**DAY 1**

**Introduction to Python**

Python is a versatile and popular programming language known for its simplicity and readability. It's widely used for web development, data analysis, machine learning, artificial intelligence, automation, and more. Here are a few common areas you might be interested in:

Control Structures: This includes topics like loops (for and while), conditional statements (if, elif, else), and how to control the flow of your program.

Functions: Understanding how to create and use functions in Python, as well as defining your own custom functions.

Data Structures: Python offers a variety of data structures like lists, tuples, dictionaries, and sets. Learning how to work with these structures is crucial.

File I/O: Reading from and writing to files is a common task in Python. You can learn how to do this effectively.

Modules and Libraries: Python has a rich ecosystem of libraries that can be imported and used in your code. For example, numpy for numerical operations, pandas for data manipulation, and many others.

Object-Oriented Programming (OOP): Python supports OOP concepts like classes and objects. You can learn how to create and use classes in Python.

Error Handling: Understanding how to handle exceptions and errors gracefully in your code.

Web Development: Python can be used for web development with frameworks like Django and Flask.

Data Science and Machine Learning: Python is widely used in data science and machine learning. You can explore libraries like scikit-learn, TensorFlow, and PyTorch for these areas.

Automation: Python is great for automating tasks on your computer. You can use it for scripting and creating bots.

Please let me know which specific aspect of Python programming you're interested in, or if you have any questions or need help with a particular Python-related task.

**DAY 2**

**Introduction to Variables**

Variables are a temporary storage space in a computer’s memory. When a variable’s value changes the program’s current state also changes. A variable act as a container to hold a different number of data items or values. All programming languages use variables, as they are among the most important elements in programming, and that is why a good understanding of variables will only make your job easier when writing programs. Variables are also used to move data between functions; this will be discussed later.

Every variable is created with an initial value. A variable can be in three states:

Variable creation (Declaration)

Variable assignment (Initialization)

Variable changed (Execution)

In Python identifiers are case sensitive, so for example, firstName, FirstName, FIRSTNAME, and firstname are four different identifiers

**NOTE**

 All of the methods, exceptions, and functions contained in

dir(\_\_builtins\_\_), dir(\_\_doc\_\_), dir(\_\_name\_\_), and dir(\_\_package\_\_) cannot be used as variable names.

**Casting**

Casting can be done in two ways

**Implicitly**- the compiler automatically casts a value from one data type to another when assured that there will be no data loss.

**Explicitly:** It does not cast from one data type to another if it will result in data loss.

**e.g.,** casting from a floating-point value to an integer value.

num1 = 15 #num1 variable is automatically assigned

#as an integer value

num2 = "30" #num2 variable is automatically assigned

#as a string value

ans = num1 + num2 #ans is assigned to num1 added to num2

print ("answer:", ans) #ans is printed

**SOLUTION**

num1 = 15 #num1 variable is automatically assigned

#to an integer value

num2 = "30" #num2 variable is automatically  assigned

#to a string value

ans = num1 + int(num2) #num2 is cast to integer type and added #to num1

print ( "num2 cast to integer  ",ans)

ans = str(num1) + num2   #num1 is cast to a string type

print( "num1 cast to string  ",ans) #to num1

print ("num2 cast to integer: ", ans)

ans = str(num1) + num2 #num1 is cast to string and num2 gets

            #added

print ("num1 cast to string:", ans)

**DAY 3**

**Introduction to Data Types**

1. **Integer (int)**: Represents whole numbers, both positive and negative, without a decimal point. Examples include -3, 0, and 42.
2. **Floating-Point (float)**: Represents real numbers, including numbers with a decimal point. Examples include -3.14, 0.0, and 3.14159.
3. **Double (double)**: Like a float but with greater precision. It's used for more precise calculations that require a larger number of decimal places.
4. **Character (char)**: Represents a single character, such as a letter, digit, or special symbol. Examples include 'A', '5', and '%'.
5. **String (str)**: Represents a sequence of characters. Strings are used to store text. For example, "Hello, World!" is a string.
6. **Boolean (bool)**: Represents a binary value, which can be either **True** or **False**. Booleans are often used for making decisions in conditional statements.
7. **Array**: A collection of values of the same data type. Arrays allow you to store multiple values under a single variable name. For example, an array of integers could be used to store [1, 2, 3, 4, 5].
8. **List**: Like an array, but in many programming languages, lists can hold values of different data types.
9. **Dictionary (or Map)**: Stores key-value pairs, where each key maps to a specific value. This is used for associative data storage. In Python, it's called a dictionary; in some other languages, it's referred to as a map or hash table.
10. **Object (or Class)**: In object-oriented programming, an object is an instance of a class. An object can contain data and methods that operate on the data. Objects are used for modeling real-world entities and their behaviors.
11. **Enumeration (enum)**: A custom data type that consists of a set of named values. It's often used to represent a fixed set of options or states.
12. **Date and Time**: Many programming languages have data types to work with dates and times, allowing you to represent and manipulate dates, times, and time intervals.
13. **Null/Undefined**: Represents the absence of a value. It's often used to indicate that a variable has not been assigned a value.
14. **Custom Data Types**: You can define your own custom data types by creating classes or structures in object-oriented programming languages.

**Conversions**

1 x = bool()

2 y = bool()

3

4 print ('Enter x as 1 or 0:')

5 x = int(input())

6

7 print ('Enter y as 1 or 0:')

8 y = int(input())

9

10 z = str(not bool(x and y))

11

12 print ('The Boolean value of x is', str(bool(x)))

13 print ('The Boolean value of y is', str(bool(y)))

14 print ('The Boolean value of (x and y) is', str(bool(x and y)))

15 print ('The Boolean value of (x not y) is', z)

**Day 4**

Operators are special symbols or keywords that perform operations on variables and values.

Also we Types of Operators

1. Arithmetic Operators- it includes +, -, \*, /, %, \*\* , and //
2. Comparison Operators- It includes ==,!=, <>, <=, >=
3. Logical Operators- they include And, Or, Not
4. Assignment Operators-
5. Bitwise Operators
6. Membership Operators
7. Identify Operators