Choice based assignment python 1

1. What is Python? List any five features of Python.

Python is a **high-level, interpreted, general-purpose programming language** created by Guido van Rossum. It's super popular because it's easy to read, beginner-friendly, and powerful for everything from web dev to AI.

Five features:

1. **Easy syntax** – looks like English, super readable.
2. **Interpreted language** – runs line-by-line, no need to compile.
3. **Dynamically typed** – no need to declare data types manually.
4. **Large standard library** – built-in modules for almost everything.
5. **Portable & cross-platform** – runs on Windows, Linux, macOS without changes.

2. Why is indentation important in Python syntax?

In Python, **indentation is not just styling**... it's the actual rule that defines code blocks.

While languages like C or Java use `{}`, Python uses **spaces/tabs** to group statements.

Example:

```
if x > 0:
    print("Positive")  # Indented block
```

If indentation is wrong, Python instantly throws an error.

So indentation decides *where a block starts and ends*, making the code clean and readable.

3. Define an identifier. Describe the rules for naming identifiers in Python.

An **identifier** is the name you give to variables, functions, classes, modules, etc.

Rules for Naming Identifiers:

- Can contain **letters (A–Z, a–z), digits (0–9), and underscore (_)**
- **Cannot start with a digit**
- **Cannot use keywords** like `class`, `if`, `while`
- **Case-sensitive** → `Age`, `age`, `AGE` are different
- Should not contain symbols like `@, $, %, space`

---

4. How Python's interpreted nature affects program execution?

Python is an **interpreted language**, meaning:

- Code runs **line-by-line**
- Errors are caught **during runtime**, not before
- Programs start executing faster (no separate compile step)
- But usually slower than compiled languages like C because interpretation happens on the fly

This makes Python great for beginners, rapid development, testing, and scripting.

---

5. A small program with single-line and multi-line comments

```
# This is a single-line comment

"""
This is a multi-line comment
used to describe a bigger idea.
"""
```

```
name = "Bharath"  # Another single-line comment
print("Hello", name)
```

---

6. Compare Python and C-like syntax – identify two major differences

Difference 1: Block definition

- **Python** uses *indentation*
- **C-like languages** use *curly braces* `{}`

Difference 2: Variable declaration

- **Python**: no need to declare type

  ```
  x = 10
  ```

- **C**: must declare type

  ```
  int x = 10;
  ```

Other differences: no semicolons in Python, simpler syntax, dynamic typing.

---

7. Write a simple Python script showing readability & ease of use

```
# Program to calculate total price

quantity = 5         # Number of items
price_per_item = 20   # Cost of each item

# Calculate total
total_price = quantity * price_per_item
```

```
# Display result
print("Total Price:", total_price)
```

This program uses:

- Meaningful identifiers
- Clean indentation
- Comments
- Very readable flow

---

8. Identifier validity check

Identifier	Valid?	Reason
_value	✔️ Valid	Starts with underscore, allowed
2name	❌ Invalid	Cannot start with a digit
total_sum or total_num	❌ Invalid	Contains spaces and `or` → Python treats it as two identifiers and a logical operator
class	❌ Invalid	It's a Python **keyword**
MyVariable	✔️ Valid	Follows all rules

---

python assignment

---

9) Define a variable and indentation in python. Explain the rules for naming variables and the purpose of it with suitable examples.

A **variable** in Python is basically a name that stores a value. It behaves like a container where you keep data for later use.
Example:

```
x = 10
name = "Bharath"
```

Indentation means giving spaces at the start of a line.
Python uses indentation to define code blocks like loops, functions, and conditions.
Example:

```
if x > 5:
    print("Hello")
```

The `print` must be indented, or Python will throw an error.

Rules for naming variables:

1. Must start with a **letter or underscore**
   Example: `age`, `_value`
2. Cannot start with a number
   ❌ `1num`
3. Can contain letters, digits, underscore
   ✔ `total_marks`, `val2`
4. Case-sensitive
   `Age` and `age` are different.
5. Cannot use Python keywords
   ❌ `for`, `while`, `class`

Purpose of variables:

- Store and reuse values
- Make programs readable
- Perform calculations
- Manage data efficiently

Example:

```
price = 50
qty = 3
total = price * qty
print(total)
```

---

10) What are literals in python? Explain different types of literals with examples.

Literals are fixed values written directly in the code.

Types of Python Literals:

1. Numeric Literals

```
a = 10      # integer
b = 3.5     # float
c = 2+3j    # complex
```

2. String Literals

```
name = "Python"
```

3. Boolean Literals

```
is_login = True
```

4. None Literal

```
data = None
```

5. Collection Literals

```
list1 = [1, 2, 3]
tuple1 = (4, 5, 6)
set1 = {7, 8, 9}
dict1 = {"name": "Bharath", "roll": 101}
```

---

11) List different standard datatypes available in python, give one-line definition for each.

Table 1

Datatype	Definition
int	Whole numbers without decimals
float	Numbers with decimals
complex	Numbers with real + imaginary part
bool	Represents True/False
str	Used to store text
list	Ordered, changeable collection
tuple	Ordered, unchangeable collection
set	Unordered collection of unique values
dict	Stores data as key–value pairs
NoneType	Represents no value

---

12) Explain the features of python that supports Rapid Application Development.

Python is super fast for building apps because of:

1. Simple Syntax

Looks like English → easy to write and understand.

2. Large Standard Library

Has built-in modules for file handling, math, networking, databases, etc.

3. Third-party Libraries

NumPy, Pandas, Django, Flask… everything is ready-made.

4. Dynamic Typing

You don’t declare variable types → saves time.

5. Interpreted Language

Runs instantly without compilation.

6. Cross-platform Support

Same code works on Windows, Linux, Mac.

7. Supports Multiple Paradigms

Object-oriented + functional + procedural.

These features allow developers to build apps super quickly.

---

13) Explain the concept of type conversion in python with example.

Type Conversion means converting data from one type to another.

There are two types:

1. Implicit Conversion (Automatic)

Python converts automatically during expressions.

```
a = 5      # int
b = 2.5    # float
c = a + b  # converts int → float
print(c)
```

2. Explicit Conversion (Manual)

Using conversion functions.

```
num = "25"
val = int(num)
print(val + 5)
```

---

14) Write a program to demonstrate the use of different arithmetic operators in python.

```
a = 10
b = 3

print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
print("Floor Division:", a // b)
```

```
print("Modulus:", a % b)
print("Exponent:", a ** b)
```

---

15) Analyze the difference between Python syntax and syntax of any other programming language like C / Java.

Table 2

Feature	Python	C / Java
Braces	Uses indentation	Uses `{}` braces
Semicolon	Not required	Required (`;`)
Variable Declaration	No datatype needed	Must specify datatype
Code Length	Short and simple	Longer and detailed
Compilation	Interpreted	Compiled
Main Function	Not compulsory	Must have `main()`
Memory Management	Automatic (garbage collector)	Manual / semi-auto

Example Comparison

Python

```
if x > 10:
    print("High")
```

C

```
if(x > 10) {
    printf("High");
}
```

Python is cleaner and more readable.
