Introduction to python

Name: Christine Kakina
Email: kristine.kakina@strathmore.edu

Unit Overview

Introduction to Python Programming

- Overview of Python programming language
- Installing Python and setting up the development environment (IDEs, text editors)
- Running Python programs and using the interactive shell

Unit Overview

Python Basics

- Variables and data types: numbers, strings, booleans
- Operators: arithmetic, assignment, comparison, logical
- Basic input and output: print function, input function

Unit Overview

Python Functions and Modules

- Defining and calling functions
- Function parameters and return values
- Local and global variables
- Organizing code with modules and packages
- Importing and using modules

Unit Overview

Python Control Flow (Conditional Logic and Loops)

- if, else, and elif statements
- for and while loop
- Control Statements

Unit Overview

Python Data Structures

- Lists: creating, indexing, slicing, modifying
- Tuples: creating, accessing elements, immutability
- Dictionaries: creating, accessing values, adding and removing items
- Sets: creating, adding and removing elements, set operations

Unit Overview

Python Exception Handling

- Understanding and handling exceptions in Python
- Using try-except blocks for error management
- Raising exceptions and creating custom exceptions

Unit Overview

**File handling and Input/Output Operations

- Opening and closing files
- Reading data from files: text files, CSV files
- Writing data to files: creating new files, appending to existing files
- Handling file errors and exceptions

Word

Practice.

Grasp the concept.

Try and relate to practical situations.

Introduction to programming

Programming is the process of creating instructions for a computer to follow. These instructions are called code, and they are written in a programming language. A programming language is a set of rules that define how code can be written.

Introduction to programming

TYPES OF PROGRAMMING LANGUAGES:

High-level languages are designed to be easy for humans to read and write. Used for general-purpose programming, such as web development, data science, and machine learning. Some examples of high-level programming languages include Python, Java, and C++.

Low-level languages are designed to be close to the machine code that computers understand. Used for system programming, such as operating systems and embedded systems. Some examples of low-level programming languages include Assembly and C.

Assembly language is a low-level programming language that is used to directly control the hardware of a computer. It is typically used for writing very efficient code, such as for embedded systems.

Scripting languages are a type of high-level programming language that is designed to be executed quickly and easily. They are typically used for web development, data science, and machine learning. Some examples of scripting languages include Python, JavaScript, and Ruby.

Concepts:

- **Variables**: Variables are used to store data. Variables are given names, and they can be used to refer to data.
- **Data types**: Data types define the type of data that can be stored in a variable. Some common data types include integers, floats, strings, and lists.
- **Operators**: Operators are used to perform operations on data. Some common operators include addition, subtraction, multiplication, and division.
- **Control flow**: Control flow statements allow you to specify the order in which statements are executed. Some common control flow statements include if statements, while loops, and for loops.
- **Functions**: Functions are blocks of code that can be reused in different parts of a program.
- **Modules**: Modules are files that contain Python code. Modules can be imported into other programs. Modules are used to organize and structure code, and to make code more reusable.
- **Data structures**: Data structures are ways of organizing data. Data structures are used to store and manipulate data in an efficient way.
- **OOP**: Object-oriented programming is a programming concept that uses objects to represent data and behavior. Objects are made up of attributes and methods. Attributes store data, and methods perform actions on data.
- **Error handling**: Error handling is the process of dealing with errors that occur in a program.

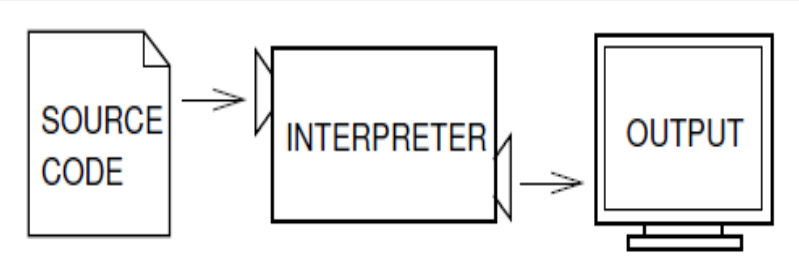
EXECUTION – Interpreter V Compilers

- Two ways to convert high-level languages into low-level languages, through: interpreters or compilers.

Interpreter:

•An **interpreter** reads a high-level program and executes it, meaning that it does what the program says. It processes the program a little at a time, alternately reading lines and performing computations
•An interpreter is a program that directly executes source code line by line.

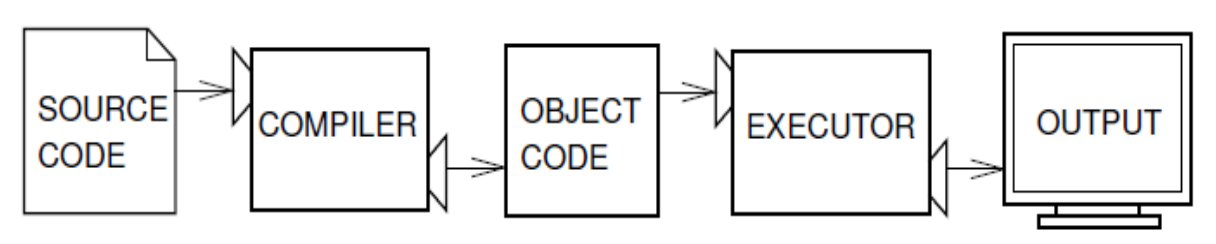
- It reads the source code, translates it into machine code or bytecode, and immediately executes each line.



Compiler:

A **compiler** is a program that translates the entire source code into machine code or bytecode before execution

In this context, the high-level program is called the source code, and the translated program is called the object code or the executable.
Once a program is compiled, you can execute it repeatedly without further translation.



Python Programming Language

History:
Python was created in the late 1980s by Guido van Rossum, released 1991.

Design philosophy:
• Python is designed to be easy to read and write. It is also designed to be powerful and versatile. Made to be fun to use

Features:
•Python has a number of features that make it a popular programming language, including:

- ❖ Simple syntax
- ❖ Dynamic typing
- ❖ Object-oriented programming
- ❖ Powerful libraries
- ❖ Free and open-source

Why Python?

- Python is a general-purpose programming language
- It is high level – programmer friendly, programmer does not need to perform memory management, or know how to write assembly language, easier to understand.
- Easy to learn – simple syntax - readable and maintainable code
- Multi-programming uses – object oriented and structured programming
- Versatile: Python can be used for a variety of tasks, including data science, machine learning, and web development.
- Open Source – The source code is freely available
- Large Community - large and active community of developers who are constantly creating new libraries and tools.
- It is flexible and OS friendly.

Media using python

Python USEs

- Web Development – Django, Flask
- Application Development –
- Desktop Apps and GUI – PyQt, Tkinter, PyGUI
- Game Development - PyGame
- Operating Systems – Used in most Linux distributions,
- AI and Machine Learning – NumPy, Pandas, Seaborn, sklearn, pyspark

Running Python

•Python is considered an interpreted language because Python programs are executed by an interpreter.
•There are two ways to use the interpreter: interactive mode and script mode.
•In interactive mode, you type Python programs and the interpreter displays the result:
•>>> 1 + 1
•2
•The chevron, >>>, is the prompt the interpreter uses to indicate that it is ready. If you type1 + 1, the interpreter replies 2.

Script Mode

•Alternatively, you can store code in a file and use the interpreter to execute the contents of the file, which is called a script. By convention, Python scripts have names that end with .py.

Debugging

•Programming is prone to errors, and in programming these errors are referred to as bugs and the process of tracking them down is called debugging.

•Three kinds of errors can occur in a program: syntax errors, runtime errors, and semantic errors

Syntax errors

•Python can only execute a program if the syntax is correct; otherwise, the interpreter displays an error message.

•Syntax refers to the structure of a program and the rules about that structure.

•For example, parentheses have to come in matching pairs, so (1 + 2) is legal, but 8) is a syntax error.

Runtime errors

•The second type of error is a runtime error, so called because the error does not appear until after the program has started running.

•These errors are also called exceptions because they usually indicate that something exceptional (and bad) has happened.

•Example: memory errors, error in source code e.g. division by zero, referencing missing files

Semantics errors

•The third type of error is the semantic error. If there is a semantic error in your program, it will run successfully in the sense that the computer will not generate any error messages, but it will not do the right thing.

•It will do something else. Specifically, it will do what you told it to do.

•Identifying semantic errors can be tricky because it requires you to work backward by looking at the output of the program and trying to figure out what it is doing.

Debugging

•Debugging is also like an experimental science. Once you have an idea about what is going wrong, you modify your program and try again.

•If your hypothesis was correct, then you can predict the result of the modification, and you take a step closer to a working program.

•The idea is that you should start with a program that does something and make small modifications, debugging them as you go, so that you always have a working program.

•For example, Linux is an operating system that contains thousands of lines of code, but it started out as a simple program Linus Torvalds used to explore the Intel 80386 chip. According to Larry Greenfield, “One of Linus’s earlier projects was a program that would switch between printing AAAA and BBBB. This later evolved to Linux.”

IDES TO USE:

- Python
- Anaconda 3.10 - https://www.anaconda.com/products/distribution
- Jupyter Notebooks – comes with Anaconda

Jupyter Notebook.

- Jupyter Notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations, and narrative text

- Jupyter Notebook is based on the concept of interactive computing, where you can run code blocks, see their output, and mix them with markdown text to create a document that can be easily shared and reproduced.

- The documents are saved as .ipynb files and can be exported to various formats, such as HTML, PDF, or slides.

- Jupyter Notebook supports multiple programming languages, including Python, R, Julia, and others, making it a flexible and versatile tool for data analysis and experimentation.

Thank you,
KAKINA.