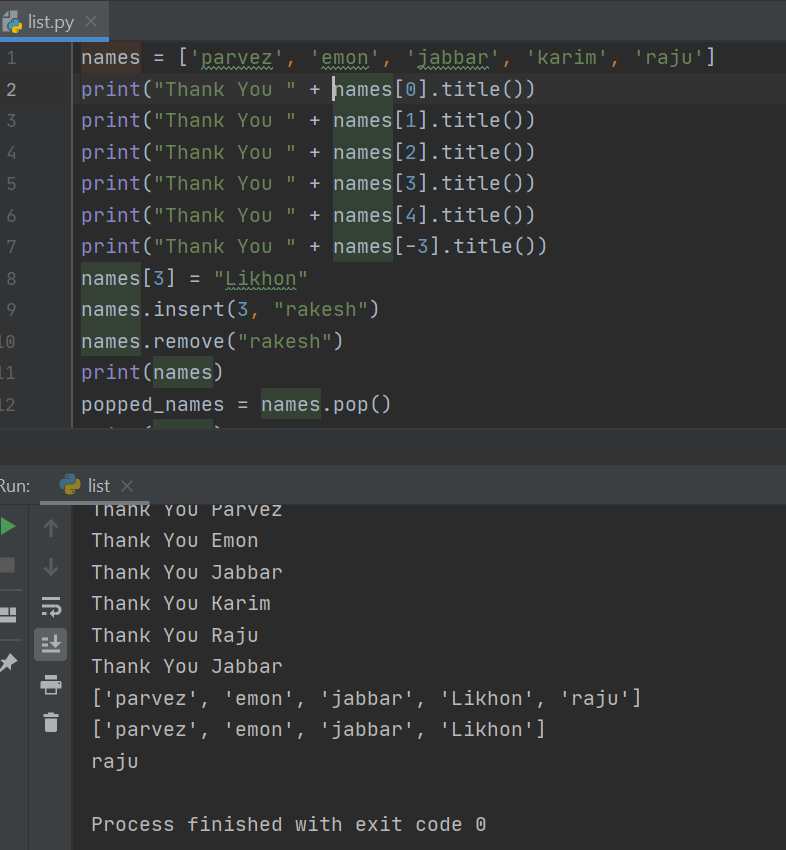
2.Create a variable inside a function with the same name as the global variable:



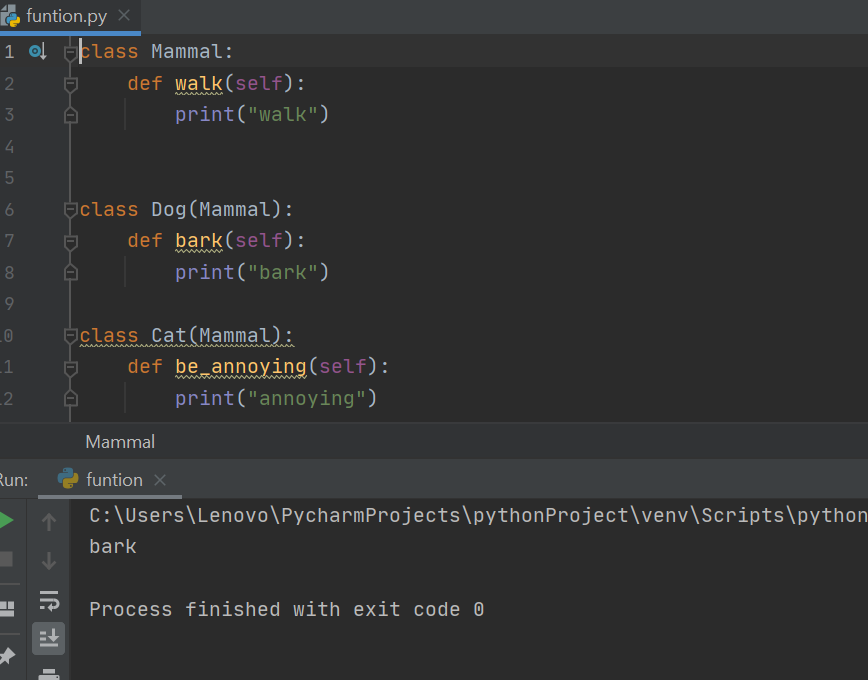
5. Random Numbers:



6. Lists in Python:



7. Class and Method:



**Conclusion:** Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.