Python interview questions

1)What is python?

Python is a high-level ,interpreted language. It is an easy to use and human understandable language. It is a dynamically typed language. It supports multiple paradigms, including procedural, object-oriented, and functional programming

It is known for its easy syntax and readability. Example: print("Hello, World!")

---

2. What is an interpreter?

An interpreter is software that reads and executes code line-by-line, translating each statement into machine instructions at runtime and An interpreted language executes instructions line by line without compiling them into machine-level code.

---

3. What is the difference between interpreter and compiler?

Interpreter: Executes code line-by-line (e.g., Python), Easier to debug

Compiler: Translates the entire program into machine code at once (e.g., C++)

---

4. What is dynamically typed language?

Yes, Python is dynamically typed, which means you don’t need to declare variable types. Python determines the type at runtime.

Example: a = 10 then a = "text" is valid.

---

5. What is data?

Data is information that can be stored and processed using variables.It can be numbers, text, or more complex structures that programs use for processing.

Example:

```
name = "Reshma"

age = 21
```

---

6. How many types of data types are there in Python?

Basic types: int, float, str, bool, list, tuple, dict, set

Python has several built-in data types:

Numeric (int, float, complex)

Sequence (list, tuple, range)

Text (str)

Set (set, frozenset)

Mapping (dict)

Boolean (bool)

---

7. Examples for data types. Need single example for each type.

int: a = 5

float: b = 3.14

str: c = "Hello"

bool: d = True

list: e = [1, 2, 3]

tuple: f = (1, 2, 3)

dict: g = {"name": "Alice"}

set: h = {1, 2, 3}

---

8. What is list in Python? Give an example for a list .

A list is a collection which is ordered and mutable.

Example:

my_list = [10, 20, 30]

fruits = ["apple", "banana", "cherry"]

---

9. What is dict in Python? Give an example for dict.

A dictionary stores data in key-value pairs. A dictionary is an unordered, mutable collection of key-value pairs.

Example:

student = {"name": "Reshma", "age": 21}

Example: my_dict = {"name": "John", "age": 25}

---

10. What is tuple in Python? Give an example for Tuple.

A tuple is an ordered, immutable collection of elements.

Example:

point = (10, 20)

my_tuple = (1, 2, 3)

11. Difference between tuple vs list vs dict?

List: mutable, ordered -> [1, 2], Duplicates allowed

Tuple: immutable, ordered -> (1, 2), Duplicates allowed

Dict: mutable, unordered key-value -> {"a": 1}, keys are unique

---

12. What is variable in Python? Give examples and difference varible in python.

A variable is used to store data in memory is a name that refers to a value stored in memory. It doesn't require type declaration.

- It holds data like numbers, strings, or objects.
- You don't need to declare the type.
- Python assigns the type automatically (because it's dynamically typed).

Example:

x = 100

name = "Python"

x = 5

name = "Reshu"

13. What is mutable & immutable? What are mutable and immutable data types in Python?

Mutable: Can be changed after creation (value can be modified). Example: You can add, remove, or update items.

- list,
- dict, set

Immutable: Cannot be changed once created. If you try to change it, Python creates a new object instead.

- int,
- float,
- str, tuple

lst = [1, 2]; lst[0] = 100 # mutable

t = (1, 2); # t[0] = 100 ❌ # immutable

14. How is it possible to mutate the list? Give an example to demonstrate state.

you can change elements, append, or remove items.

Example-1:

colors = ["red", "blue"]

colors[0] = "green" # mutated

Example-2:

my_list = [1, 2, 3]; my_list[0] = 100

15. Can we increase the size of list? If yes, give an example for that.

Yes,

1.lists can grow using append() or extend().

Example:

nums = [1, 2]

nums.append(3) # Output:[1, 2, 3]

2. Using insert(index, value) – Adds item at a specific position:

 fruits.insert(1, "orange")

print(fruits) # Output: ['apple', 'orange', 'banana', 'cherry']

3. Using extend() – Adds multiple items at once:

 fruits.extend(["grape", "melon"])

print(fruits) # Output: ['apple', 'orange', 'banana', 'cherry', 'grape', 'melon']

---

16. What is operator? Types in Python:

Operators are symbols used to perform operations. Types:

Arithmetic: +, -

Assignment: =, +=

Comparison: ==, !=

Logical: and, or, not

Bitwise, Identity, Membership

---

17)What are logical operators? Give examples for each and every operator in it?

Logical operators are used to combine conditional (boolean) expressions and return True or False based on the logic.

Python has 3 logical operators: Logical AND, Logical OR, Logical NOT.

Examples

1.and Operator: x = 5 print(x > 2 and x < 10) # True and True → Output: True print(x > 10 and x < 20) # False and True → Output: False

 2. or Operator: x = 5 print(x < 2 or x < 10) # False or True → Output: True print(x > 10 or x < 2) #False or False → Output: False

3. not Operator: x = True print(not x) # Output: False y = 5 print(not (y > 2)) # not True → Output: False

18.What are arithmetical operators? Types of operators in it and examples?

Used to perform math operations.

Operator	**Meaning**	**Example**	**Output**
+	Addition	5 + 2	7
-	Subtraction	5 - 2	3
*	Multiplication	5 * 2	10
/	Division	5 / 2	2.5
//	Floor Division	5 // 2	2
%	Modulus	5 % 2	1
**	Exponentiation	2 ** 3	8

19.What are compositional operators and examples for each of them?

Used to compare values (returns True or False).

Operator	Meaning	Example	Output
==	Equal to	5 == 5	True
!=	Not equal to	5 != 3	True
>	Greater than	5 > 3	True
<	Less than	3 < 5	True
>=	Greater or equal	5 >= 5	True
<=	Less or equal	4 <= 5	True

---

20)what is the difference between the += and -=?

Both += and -= are assignment operators used to update the value of a variable by adding or subtracting another value.

+= (Addition Assignment):

- Adds a value to the existing variable.
- Equivalent to: x = x + value

Example: x = 5 x += 3 # same as x = x + 3

print(x) # Output: 8

-= (Subtraction Assignment):

● Subtracts a value from the existing variable.

● Equivalent to: x = x – value

Example: x = 10 x -= 4 # same as x = x - 4

print(x) # Output: 6

---

21)What is the difference between logical and logical or?

Both and and or are logical operators used to combine two or more conditions and return a Boolean value (True or False), but they work differently.

Logical and:

● Returns True only if both conditions are True.

● If any one is False, the result is False.

Syntax: condition1 and condition2

Example:

a = 5

print(a > 2 and a < 10) # True and True → True

print(a > 2 and a > 10) # True and False → False

Logical or:

● Returns True if at least one condition is True.

● Returns False only if both conditions are False.

Syntax: condition1 or condition2

Example:

a = 5 print(a < 2 or a < 10) # False or True → True

print(a > 10 or a < 2) # False or False → False

22.what is difference between in and not in operator?

in: Checks if a value exists in a sequence.

not in: Checks if a value **does not exist** in a sequence.

Example:

"p" in "python"   # True

"z" not in "python" # True

23.what is memory pooling in python?

A) Memory pooling is an optimization where Python **reuses memory** for small objects (like integers from -5 to 256) instead of creating new ones, to improve performance.

24.what is range for int in python?

A) Python integers have **unlimited precision** — they can grow as large as the available memory allows.

Int Value Range in Python:

In Python, integers can be any size, limited only by your computer's memory. There is no maximum or minimum limit.

No limits like other languages: Python's integers can grow as big or small as needed.

25.what is the difference between not & not in operators?

A) not: Logical operator that negates a condition.

not in: Membership operator that checks if a value **does not exist** in a sequence.

Example:

not True   # False

"z" not in "abc" # True

26.what is difference between // and / in python?

A)/: Regular division → returns **float**

//: Floor division → returns **integer value**, discards decimal

Example:

5 / 2 → 2.5

5 // 2 → 2

27.what is difference between % & / in python?

/: Division (returns quotient)

%: Modulus (returns remainder)

Example:

```
7 / 2  → 3.5
7 % 2  → 1
```

28.what are conditional statements in python give an example for that?

A) They allow decision-making in code using if, elif, and else.

Example:

```
x = 10
if x > 0:
    print("Positive")
else:
    print("Non-positive")
```

29.what is the difference between else and elif in python?

A) elif: "Else if" — checks additional condition if previous if was false.

else: Runs when all conditions above are false.

Examples:

1. Else:

```
x = 5
if x > 10:
    print("x is greater than 10")
```

```
else:
    print("x is less than or equal to 10")
```

2. Elif:

```
x = 5
if x > 10:
    print("x is greater than 10")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is less than 5")
```

In simple words:

1. Else: Executes code when all conditions are false.

2. Elif: Checks another condition if the initial one is false, allowing for multiple conditional checks.

Think of it like a flowchart:

- If: Check the first condition.
- Elif: If the first condition is false, check another condition.
- Else: If all conditions are false, execute this code.

30.what is if-else ladder write an example for that?

A) Multiple if-elif-else conditions stacked one after another.

Example:

```
x = 10
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

31.what is loop ? No of loops in python?

A) A loop repeatedly executes a block of code.

Types in Python:

- for loop
- while loop

32.what is for loop ?write syntax and give an example for that in python?

A) Used to iterate over a sequence like list, string, etc.

Syntax:

```
for variable in sequence:
   # code block
```

Example:

```
for i in range(3):
 print(i)
```

33.what is reverse tracking give an example for that using for loop?

A) Reverse iteration over a sequence.

Example:

```
for i in range(5, 0, -1):
print(i)
```

34.what is difference between while and for loop?

A)Usage Used with known sequences Used when condition is unknown

Control Iterates over items Runs till condition is true

35.reverse a string using for loop?

A) s = "python"

```
rev = ""
for i in range(len(s)-1, -1, -1):
rev += s[i]
```

```
    print(rev)  # Output: nohtyp
```

36.what is range in python and why we use it and where do we use it?

A) range() generates a sequence of numbers. Commonly used in loops.

Example:

range(5) → 0, 1, 2, 3, 4

37.what is str and how it is different from single character?

A) str is a string datatype, which can hold one or more characters.

Difference:

- "a" is a string of one character.
- "apple" is a string of five characters.
  Both are of type str.

38.what is len() method and how can we find length of a integer with len()?

A) len() returns the number of items in a sequence.

To find length of an integer:

```
x = 12345
length = len(str(x))  # Output: 5
```

39.find the length of str without using len()?

```
A) s = "python"
   count = 0
   for i in s:
   count += 1
   print(count)
```

40)What is function in python give an example?

A) A function is a block of reusable code.

```
Example:
def greet(name):
return "Hello, " + name
```

41.write nested functions and access global variable inside the deep most local scope?

A) x = "global"

```
def outer():
def inner():
    print("Accessing:", x)  # accesses global
      inner()
     outer()
```

42. What are *args vs **kwargs? Example:

*args: Accepts variable number of positional arguments

**kwargs: Accepts variable number of keyword arguments

```
Example:
def func(*args, **kwargs):
print(args)
print(kwargs)
func(1, 2, a=10, b=20)
```

43.what is difference between global and local scope?

A) In Python, scope refers to the visibility of a variable.

- Local scope means the variable is declared inside a function and can be accessed only within that function.

- Global scope means the variable is declared outside all functions, and it can be accessed throughout the program, including inside functions (if not shadowed).

Example:

```
x = "global"  # Global scope
def func():
y = "local"  # Local scope
print(x, y)
```

44.what is difference between nonlocal and global keyword and usecase of them?

A) The **global** keyword is used to **modify a global variable** inside a function.

The **nonlocal** keyword is used to **modify a variable from the nearest enclosing (non-global) scope**, typically in nested functions.

Use Case Example:

```
x = "global"
def outer():
y = "outer"
def inner():
nonlocal y
 y = "modified"
    inner()
    print(y)  # Output: modified


def change_global():
    global x
    x = "changed"
```

45.How can I access local scoped variable outside of that local scope give an example?

A) Normally, **local variables are not accessible outside their function**. But we can **return** them and access the value outside.

Example:

```
def my_func():
    local_var = 10
    return local_var


x = my_func()
print(x)  # Output: 10
```

46.what is LEGB rule in python?

A) LEGB is the rule Python follows to resolve variable names in nested scopes. It stands for:

- **L**: Local — inside the current function
- **E**: Enclosing — inside outer (enclosing) functions
- **G**: Global — top-level script or module
- **B**: Built-in — Python's built-in names like len, sum, etc.

Python looks for variables in this order: L → E → G → B.

Example:

```
x = "global"


def outer():
    x = "enclosing"
    def inner():
        x = "local"
        print(x)  # Output: local
    inner()
```

47.what is operators precedence write an example for that?

A) Operator precedence determines the order in which operations are performed in an expression.

For example:

```
result = 10 + 2 * 3
# Output: 16, not 36
```

Here, * has higher precedence than +, so multiplication happens first.

Precedence order example (high to low):

1. () — Parentheses
2. ** — Exponent
3. *, /, //, %
4. +, -
5. Comparison: ==, !=, >, <, etc.
6. Logical: not, and, or