1.What is python?

Ans. Python is high level, easy to read, multipurpose, powerful computer programing language.

Developed by Guido Van Rossum a dutch programmer. Python use to build website, software, automate task and analyze data.

2.what are the features of python.

Ans.

1.Easy to Learn and Use.

Python has a clean and readable syntax, like plain English. This makes it easy for beginners to learn.

2.Free and Open source

Python is free to download and use. You can also see and change its source code.

3.interpreted language.

Python runs the code line by line, which makes debugging easier.

4.High-level language

You don't need to worry about low-level details like memory management — Python handles that for you.

5. Object-Oriented and Functional.

3.What are the applications of python.

Ans.

1. Web Development : Python is widely used for building web applications and frameworks. Ex. Example: Instagram and YouTube use Python. Tools: Django, Flask (Python web frameworks)
2. Data science and analysis : Helps people analyze data, find trends, and make decisions. Used in fields like business, healthcare, and sports. Tools: Pandas, NumPy, Matplotlib
3. Machine learning & AI : Helps computers learn and make smart decisions. Used in self-driving cars, chatbots, and recommendation systems (like Netflix suggestions). Tools: TensorFlow, Scikit-learn, PyTorch
4. Automation / Scripting : Automates boring or repetitive tasks. Example: Renaming lots of files, sending emails automatically, or scraping data from websites.

5. Game Development : Used to make simple games. Example: Games like “Civilization IV” used Python. Tools: Pygame

4.What is variable.

Ans. Is nothing but it has a temporary memory locator where we can store the data.
You can store something in it — like a number, a word, or other types of data — and give that box a name so you can use it later. Variable case also sensitive.

Ex .a=10 ,b=”Raj” – a and b is variable can store the values.

5.What is Datatype.

Ans. A datatype is classification that tells to the computer how to interpret the value of a data.

In python datatypes categories into to part

1.Mutable datatypes.

2.Immutable datatypes.

1.Immutable datatypes :

Immutable Data types means whose values can not change after they are created.

- Numarical data types – int, float, complex
- String -String is collection of similar and mixed values which is surrounded by ‘ ‘, “ “, ‘’’ ‘’’.
- Tuple – A tuple is a built-in data type in Python that allows you to store a sequence of items. Which is surrounded by (, ,)

2. mutable datatypes :

Mutable Data types means whose values can be change after they are created.

- List – List is a collection of similar and mixed values which is surrounded by [, ,] ex.[1,2,3,4]
- Dictionary – Dictionary is a collection of keys and value pairs which is surrounded by { , , } keys and values separated by ‘ , ‘ and keys are also unique.
- Set – Set is collection of unique elements which is surrounded by { , , } bracket and separated by ‘ , ‘.

6.What are the identifiers.

Ans. An identifier is the name used to identify a variable, function, class, module, or object.

```
name = "Alice"       # 'name' is an identifier

age = 25            # 'age' is an identifier


def greet():         # 'greet' is an identifier (function name)

   print("Hello")
```

7.What is the difference between mutable and immutable datatype.

- Mutable Data Types –
  1.Mutable Data types means whose values can change after they are created.
  2. Can be changed after creation.
  3. You can add, remove, or modify elements.
  4. Stored in memory in a way that allows updates.
- Immutable Data Types –
  1. Immutable Data types means whose values can not be change after they are created.
  2. Cannot be changed after creation.
  3. Any "change" creates a new object.
  4. More secure and used when data should stay constant.

8. What is the difference between list and tuple.

Ans. List –

1. List is a collection of similar and mixed values.
2. List values written inside square bracket.
3. Slower than tuple.
4. List is Mutable datatype.
5. Ex.List=[“Amar”,”Karan”,”Sai”]

Tuple –

1. Tuple is a collection of items that can not be changed.
2. Tuple values written inside round bracket.
3. Faster than list.
4. Tuple is Immutable datatype.
5. Tuple=(1,2,3)

9. What is the difference between list and set.

List –

1. List is a collection of similar and mixed values.
2. List values written inside square bracket.
3. List can follow indexing.
4. Ex.List=[“Amar”,”Karan”,”Sai”]

Set –

1. Set is a collection of unique elements.
2. Set values written inside {} bracket.
3. Set can’t follow indexing.
4. Ex.Set ={1,2,3,4}

10.What is Dictionary.

Ans. Dictionary is a collection of key-value pairs which is surrounding by { , , , }.

Keys and values separated by ‘ , ‘. Key are also unique. Dictionary can not followed indexing.

Ex. {‘a’:10,’ b’:20}

11.What is difference between split and join method of string.

Ans. split() –

1. Breaks a string into parts.
2. Return a list of words.
3. Split () string to list.

join () –

1. Combines a list of strings into one string.
2. Return a string.
3. join() list to string.

12. Difference between append and extend in list.

Ans. Append –

1. Adds a single elements to the end of the list.
2. Adds the entire object as one time.
3. Ex. a=[1,2,3] a.append ([4,5])   a=[1,2,3[4,5]]

Extend –

1. Adds multiple elements (from another list, tuple or iterable) to the end of the list.
2. Adds each element of the iterable individually.
3. a=[1,2,3] a.extend([4,5]) a=[1,2,3,4,5]

13.What is set default method does in dictionary.

Ans. The setdefault() method in dictionary is used to get the value of a key and if the key does not exist, it adds the key with a default value.

If the key is already present, it return the exiting value.

If the key is not present, it adds the key with the given default value, and the return that value.

Ex. a={'name':'Rohan','age':18}

a.setdefault('age',20)   op. a={'name':'Rohan','age':18}

a.setdefault('Location','Mumbai')   op. . a={'name':'Rohan','age':18,’Location’:’Mumbai’}

14. What is function.

Ans. A function is a block of code that perform a specific task and can be used multiple time in a program.

```
Ex. def add(a,b):

          Result = a + b

          Return result

Print(“The sum is: “ Result)
```

15.What are the types of function.

Ans. 1.Built-in function.

These are pre defined function that comes with python. You can use them directly without defining them yourself.

Ex. print(), len(), type(), min(), max(), avg(), range(), int().

2.User-difined functions.

These are functions that you create yourself to perform specific task.

```
Ex.def add(a,b):

          Return a + b
```

Print(add(1,5))

Userdifined functions are ferther classified into 4 types.

1. No argument no return value.
2. With argument no return value.
3. No argument with return value.
4. With argument with return value.

16.What is defference between userdifined function and lambda function.

Ans. User-difined function –

1. A function define using the def keyword.
2. Uses def keyword with a name, parameter and a block of code.
3. Ex. def add(a, b) : return a + b
4. Used for complex or reusable code.

Lambda function –

1. A function defined using the lambda keyword.
2. Uses lambda keywords with parameter and a single expression.
3. Ex. add = lambda a,b: a + b
4. Used for short, simple, temporary operation.

17.what is anonymous.

Ans. In Python, an "anonymous function" is a small, unnamed function defined using the lambda keyword. It's also commonly called a lambda function.

- **No Name:**

Unlike regular functions defined with def, lambda functions don't have a specific name.

- **Single Expression:**

They are designed for short, one-line operations that evaluate a single expression and automatically return its result.

- **Concise Syntax:**

They use the lambda keyword followed by arguments, a colon, and then the expression.

# Regular function

def add_two(x):

```
    return x + 2

print(add_two(5)) # Output: 7

# Anonymous (lambda) function doing the same thing
add_two_lambda = lambda x: x + 2

print(add_two_lambda(5)) # Output: 7
```

18.What is recursion.

Ans. Recursion in Python, and in programming generally, is a technique where a function calls itself to solve a problem.

```
def factorial(n):
    # Base Case: Factorial of 0 or 1 is 1
    if n == 0 or n == 1:
        return 1
    # Recursive Case: n! = n * (n-1)!
    else:
        return n * factorial(n - 1)
print(factorial(5))
```

19.Explain the drawback of recursion.

Ans.
1. High Memory Usage

Each recursive call adds a new layer to the call stack.

For very deep recursion, this can consume a lot of memory and may crash the program.

2. Risk of Stack Overflow

If there is no proper base case or recursion is too deep, the program can run into a stack overflow error.

Example: Calling factorial(-1) without checking base case.

3. Slower Performance

Recursion involves function call overhead for each call.

Iterative solutions (loops) are often faster and more efficient for simple tasks.

4. Harder to Debug

Tracing the flow of multiple recursive calls can be confusing, especially for beginners.

5. Not Always Suitable for Large Inputs

Problems with very large input sizes may cause recursion to fail or be inefficient.

Example: Recursive Fibonacci function without memoization is very slow for large n.

20.what is memoization.

Ans. In Python, memoization is an optimization technique where you store the results of "expensive" function calls (functions that take a long time to compute) and reuse those stored results when the same inputs occur again.

```
from functools import lru_cache

@lru_cache(maxsize=None) # maxsize=None means unlimited cache size

def fibonacci(n):

   if n <= 1:

      return n

   else:

      return fibonacci(n - 1) + fibonacci(n - 2)

print(fibonacci(10))
```

21.Defference between filter and map.

Ans. map(): Transformation

- map() applies a given function to every item in an iterable (like a list or tuple) and returns a new iterable containing the results of that function for each item.
- It's about transforming each element into something new.
- Ex. numbers = [1, 2, 3, 4]

  squared_numbers = list(map(lambda x: x * x, numbers))

  # squared_numbers will be [1, 4, 9, 16]

filter(): Selection

- filter() applies a given function (which must return a boolean value – True or False) to every item in an iterable. It then returns a new iterable containing only those items for which the function returned True.
- It's about selecting or filtering out elements based on a condition.
- numbers = [1, 2, 3, 4, 5, 6]

  even_numbers = list(filter(lambda x: x % 2 == 0, numbers))

  # even_numbers will be [2, 4, 6]

23.What is object oriented programing.

Ans. Object-Oriented Programming (OOP) in Python is a way of organizing your code around "objects" rather than just functions and logic.

24. what are the features of object oriented programing.

Ans. 1. Class –

A class is a blueprint or template for creating objects.

It defines the data (attributes) and behavior (methods) of an object.

```
Ex.class Car:
  def __init__(self, brand, color):
    self.brand = brand
    self.color = color
```

2. Object

An object is an instance of a class.

It represents a real-world entity with specific data.

Ex.my_car = Car("Toyota", "Red")

3. Encapsulation

It means binding data and methods together in a single unit (class).

It also helps in hiding the internal details from the outside world.

```
Ex.class Student:
    def __init__(self, name, marks):
        self.__marks = marks # private variable

    def show_marks(self):
        print(self.__marks)
```

4. Inheritance

Allows a new class to use properties and methods of an existing class.

Promotes code reusability.

```
Ex.class Animal:
    def speak(self):
        print("Animal speaks")
class Dog(Animal):
    def bark(self):
        print("Dog barks")
dog = Dog()
dog.speak() # Inherited method
```

5. Polymorphism

Means one name — many forms.

The same function or method can behave differently in different classes.

```
Ex.class Cat:
    def sound(self):
        print("Meow")
class Dog:
    def sound(self):
        print("Bark")
for animal in (Cat(), Dog()):
    animal.sound()
```

6. Abstraction

Hides unnecessary details and shows only the essential features.

Achieved using abstract classes or interfaces.

```
from abc import ABC, abstractmethod
Ex.class Shape(ABC):
    @abstractmethod
    def area(self):
        pass
```

25.What is defference between Abstraction and . Encapsulation.

Ans.

Abstraction –

1. Hiding unnecessary details and showing only important features.
2. To simplify complex systems by showing only what is needed.
3. Focuses on what a class does.

Encapsulation. –

1. Wrapping data and methods into a single unit.
2. To protect data from unauthorized access modifiers.
3. Focuses on how data is hidden and controlled.

26.What is access modifiers in python.

Ans. 1. Public -

Members (variables or methods) are accessible everywhere — inside and outside the class.

Default in Python (if you don’t specify anything).

Example:

```
class Student:
    def __init__(self, name):
        self.name = name # public variable
    def display(self): # public method
        print("Name:", self.name)
obj = Student("Roshan")
obj.display()
print(obj.name) # accessible directly
```

2. Protected

Members are meant to be used within the class and its subclasses.

Defined by using a single underscore _ before the name.

It’s only a convention, not strict — can still be accessed from outside.

Example:

```
class Student:
    def __init__(self):
        self._marks = 85 # protected variable
class Derived(Student):
    def show(self):
        print("Marks:", self._marks)
obj = Derived()
obj.show()
print(obj._marks) # can access, but not recommended
```

3. Private

Members are completely hidden from outside access.

Defined by using double underscore __ before the name.

Can only be accessed inside the class.

Example:

```
class Student:
    def __init__(self):
        self.__grade = "A" # private variable
    def show_grade(self):
        print("Grade:", self.__grade)
obj = Student()
obj.show_grade()
# print(obj.__grade)  Error (cannot access directly)
```

If you really need to access it (not recommended), Python internally changes the name:

```
print(obj._Student__grade) # Name mangling.
```

27.What is constructer.

Ans. A constructor in Python is a special function that automatically runs when you create an object of a class.

It is mainly used to initialize (set) the values of variables in that object.

In Python, the constructor method name is __init__().

Example:

```
class Student:
    def __init__(self, name, age):
        self.name = name # set name
        self.age = age # set age
```

```
# creating object

s1 = Student("Roshan", 21)

print(s1.name)

print(s1.age)

 Output:

Roshan

21
```

28.What is Destructer.

Ans. A destructor in Python is a special function that is called automatically when an object is deleted or destroyed.

It is mainly used to clean up resources like closing files or releasing memory before the object is removed from memory.

In Python, the destructor method name is __del__().

Example:

```
class Student:

    def __init__(self, name):

        self.name = name

        print(f"Object created for {self.name}")

    def __del__(self):

        print(f"Object destroyed for {self.name}")

# creating object

s1 = Student("Roshan")

# deleting object

del s1
```

Output:

```
Object created for Roshan
```

Object destroyed for Roshan

29.What are the defference between method and constructer.

Ans.

Aspect	Method	Constructor
Definition	A function defined inside a class that performs a specific task or behavior of the object.	A special method that is automatically called when an object is created to initialize it.
Purpose	To perform operations or actions related to the object.	To initialize the object's attributes when it is created.
Name	Any valid function name (e.g., start, drive)	Always __init__ in Python
Call	Must be explicitly called on an object	Automatically called when the object is created
Parameters	Can take any number of parameters	Usually takes parameters to initialize attributes (self is mandatory)
Return Value	Can return a value or nothing	Does not return any value (None is default)

29. What is difference between class variable & instance variable?

→

Aspect	Class Variable	Instance Variable
Definition	A variable shared by all instances of a class.	A variable unique to each instance of a class.
Declaration	Defined inside the class, but outside any method.	Defined inside a method, usually __init__(), using self.
Access	Can be accessed by class name or object.	Accessed only via object (using self).
Purpose	Used for common/shared data across objects.	Used for data specific to an object.

30. What is difference between local variable & global variable?

→

Aspect	Local Variable	Global Variable
Definition	A variable declared inside a function or block.	A variable declared outside all functions or classes, at the module level.
Scope	Accessible only within the function where it is defined.	Accessible throughout the module, in all functions (if declared global inside functions when modifying).
Lifetime	Exists only while the function is executing.	Exists as long as the program runs.
Declaration	Defined inside a function.	Defined outside functions, usually at the top of the file.
Access/Modification	Can't be accessed outside its function.	Can be accessed anywhere; to modify inside a function, use the global keyword.

31. What is difference between formal parameter & actual parameter ?

→

Aspect	Formal Parameter	Actual Parameter
Definition	The variable defined in the function declaration/definition.	The real value or argument passed to the function when it is called.
Location	Inside the parentheses of the function definition.	Inside the parentheses when calling the function.
Purpose	Acts as a placeholder to receive values.	Provides the actual data for the function to process.
Lifetime	Exists only during function execution.	Exists in the calling environment until function execution completes.

32. What is difference between keyword argument & required argument ?

→

Aspect	Required (Positional) Argument	Keyword Argument
Definition	Arguments that must be passed in the correct order when calling a function.	Arguments that are passed by explicitly specifying the parameter name, regardless of order.
Order	Order matters; must match function definition.	Order does not matter; matched by parameter name.
Mandatory	Must be provided, otherwise Python throws an error.	Optional if defaults are provided; otherwise, still required.
Syntax	func(value1, value2)	func(param1=value1, param2=value2)

33.What is class and object ?

→ 1. Class

- A class is a blueprint or template for creating objects.
- It defines the structure (attributes) and behavior (methods) that the objects created from it will have.

2. Object

- An object is an instance of a class.
- It is a real, tangible entity created based on the class blueprint.

34.What is self parameter ?

→

The self parameter in Python refers to the instance of the class itself.

It is used inside a class to access instance variables and methods.

self represents the current object calling the method.

It is not a keyword; you could name it differently, but using self is the standard convention.