CHAPTER 1

FUNDAMENTALS OF PYTHON

Topics:

1. Introduction to Python, History of Python, Python Features, Python Applications
2. Basic Structure of Python program
3. Keywords and Identifiers
4. Data types and Variables
5. Type Casting
6. Input-Output functions: input, print
7. Operators

- Difference between Scripting language and programming language

Scripting Language	Programming Language
A scripting language is a type of programming language designed for a runtime system to automate the execution of tasks.	A programming language is a computer language that is used to communicate with computers using a set of instructions.
It uses an interpreter to convert source code into machine code.	It uses a compiler to convert source code into machine code.
It is interpreted language or interpreter-based language	It is compiled language or compiler-based language
Examples include Perl, PHP,Python, JavaScript, etc.	Examples include C, C++, Java, Python, etc.
Execution of a script takes less time as scripts are generally short.	Execution of a program takes more time since they are compiled.
Do not create a **.exe** file.	These generate **.exe** files.
All the scripting languages are programming languages.	All the programming languages are not scripting languages.
There is less maintenance cost.	There is the high maintenance cost.

1. Introduction to Python, History of Python, Python Features, Python Applications

=> **Python** is a high-level, interpreted programming language known for its simplicity and readability. Designed by Guido van Rossum and first released in 1991, Python is popular among beginners and experienced developers alike for its straightforward syntax and wide range of applications.

=> **Python Features**

- **Easy to Read and Write:** Python's clear and easy-to-read syntax makes it a great choice for beginners. Its code often resembles plain English, which helps in understanding and writing code easily.
- **Interpreted Language**: Python is an interpreted language, meaning you can execute code directly without compiling it first. This allows for quick testing and iteration.
- **Dynamically Typed:** Python handles data types automatically. You don’t need to declare the type of a variable explicitly; Python determines it at runtime.
- **Versatile:** Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.
- **Extensive Standard Library:** Python comes with a rich set of libraries and modules that extend its functionality, allowing you to perform various tasks like file handling, web development, and data analysis.
- **Cross-Platform:** Python is available on various platforms such as Windows, macOS, and Linux, making it a flexible choice for different operating systems.

=> **Python Applications**

- **Web Development:** Frameworks like Django and Flask make Python a powerful tool for building websites and web applications.
- **Data Science and Machine Learning:** Libraries such as NumPy, pandas, and scikit-learn are widely used for data analysis and machine learning projects.
- **Automation:** Python is commonly used for scripting and automating repetitive tasks, such as data processing or file management.
- **Software Development**: Python is used to develop desktop applications and games, leveraging tools like Tkinter for GUIs or Pygame for game development.

- **Networking:** Python's libraries and frameworks are used to build network applications, perform network analysis, and handle protocols.

2. Basic Structure of Python program

=> Python Basic Structure key points is shown below

1. Comments

=> Comments are notes you add to your code to explain what it does. They are ignored by the Python interpreter and are useful for documenting your code.

2. Variables

=> Variables are used to store data that your program can use. You create a variable by assigning a value to it.

3. Data Types

=> Python supports various data types, such as:

String, Integer, Float, Boolean

4. Basic Syntax

=> **Indentation**: Python uses indentation (spaces or tabs) to define blocks of code. Consistent indentation is crucial as it defines the structure of the program.

=> **Statements**: Each line of code that performs an action is called a statement. Statements are executed one after another.

5. Functions

=> Functions are reusable blocks of code that perform a specific task. They help organize code into manageable sections.

- Defining a function: Use the def keyword.
- Calling a function: Use the function's name followed by parentheses.

6. Control Flow

=> Control flow statements help you make decisions and repeat actions.

- Conditional Statements: Use if, elif, and else to execute code based on conditions.
- Loops: Use for and while loops to repeat actions.

=> Putting It All Together

```
# This program greets the user and calculates the sum of two numbers
def main():
        name = "Alice"
        age = 30
        # Greet the user
        print( "Hello, {name}!")
        # Check if the user is an adult
        if age > 18:
            print("You are an adult.")
        else:
            print("You are not an adult.")
        # Calculate the sum of two numbers
        x = 5
        y = 10
        sum = x + y
        print("The sum of {x} and {y} is {sum}")
main()
```

3. Keywords and Identifiers

=> Keywords

Keywords are reserved words in Python that have special meanings and cannot be used for anything other than their intended purpose. They are the building blocks of the Python language, defining the structure and control flow of programs.

Keywords in Python programming language				
False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

=> Identifiers

Identifiers are names used to identify variables, functions, classes, modules, or other objects in your code. Unlike keywords, identifiers are not reserved by Python and can be chosen by the programmer. However, they must follow specific rules:

Rules for Naming Python Identifiers

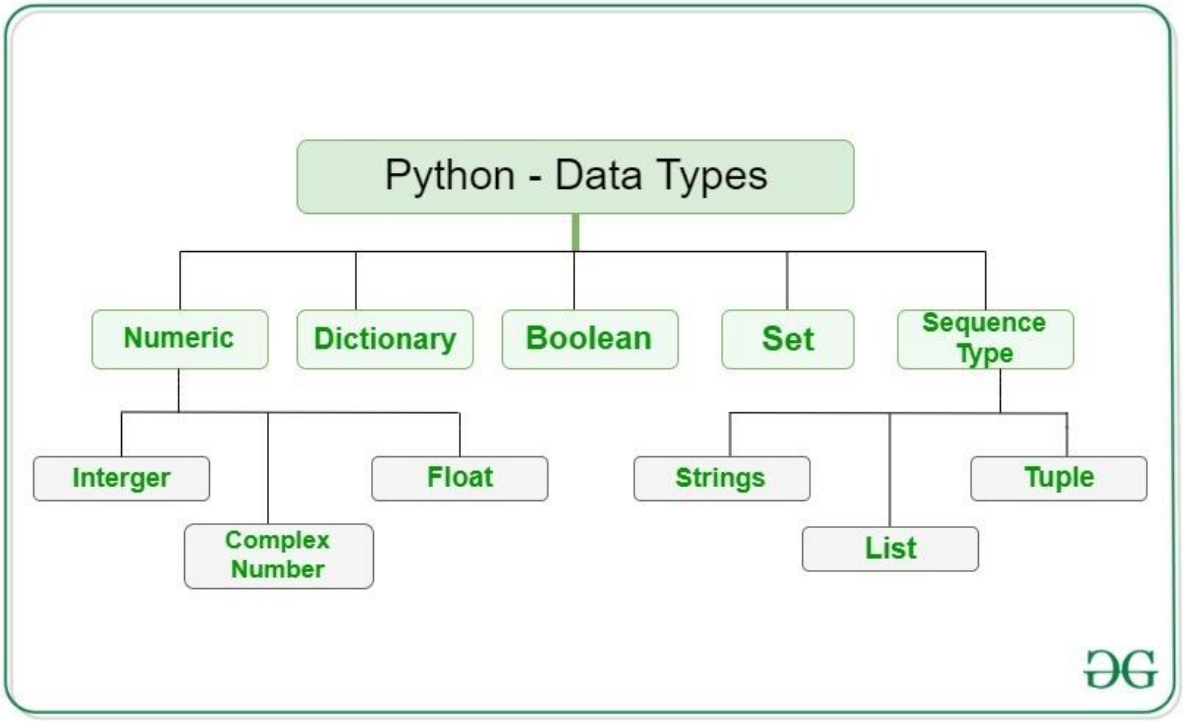
- It cannot be a reserved python keyword.
- It should not contain white space.
- It can be a combination of A-Z, a-z, 0-9, or underscore.
- It should start with an alphabet character or an underscore (_).
- It should not contain any special character other than an underscore (_).

Valid identifiers:

- *var1, _var1, _1_var, var_1,*
- ***Invalid Identifiers***
- *!var1, 1var, 1_var, var#1, var 1*

4. Data types and variable

=> **Data types** define the kind of data that can be stored and manipulated in a program. Python is dynamically typed, meaning you don't need to explicitly declare a variable's type.



Numbers

- **Integer (int)**: Whole numbers, positive or negative. Example: 42
- **Floating-point (float)**: Numbers with decimal points. Example: 3.14
- **Complex (complex)**: Numbers with real and imaginary parts. Example: 2 + 3j
- x = 10 # int
- y = 3.14 # float
- z = 1 + 2j # complex

Strings (str)

- Textual data enclosed in single quotes ('), double quotes ("), or triple quotes (''' or """ for multi-line strings). Example: "Hello, World!"
- message = "Hello, Python!"

Lists

- Ordered, mutable collections of items, which can be of different types. Lists are defined using square brackets ([]). Example: [1, 2, 3, "Python"]
- fruits = ["apple", "banana", "cherry"]

Tuples

- Ordered, immutable collections of items, defined using parentheses (()). Example: (1, 2, 3, "Python")
- coordinates = (10.0, 20.0)

Dictionaries (dict)

- Unordered collections of key-value pairs, defined using curly braces ({}). Example: {"name": "Alice", "age": 30}
- person = {"name": "Alice", "age": 30}

Sets

- Unordered collections of unique items, defined using curly braces ({}). Example: {1, 2, 3}
- unique_numbers = {1, 2, 3}

Booleans (bool)

- Represents truth values: True or False
- is_active = True

=> **Variables**

Variables are used to store data that can be referenced and manipulated in a program.

=> Defining Variables

- Assign a value to a variable using the assignment operator (=). Python variables do not require explicit type declarations.

  age = 25

  name = "Alice"

  height = 5.6

=> Naming Conventions

- **Start with a Letter or Underscore**: Variable names must start with a letter (a-z, A-Z) or an underscore (_).
- **Use Descriptive Names**: Choose names that indicate the variable's purpose, such as total_amount instead of ta.
- **Avoid Keywords**: Do not use Python keywords as variable names.
- **Follow Conventions**: Use snake_case for variables (e.g., user_age) and camelCase for classes (e.g., StudentRecord).

  user_age = 30

  account_balance = 1000.50

=> Dynamic Typing

- Python automatically handles type changes during runtime. A variable can hold different types of data at different times.

  data = "Hello"

  data = 123 # The same variable now holds an integer

5. Type Casting

=> **Type casting** (or type conversion) is the process of converting a variable from one data type to another. This is useful when you need to perform operations that require specific data types or when you want to ensure that data is in the correct format for a given task.

=>

Example

```
# Converting a float to an integer
height = 5.9
height_int = int(height)  # Results in 5

# Converting an integer to a string
age = 25
age_str = str(age)  # Results in "25"

# Converting a string to a float
```

```
price_str = "19.99"
price_float = float(price_str)  # Results in 19.99

# Converting a string to a list
text = "hello"
text_list = list(text)  # Results in ['h', 'e', 'l', 'l', 'o']

# Converting a list to a tuple
numbers_list = [1, 2, 3, 4]
numbers_tuple = tuple(numbers_list)  # Results in (1, 2, 3, 4)

# Converting a list to a set
numbers_list = [1, 2, 2, 3, 3]
numbers_set = set(numbers_list)  # Results in {1, 2, 3}
```

6. Input-Output functions: input, print

=> In Python, handling input and output is essential for interacting with users and displaying information. The primary functions for these tasks are input() and print().

=> **1. print() Function**

The print() function is used to display information to the user or output data to the console.

Syntax

print(value, ..., sep=' ', end='\n')

Example

print("Hello, World!")

=> **2. input() Function**

The input() function is used to read user input from the console. It returns the input as a string.

Syntax

input(“prompt”)

Example

```
name = input("Enter your name: ")

print("Hello, " + name + "!")
```

7. Operators in python

=> Operators are special symbols in Python that perform operations on values or variables. They are essential for performing arithmetic, comparison, logical, and other operations.