1.what is python?

Python is a high-level, interpreted, and object-oriented programming language. It is dynamically typed and easy to read, which makes development faster and more efficient.

it is widely used in web development, data science, machine learning, automation, and many other domains.

2. Is Python an interpreted language? If yes, explain.

Yes, Python is an interpreted language. This means that Python code is executed line by line by the Python interpreter, without the need for a separate compilation step

without the need for a separate compilation step

3.what is the difference between interpreter and compiler?

A compiler translates the entire source code of a program into machine code in one go, creating an executable file. Once compiled, the program runs very fast because it's already in machine code. Examples: C, C++.

On the other hand, an interpreter translates and executes code line by line at runtime. This makes debugging easier since errors are shown immediately, but it can be slower than compiled programs. Examples: Python, JavaScript.

4.what is data? Data types in python?

Data refers to the information that a program processes and stores. In Python, every value we use or create is treated as an object, and each object has a specific type that defines what kind of data it can hold and what operations can be performed on it

Numeric Types

- *int → whole numbers (e.g., 10, -5)*
- *float → decimal numbers (e.g., 3.14, -2.5)*
- *complex → complex numbers (e.g., 3 + 4j)*

Sequence Types

- *str → strings (e.g., "Hello")*
- *list → ordered, mutable collection (e.g., [1, 2, 3])*
- *tuple → ordered, immutable collection (e.g., (1, 2, 3))*
- *range → sequence of numbers (e.g., range(5))*

Mapping Type

- *dict → key-value pairs (e.g., {"name": "Prem", "age": 23})*

Set Types

- *set → unordered collection of unique items (e.g., {1, 2, 3})*
- *frozenset → immutable version of a set*

Boolean Type

- *bool → True or False*

Binary Types

- *bytes, bytearray, memoryview (used for handling binary data)*

5.what is list in python? Give an example for that

In Python, a list is a built-in data structure that allows us to store multiple items in a single variable. Lists are ordered and can hold elements of different data types such as integers, strings, or even other lists.

Lists are defined using ***square brackets []***

Example

```
my_list = [10, "Python", 3.14, True]
```

6.what is dict in python? Give an example for that.

A dictionary in Python is an ***unordered, mutable collection of key–value pairs****. Keys must be unique and immutable, while values can be of any type.*

It is unordered mutable and indexed by keys instead of numeric indexes.

Egcode:

```
student = {"name": "Avinash", "age": 25, "course": "Python"}

print(student["name"])  # Avinash

student["age"] = 26     # modify value
```

7.what is tuple in python? Give example for that

A tuple in Python is an ***ordered, immutable collection*** *of items. It is similar to a list, but once created, its elements* ***cannot be changed****. Tuples are defined using* ***round brackets ()***

Eg code:

```
my_tuple = (10, "Python", 3.14)

print(my_tuple[0])    # 10

print(my_tuple[1])    # Python
```

8.what is the difference between mutable and immutable data types in python?

In Python, data types are classified into mutable and immutable based on whether their values can be changed after creation.

Mutable Data Types:
These can be modified after creation. We can add, remove, or change elements without creating a new object.
Examples → list, dict, set, bytearray

Eg code:

```
my_list = [1, 2, 3]

my_list[0] = 99

print(my_list)  # [99, 2, 3]  ✅ modified in place
```

Immutable Data Types:
These cannot be modified after creation. Any change creates a new object in memory instead of modifying the original.
Examples → int, float, str, tuple, frozenset, bytes

Eg code:

```
my_str = "hello"

new_str = my_str.upper()

print(my_str)   # "hello"  (unchanged)

print(new_str)  # "HELLO"  (new object created)
```

9.what is difference between tuple and list in python?

Both **lists** and **tuples** are sequence data types in Python, but they have some key differences:

Feature	**List**	**Tuple**
Definition	Ordered collection, defined with []	Ordered collection, defined with ()
Mutability	**Mutable** → can be changed (add, remove, update elements)	**Immutable** → cannot be changed after creation
Performance	Slightly **slower** due to mutability	**Faster** (because immutable)
Use Case	When data may need to change (e.g., dynamic data)	When data should not change (e.g., fixed values, coordinates)
Methods	Many methods like append(), remove(), sort()	Fewer methods (mainly count, index)

10. how can we mutate the list in python?

In Python, a list is **mutable**, which means we can change its contents after creation. Mutation can be done by **updating existing elements, adding new elements, or removing elements**. For example, we can directly assign a new value to an index to update it.

Eg code:

Changing elements by index

```
nums = [10, 20, 30]

nums[1] = 99

print(nums)  # [10, 99, 30]
```

Adding elements

```
nums.append(40)       # adds at the end
nums.insert(1, 15)    # inserts at index 1
print(nums)  # [10, 15, 99, 30, 40]
```

Removing elements

```
nums.remove(99)       # removes by value
popped = nums.pop(2)   # removes by index
print(nums)  # [10, 15, 40]
```

Extending with another list

```
nums.extend([50, 60])
print(nums)  # [10, 15, 40, 50, 60]
```

Slicing for replacement

```
nums[1:3] = [111, 222]
print(nums)  # [10, 111, 222, 50, 60]
```

11. what is the difference between append and insert methods in python? While mutating the list?

Both append () and insert () are used to add elements to a list, but they differ in how they work.

append () adds a new element **only at the end** of the list.

insert () allows us to add a new element at a **specific index** in the list.

Syntax :

```
List.append(item)
List.insert(index, item)
```

Eg code:

```
nums = [10, 20, 30]

# Using append() → adds at the end
nums.append(40)
print(nums)   # [10, 20, 30, 40]

# Using insert() → adds at a given index
nums.insert(1, 15)
print(nums)   # [10, 15, 20, 30, 40]
```

12. what is the difference between pop and pop(index) in python?

The pop() method in Python is used to remove and return elements from a list

pop() removes and returns the **last element** of the list by default.

pop(index) removes and returns the element at the **specified index**.

Eg code:

```
my_list = [10, 20, 30, 40]
print(my_list.pop())    # Removes last → 40
print(my_list)          # [10, 20, 30]
print(my_list.pop(1))   # Removes element at index 1 → 20
print(my_list)          # [10, 30]
```

13. how can you mutate the dict in python? Give an example.

Dictionaries in Python are **mutable**, which means we can change them after creation. Mutation can be done in the following ways:

1. **Updating values** for existing keys.
2. **Adding new key–value pairs**.
3. **Removing elements** using pop(), del, or popitem().

Eg code:

```
student = {"name": "Avinash", "age": 25}
# Update value
student["age"] = 26
# Add new key-value pair
student["course"] = "Python"
# Remove a key
student.pop("name")
print(student)  # {'age': 26, 'course': 'Python'}
```

Dictionaries are mutable. We can update values, add new key–value pairs, or remove keys using methods like update(), pop(), popitem(), del, or clear().

14.write nested dictionaries for electronic product in python.

```
# Nested dictionary for electronic products
electronics = {
  "laptop": {
    "Dell": {
      "model": "Inspiron 15",
      "price": 55000,
      "features": ["i5 Processor", "8GB RAM", "512GB SSD"]
    },
    "HP": {
      "model": "Pavilion 14",
      "price": 60000,
      "features": ["i7 Processor", "16GB RAM", "1TB SSD"]
    }
  },
  "mobile": {
    "Samsung": {
      "model": "Galaxy S23",
      "price": 75000,
      "features": ["AMOLED Display", "8GB RAM", "128GB Storage"]
    },
    "Apple": {
      "model": "iPhone 15",
      "price": 120000,
      "features": ["Super Retina Display", "6GB RAM", "256GB Storage"]
    }
  },
  "television": {
    "Sony": {
      "model": "Bravia X90",
      "price": 90000,
      "features": ["4K UHD", "Smart TV", "55 inch"]
    },
    "LG": {
```

```
            "model": "NanoCell 80",
            "price": 85000,
            "features": ["NanoCell Display", "Smart TV", "50 inch"]
        }
    }
}
# Example: Accessing nested values
print(electronics["laptop"]["Dell"]["features"])
print(electronics["mobile"]["Apple"]["price"])


output:
['i5 Processor', '8GB RAM', '512GB SSD']
120000
```

Here we have **3 categories (laptop, mobile, television)** → inside each category, multiple brands → each brand has model, price, and features.

15. write a list of dictionaries in python.

```
# List of dictionaries
students = [
  {"id": 1, "name": "Prem", "age": 23, "course": "Python"},
  {"id": 2, "name": "Kiran", "age": 22, "course": "Java"},
  {"id": 3, "name": "Ravi", "age": 24, "course": "Data Science"},
  {"id": 4, "name": "Anu", "age": 21, "course": "Web Development"}
]

# Example: accessing values
print(students[0]["name"])   # Prem
print(students[2]["course"]) # Data Science

output:
Prem
Data Science
```