

# INTERVIEW QUESTIONS ON CORE PYTHON

## **1.What is python?**

Python is high-level language – it is similar to English (human understandable language). It is interpreter language (line by line).

## **2.What is interpreter?**

- An interpreter reads your code line by line and runs it right away.
- If there is a mistake, it stops at that line.
- Example: Python uses an interpreter.

## **3.What is difference between interpreter vs Compiler?**

### **Interpreter:**

- An interpreter reads your code line by line and runs it right away.
- If there is a mistake, it stops at that line.
- Example: Python uses an interpreter.

### **Compiler:**

- A compiler reads the whole code at once, checks it, and turns it into a separate file (called machine code).
- Then you can run that file without needing the original code.
- Example: C and C++ use compilers.

## **4.What is dynamically typed language? Is that python is dynamically typed language?**

A dynamically typed language is a programming language in which the type of a variable is determined at runtime, not in advance during code writing (i.e., not at compile-time). This means:

- You don't need to declare the type of a variable when you create it.
- The type can change as the program runs.

## **5.What is data?**

- Data is collection of information or Set of information.
- Data is any raw fact or figure that can be recorded and used for processing.

## **6.How many types of datatypes are there in python?**

There are two types of datatypes in python.

1.Primitive Datatypes

2.Non-primitive Datatypes

### **1.Primitive Datatypes:-**

- Integers
- String
- Float
- Boolean

### **2.Non-Primitive Datatypes:-**

- List
- Dictionary
- Tuple

## **7.Examples for datatypes need single example for every type?**

### **Primitive Datatype:-**

The datatype which can't be changed.

There some of primitive datatypes. They are:-

## 1)Integer(int):-

```
x=10
Print(x)
Print(type(x))
```

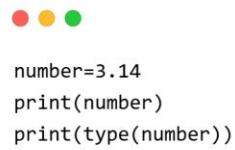
## 2)String(str):-

```
a="Akshitha"
print(a)
print(type(a))
```

## 3)Boolean(bool):-The output in the form of true or false.

```
is_student=True
print(is_student)
print(type(is_student))
```

## 4)Float:- It stands for “floating-point number”.

A white rectangular box with a subtle drop shadow, containing Python code. At the top left of the box are three small colored circles: red, yellow, and green. The code is as follows:

```
number=3.14
print(number)
print(type(number))
```

## 2.Non-primitive datatypes:-

- List
- Dictionary
- Tuple

**1)List:-** List is a collection of multiple values or multiple datatypes.

- List contains heterogenous values and homogenous values.
- Homogenous values:-Same datatypes or values.
- Hetrogenous values:- Different datatypes or values.
- Ex:-

A white rectangular box with a subtle drop shadow, containing Python code. At the top left of the box are three small colored circles: red, yellow, and green. The code is as follows:

```
Fruits=["apple","banana","cherry"]
print(Fruits)
```

**2)Dictinoary:-** Each item is a pair.

- Key:value
- Key must be unique.
- Values can be any type.

- Ex:-

A screenshot of a code editor with a pink-to-blue gradient background. In the center is a white rectangular box with three colored dots (red, yellow, green) at the top left. Inside the box, the following Python code is written:

```
student={"name":"Akhi",  
        "age":15,  
        "course":"python"  
}  
print(student)
```

In the bottom left corner of the gradient background, the text "johndoe" is visible.

**3)Tuple:-** It can store multiple items in single variable.

- Ex:-

A screenshot of a code editor with a pink-to-blue gradient background. In the center is a white rectangular box with three colored dots (red, yellow, green) at the top left. Inside the box, the following Python code is written:

```
tuple=(10,20)  
print(tuple)
```

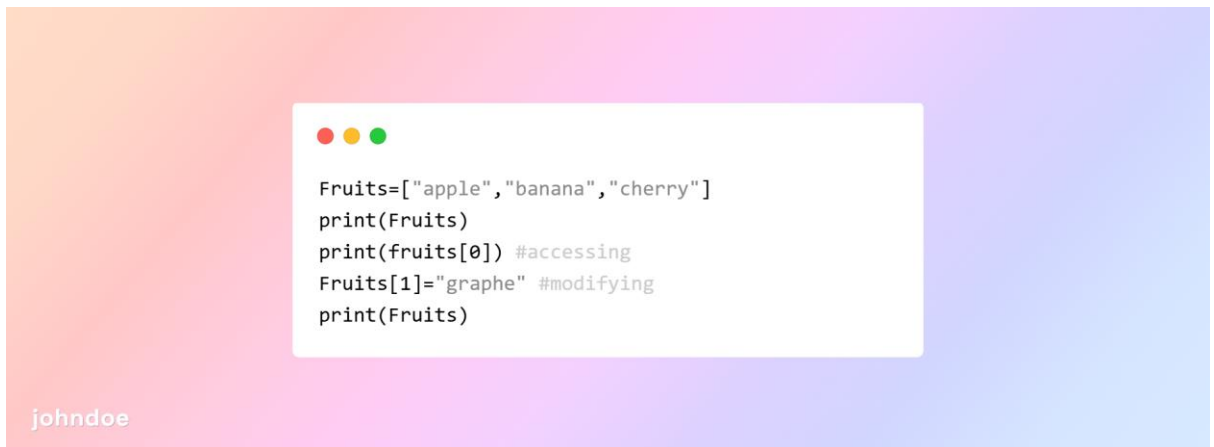
In the bottom left corner of the gradient background, the text "johndoe" is visible.

## 8.What is list in python? Give an example for list?

**List:-** List is a collection of multiple values or multiple datatypes.

- List contains heterogenous values and homogenous values.
- Homogenous values:-Same datatypes or values.
- Hetrogenous values:- Different datatypes or values.

- Ex:-

A screenshot of a code editor with a pink-to-blue gradient background. In the center, there is a white rectangular box with rounded corners containing Python code. The code defines a list 'Fruits' with elements 'apple', 'banana', and 'cherry'. It then prints the list, accesses the first element (index 0), modifies the second element (index 1) to 'grape', and prints the list again. The username 'johndoe' is visible in the bottom-left corner of the editor.

```
Fruits=["apple","banana","cherry"]  
print(Fruits)  
print(Fruits[0]) #accessing  
Fruits[1]="grape" #modifying  
print(Fruits)
```

johndoe

## 9.What is dictionary in python? Give an example for dictionary?

- Each item is a pair.
- Key:value
- Key must be unique.
- Values can be any type.
- Dictionary is mutable.
- Ex:-

A screenshot of a code editor with a pink-to-blue gradient background. In the center, there is a white rectangular box with rounded corners containing Python code. The code defines a dictionary 'student' with keys 'name', 'age', and 'course'. It then prints the value for the 'name' key and the value for the 'age' key using the 'get' method. The username 'johndoe' is visible in the bottom-left corner of the editor.

```
student={"name":"Akhi"  
        "age"= 15  
        "course" = "python"  
        }  
print(student["name"])  
print(student.get("age"))
```

johndoe

## 10.What is tuple in python? Give an example for tuple?

- Tuple is a non-primitive datatype.
- Tuple can be store multiple datatypes in single variable.

- Tuple can allow duplicates.
- Ex:-



```
fruits=('apple','mango','grape')
print(fruits)
print(fruits[0])
print(fruits[-1])
```

johndoe

## 11.What is difference between tuple vs list vs dictionary?

Tuple	List	Dictionary
1.Tuple is ordered Collection.	1.List is ordered Collection.	1.Dictionary is unordered collection of key-value pairs.
2.Represented by “ ()”.	2.Represented by “ [ ]”.	2.Represented by “ {key: value} ”.
3.Tuple is Immutable.	3.List is mutable.	3.Dictionary is mutable.
4.Indexing can be done by position.	4.Indexing can be done by position.	4.Dictionary uses keys.
5.Duplicate values allowed.	5.Duplicate values allowed.	5.Keys are unique.

## 12.What is Variable in python? Give examples for different variable in python?

- A variable in python is a name that is used to store data.
- It acts as a container to hold values.

### Rules:-

- Must start with a letter(A-Z or a-z).
- Cannot start with a number.
- Cannot use keywords(like if, for, else).

### Examples of different variables in python:-

## 1.Integer variable :- age=25

## 2.String variable:- name="akhi"

### 3.Float variable:- pi=3.14

#### 4.Boolean Variable:- is\_student=True

### 5. List variable:-fruits= ("apple", "banana")

### 6. Tuple variable:- tuple=(10,100)

**7.Dictionary variable:-** student{"name": "ruchi",  
"age" : 16  
}

### 13.What is mutable and immutable? What are mutable datatypes and immutable datatypes in python?

**Mutable:-** Can be changed after creation (i.e., you can modify, add, or remove elements).

**Immutable:-** Cannot be changed after creation. Any operation that modifies it creates a new object.

### Mutable Datatypes:-

- List
- Dict(dictionary)
- Set



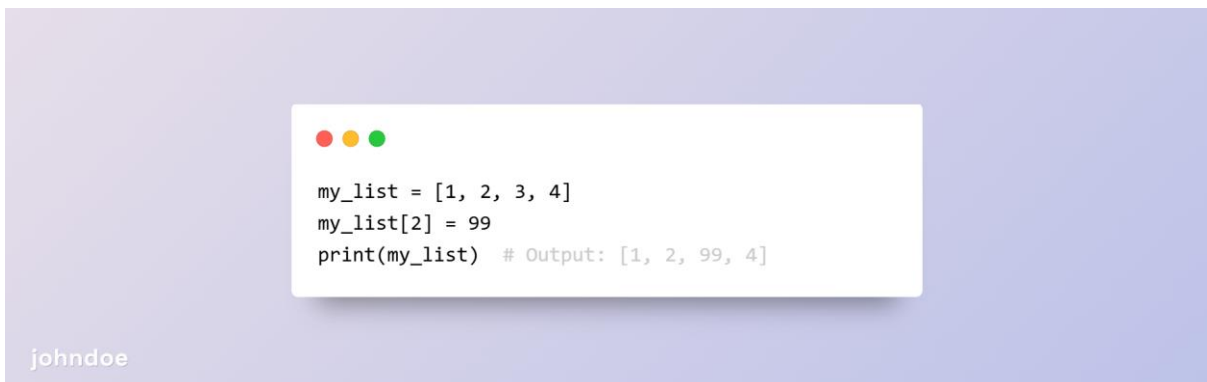
## Immutable Datatypes:-

- float
- bool
- str (string)
- tuple

### 14. Is possible to mutate the list? If yes, give an example to demonstrate that?

Yes, it is possible to mutate a list in Python. Lists are mutable, which means their content can be changed after creation (unlike strings or tuples).

Example:-



```
my_list = [1, 2, 3, 4]
my_list[2] = 99
print(my_list) # Output: [1, 2, 99, 4]
```

johndoe

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

- Yes, you can increase the size of a list in Python.
- Lists are dynamic, which means you can add (append, extend, insert) elements at any time, increasing their size.



```
my_list = [1, 2, 3]
my_list.append(4)
print(my_list) # Output: [1, 2, 3, 4]
```

johndoe



```
my_list = [1, 2, 3]
my_list.extend([4, 5])
print(my_list) # Output: [1, 2, 3, 4, 5]
```

johndoe



```
my_list = [1, 2, 3]
my_list.insert(1, 100)
print(my_list) # Output: [1, 100, 2, 3]
```

johndoe

## 16.What is operator? Types of operators in python?

An operator in Python is a special symbol or keyword used to perform operations on variables and values (also called operands).

**Types of operators:-**

- 1)Arithmetic Operator
- 2)Assignment Operator
- 3)Comparison Operator
- 4)Logical Operator

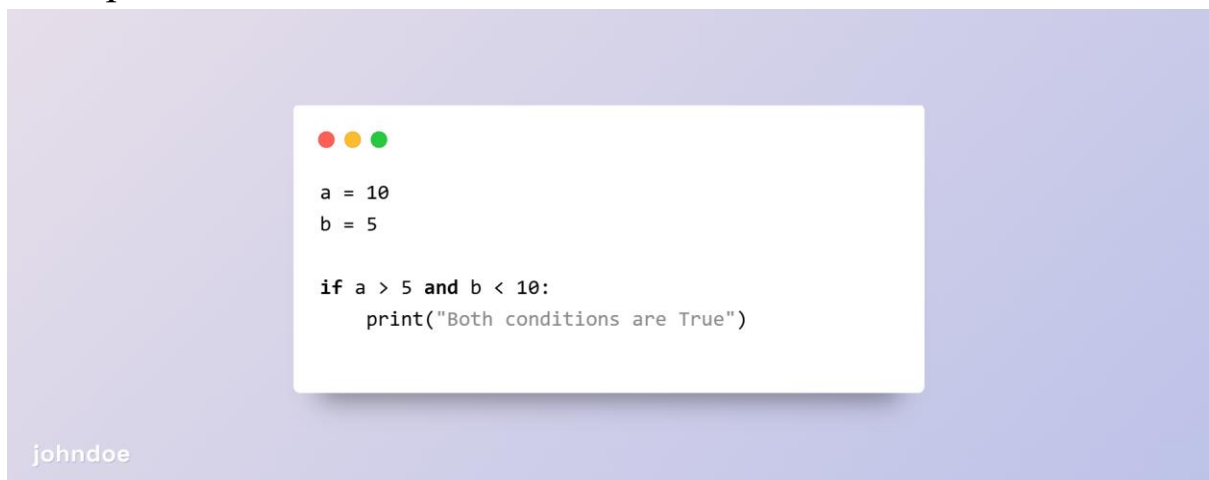
## **17.What are logical operators? Give examples for each and every operator in it?**

Logical operators are used to combine multiple conditional expressions (i.e., expressions that return True or False). They help control program logic, especially in if, while, and other control structures.

### **Types of Logical operator:-**

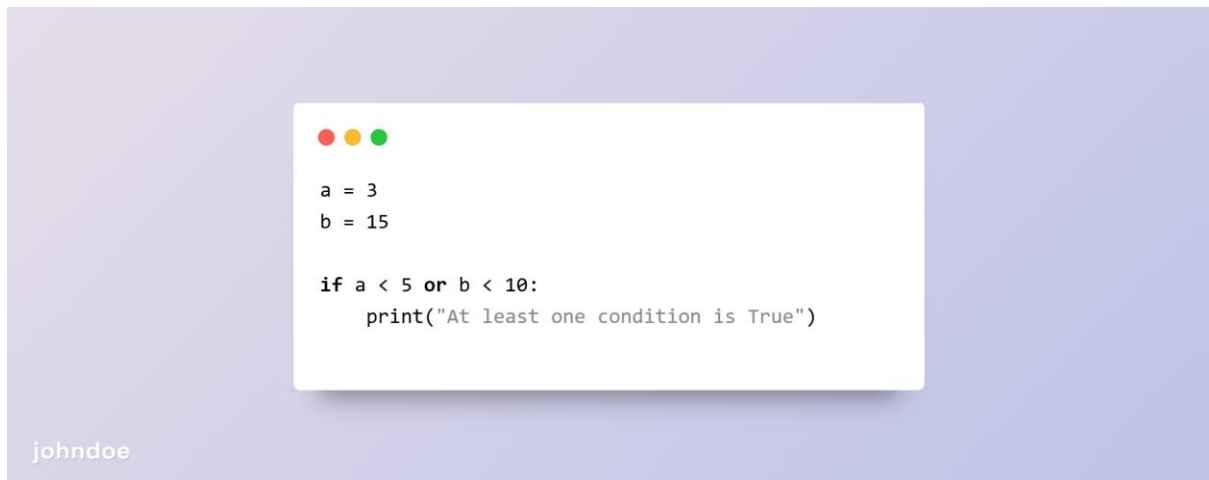
**1)And:-** Returns True if both conditions are true.

Example:-

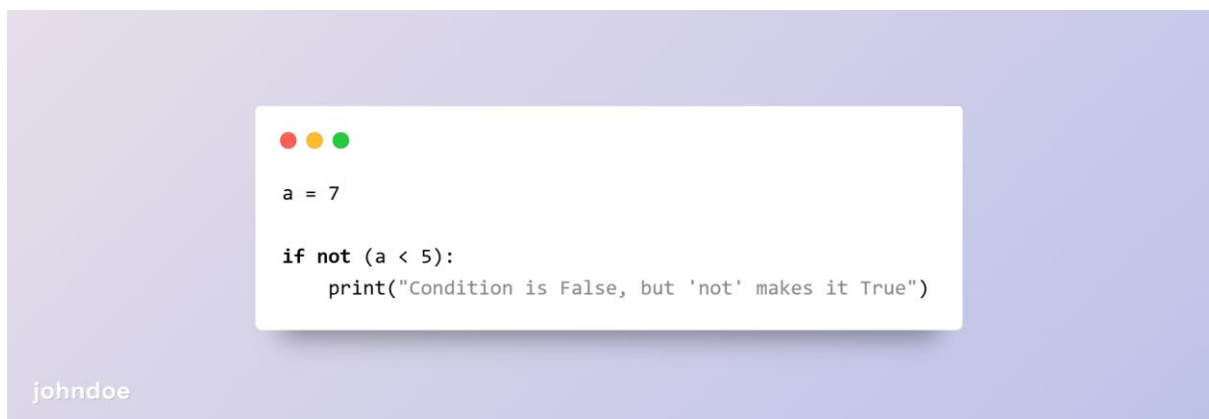


**2)Or:-** Returns True if at least one condition is true.

Example:-



**3)Not:-** Reverses the result (i.e., True becomes False and vice versa).



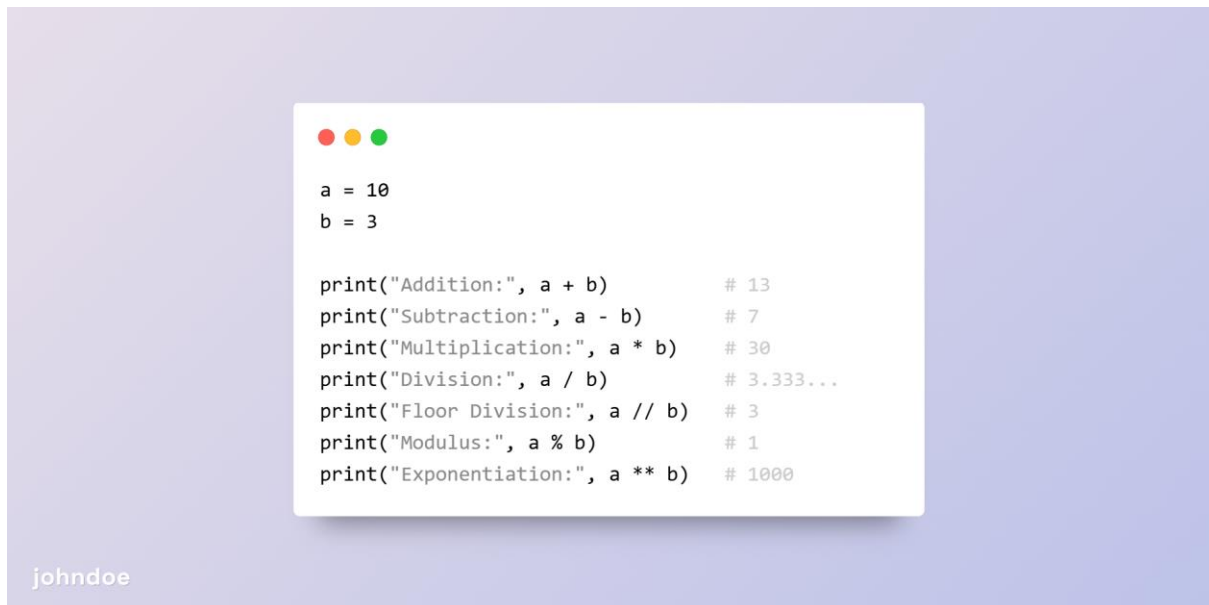
## 18.What are Arithmetical Operators? Types of operators and example?

Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication, division, etc., between numeric values (integers, floats, etc.).

### Types of operators:-

- +
- -
- \*
- /
- //
- %

○ \*\*



```
a = 10
b = 3

print("Addition:", a + b)      # 13
print("Subtraction:", a - b)   # 7
print("Multiplication:", a * b) # 30
print("Division:", a / b)      # 3.333...
print("Floor Division:", a // b) # 3
print("Modulus:", a % b)       # 1
print("Exponentiation:", a ** b) # 1000
```

johndoe

## 19.What are Comparisonal Operators and examples for each of them?

Comparisonal Operators (often called comparison operators) are used in programming to compare two values. The result of a comparison is always a Boolean value: either True or False.

### Types of comparisonal operators:-

1. Equal to (==)
2. Not equal to (!=)
3. Greater than (>)
4. Greater than or equal to (>=)
5. Less than or equal to (<=)

## 20. What is the difference between the += and -=?

The += and -= operators are compound assignment operators in programming. They are used to update the value of a variable by adding to or subtracting from it.

### 1) += (Addition Assignment):-

This operator adds the right-hand value to the left-hand variable and assigns the result back to that variable.

## 2) -= (Subtraction Assignment):-

This operator subtracts the right-hand value from the left-hand variable and assigns the result back to that variable.

## 21.What is the difference between the and-or operator?

The and and or operators are logical operators used to combine multiple Boolean expressions (conditions). They determine whether a combined condition is True or False.

### 1)and Operator:-

- ☐ Returns True only if both conditions are True.
- ☐ If any one of the conditions is False, the result is False.

### 2) or Operator:-

- ☐ Returns **True** if **at least one** of the conditions is True.
- ☐ Only returns False if **both** conditions are False.

## 22.What is the difference between the in and not in operator?

### in Operator

- Returns **True** if a value **exists** in the given sequence.
- Otherwise, returns **False**.

### not in Operator

- Returns **True** if a value **does NOT exist** in the sequence.
- Returns **False** if the value **is present**.

### **23.What is memory pooling in python?**

Memory pooling in Python refers to the internal memory management mechanism used to reuse memory blocks for small objects instead of frequently allocating and deallocating memory from the operating system. This helps improve performance and reduce memory fragmentation.

### **24.What is value range for int in python?**

In Python 3, the int type supports arbitrary precision, meaning:  
There is no fixed maximum or minimum value for an integer.

### **25.What is the difference between the not and not in operator?**

#### **not Operator**

- It is a logical negation operator.
- It inverts the Boolean value of an expression.
- not True becomes False, and not False becomes True.

#### **not in Operator**

- It is a **membership operator**.
- Checks if a value is **NOT present** in a sequence (like a list, string, tuple, etc.).
- Returns True if the value is **absent**.

### **26.What is difference between the // & / in python?**

**/ — True Division (Floating-point Division)**

- Always returns a **float** (decimal value), even if the result is a whole number.
- Works with both integers and floats.

### // — **Floor Division (Integer Division)**

- Returns the **floor value** of the division (i.e., rounds down to the nearest whole number).
- If both operands are integers, result is an integer.
- If at least one operand is float, result is a float (but still rounded down).

## 27.What is the difference between the % & / in python?

### / — **True Division**

- Divides the left number by the right and returns the **quotient** as a **float**.

### % — **Modulus (Remainder)**

- Returns the **remainder** after division.
- Useful for checking divisibility, cycling through values, or working with even/odd checks.

## 28.What are conditional statements in python?

Conditional statements in Python are used to make decisions in a program based on whether a certain condition is True or False. These statements control the flow of execution depending on the evaluation of logical expressions.

### **Types of conditional statements :-**

- if statement

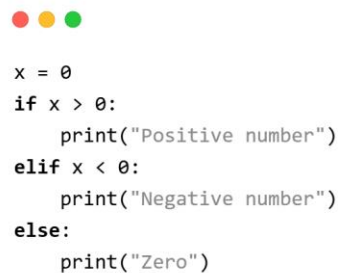


- if-else statement
- if-elif-else statement

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

### elif-statement

Checks multiple conditions. The first True condition will be executed.

A code editor window with a light blue background and a white code area. The code is as follows:

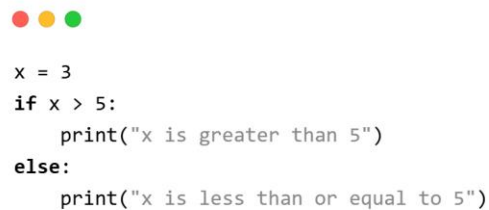
```
x = 0
if x > 0:
    print("Positive number")
elif x < 0:
    print("Negative number")
else:
    print("Zero")
```

The code is written in a monospaced font. The editor has three colored dots (red, yellow, green) in the top left corner. The username 'johndoe' is visible in the bottom left corner of the editor area.

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

### else statement

Executes one block of code if the condition is True, otherwise executes another block.

A code editor window with a light blue background and a white code area. The code is as follows:

```
x = 3
if x > 5:
    print("x is greater than 5")
else:
    print("x is less than or equal to 5")
```

The code is written in a monospaced font. The editor has three colored dots (red, yellow, green) in the top left corner. The username 'johndoe' is visible in the bottom left corner of the editor area.

```
x = 3
if x > 5:
    print("x is greater than 5")
else:
    print("x is less than or equal to 5")
```

## 30. What is if-else ladder? Write an example for that?

- An if-else ladder is a chain of multiple if, elif, and else statements used to test multiple conditions one after another.
- It checks each condition from top to bottom and executes the block of the first True condition.
- If none of the if or elif conditions are True, then the else block is executed.

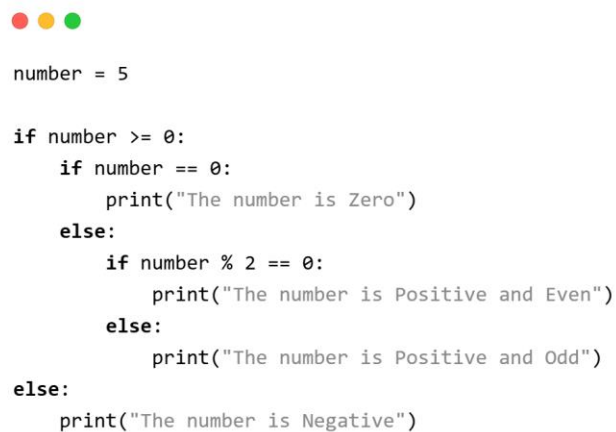


```
marks = 75

if marks >= 90:
    print("Grade: A")
elif marks >= 80:
    print("Grade: B")
elif marks >= 70:
    print("Grade: C")
elif marks >= 60:
    print("Grade: D")
else:
    print("Grade: F")
```

johndoe

**31. Write nested if-else condition example?**



```
number = 5

if number >= 0:
    if number == 0:
        print("The number is Zero")
    else:
        if number % 2 == 0:
            print("The number is Positive and Even")
        else:
            print("The number is Positive and Odd")
else:
    print("The number is Negative")
```

johndoe

### **32.What is loop? Number of loops in python? Write syntax? Write an example for for-loop?**

**Loop:-** A loop is used in Python to repeat a block of code multiple times. It helps reduce code repetition when the same set of instructions needs to run multiple times.

#### **Types of loops:-**

There are mainly 2 types of loops in Python:

1. for loop
2. while loop

#### **Syntax of for loop in Python:-**

for variable in sequence:

    # Block of code to repeat

#### **Example of for loop:**

for i in range(1, 6):

```
print(i)
```

### 33.What is reverse tracking? Give an example for that using for loop?

- Reverse tracking in a loop means iterating through a sequence in reverse order — from end to start instead of from start to end.
- In Python, this is usually done using the range() function with a negative step in a for loop.

#### **Syntax of Reverse Tracking using for loop:-**

```
for variable in range(start, end, step):
```

```
    # Block of code
```

#### **Example: Print numbers from 5 to 1 (Reverse Order)**

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


```
    print(i)
```

### 34.What is difference while & for loop?

	<b>for Loop</b>	<b>while Loop</b>
<b>Usage</b>	Used when the number of iterations is known or we are iterating over a sequence (list, range, etc.).	Used when the number of iterations is unknown and depends on a condition.
<b>Condition Check</b>	Condition is typically defined using a sequence like range().	Condition is explicitly written and checked before each iteration.
<b>Syntax</b>	for variable in sequence:	while condition:

	<b>for Loop</b>	<b>while Loop</b>
<b>Purpose</b>	Best suited for count-based or sequence-based repetition.	Best suited for condition-based repetition.
<b>Example Usage</b>	Iterating over a list, string, or using range().	Running until a user input is valid, or until a condition becomes False.

### 35.Reverse of string using for-loop?



```

name="akshiths"
for char in range(len(name)-1,-1,-1):
    print(name)

```

johndoe

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

#### Range in python:-

The range() function in Python is used to generate a sequence of numbers.

It is commonly used when you want to repeat something a specific number of times, especially in loops.

#### Use of range( ):-

- To control the number of iterations in a for loop.

- To generate a sequence of numbers without creating a separate list.
- To simplify writing loops where we need counting numbers.

### **Where do we use range( )?**

- **For loops:** To iterate a fixed number of times.
- **Indexing:** When accessing elements of lists or strings by position.
- **Reverse loops:** Using negative step values.
- **Generating sequences:** For simple arithmetic sequences.

### **37.What is str and how it is different from a single character?**

In Python, str stands for string, which is a data type used to represent a sequence of characters (letters, numbers, symbols, spaces, etc.).

- A string can contain zero, one, or many characters. Strings are always written inside quotes — either single quotes `' '` or double quotes `" "`.

### **38.What is len( ) method and how can we find length of an integer with len( ) ?**

The len() function in Python is used to find the length (i.e., number of elements) of:

- Strings (number of characters)
- Lists (number of items)
- Tuples
- Dictionaries (number of keys)
- Other sequences

Example for len() with String:-

```
text = "Python"
```

```
print(len(text)) # Output: 6
```

**To find the "length" (number of digits) of an Integer using len() :-**

Convert the integer to a string first.

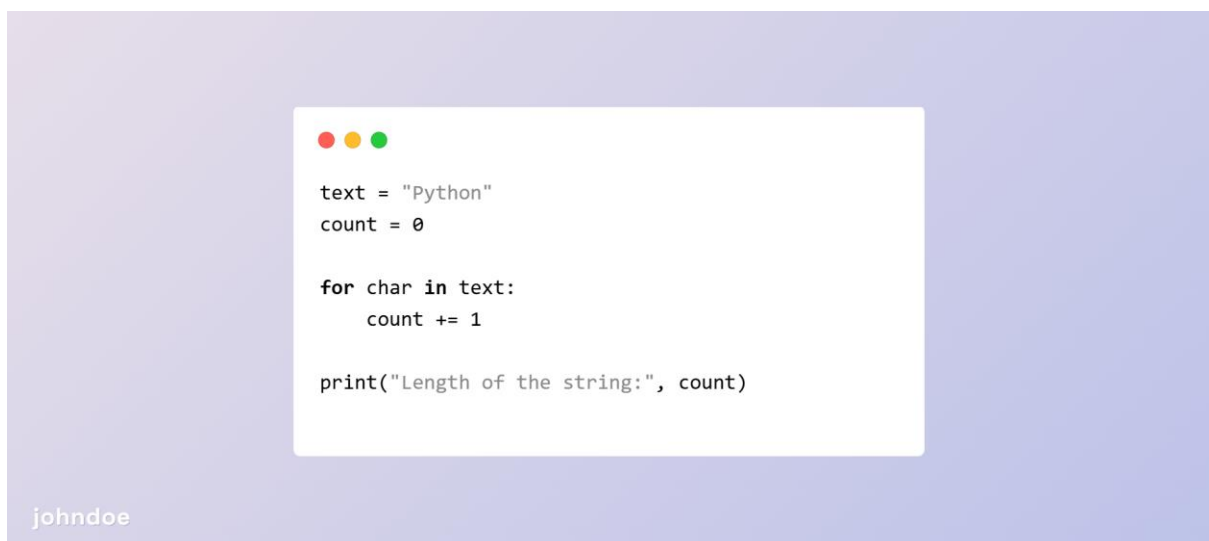
```
num = 12345
```

```
length = len(str(num))
```

```
print(length)
```

Output:-5

**39.Find the length of str without using len( ) method?**



```
text = "Python"
count = 0

for char in text:
    count += 1

print("Length of the string:", count)
```

johndoe

**40.What is function in python? Give an example.**

- A function in Python is a block of reusable code that performs a specific task.
- Functions help in writing **organized, clean, and reusable code**.

- Instead of writing the same code again and again, you can **define it once** in a function and **call it whenever needed**.

Example:-

```
def greet(name):  
    print("Hello,", name)
```

# Calling the function

```
greet("Alice")  
greet("Bob")
```

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

### **Nested Functions in Python**

A nested function is simply a function defined inside another function. Nested functions are often used to logically organize code and manage scope.

**Example:-**





## 42.What is arguments vs parameters? An example?

Term	Explanation	Location
<b>Parameter</b>	A variable in the function definition that accepts a value.	Appears when defining the function.
<b>Argument</b>	The actual value passed to the function when it is called.	Appears when calling the function.

## 43.What is difference between the default arguments and positional arguments and keyword arguments?

### Positional Arguments

- Values are passed in order, matching the function's parameters positionally.

- The first argument goes to the first parameter, the second to the second, and so on.

### Example:-

```
def greet(name, age):
    print(f"Hello {name}, you are {age} years old.")
greet("Alice", 25) # Positional: "Alice" -> name, 25 -> age
```

### Default Arguments

- A **default value** is provided for a parameter.
- If the caller **doesn't provide a value**, the default is used.
- If the caller **provides a value**, it overrides the default.

### Example:-

```
def greet(name, age=18):
    print(f"Hello {name}, you are {age} years old.")
greet("Bob")      # Uses default age = 18
greet("Alice", 25) # Overrides default
```

### Keyword Arguments

- **Name the parameters explicitly** during the function call.
- Order does **not matter** when using keyword arguments.

### Example:

```
def greet(name, age):
    print(f"Hello {name}, you are {age} years old.")

greet(age=30, name="Charlie") # Keyword arguments
```

#### 44.What is Scope? Types of scopes in python?

Scope refers to the area of the program where a variable is recognized and accessible. It determines where you can use a variable (inside a function, outside, globally, etc.).

##### Types of Scopes in Python:

Python follows the **LEGB Rule** for scope:

- **L:** Local
- **E:** Enclosing
- **G:** Global
- **B:** Built-in

#### 45.What is difference between global scope vs local scope?

Aspect	Global Scope	Local Scope
<b>Definition</b>	Variable declared outside all functions.	Variable declared inside a function.
<b>Access</b>	Can be accessed anywhere in the program.	Can be accessed only inside the function where it is defined.
<b>Lifetime</b>	Exists throughout the entire program.	Exists only while the function is running.
<b>Modification</b>	To modify inside a function, you must use the global keyword.	Modified freely within the function.
<b>Example Use</b>	Constants, configurations	Temporary calculations.

#### **46.What is difference between the nonlocal vs global keyword and usecase of them?**

Both global and nonlocal are used to modify variables outside the current local scope, but they apply to different levels of scope.

##### **global Keyword**

- Refers to variables declared **in the global scope** (outside any function).
- Allows a **function to modify a global variable**.

##### **nonlocal Keyword**

- Refers to variables declared **in the nearest enclosing function** (but not global).
- Allows an **inner (nested) function to modify a variable from the outer function**.

#### **47.How can we access local scoped variable outside of that local scope?Give me an example.**

By returning the local variable from the function. You cannot access a local variable directly from outside its scope. However, you can return it from the function and store it in a global or outer variable to use it outside.

```
def my_function():  
    x = 100 # Local variable  
    return x # Return the local variable  
  
# Accessing outside the function  
result = my_function()  
print("Accessed outside:", result)
```

#### **48.What is LEGB rule?**

Python follows the **LEGB Rule** for scope:

- **L:** Local
- **E:** Enclosing
- **G:** Global
- **B:** Built-in

#### **49.What are operators precedence? Write an example for that?**

Operator precedence in Python determines the order in which operators are evaluated in an expression. When multiple operators are used together, Python follows a specific hierarchy to decide which operation to perform first.

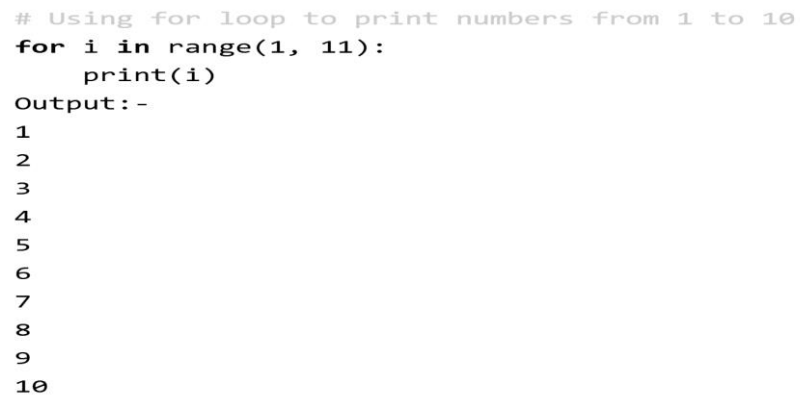
Example:-

```
result = 10 + 5 * 2
```

```
print(result)
```

## PROBLEM SOLVING(1-20)

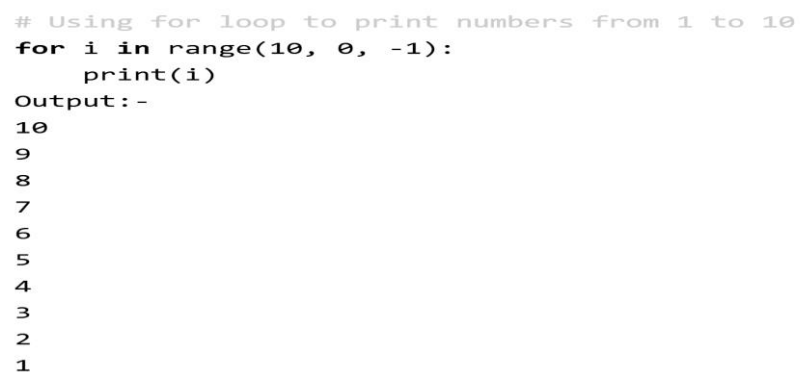
**1. Write a program to print numbers from 1-10 using for loop?**



```
# Using for loop to print numbers from 1 to 10
for i in range(1, 11):
    print(i)
Output:-
1
2
3
4
5
6
7
8
9
10
```

johndoe

**2. Write a program to print numbers from 10-1 using for loop?**



```
# Using for loop to print numbers from 10 to 1
for i in range(10, 0, -1):
    print(i)
Output:-
10
9
8
7
6
5
4
3
2
1
```

johndoe

**3. Write a program to print numbers from -1-(-10) using for loop?**



```
# Using for loop to print numbers from -1 to -10
for i in range(-1, -11, -1):
    print(i)
```

Output:-

```
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
```

johndoe

**4. Write a program to print numbers from -10(-1) using for loop?**



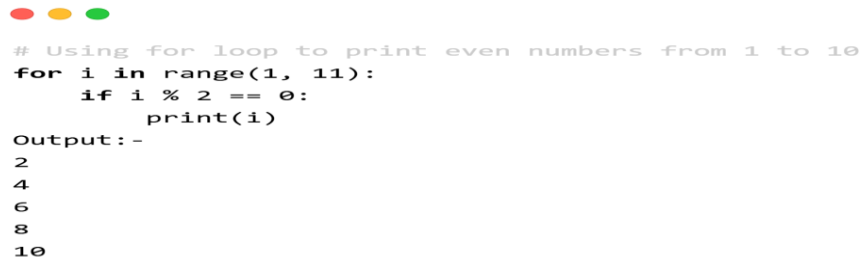
```
# Using for loop to print numbers from -10 to -1
for i in range(-10, 0, 1):
    print(i)
```

Output:-

```
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
```

johndoe

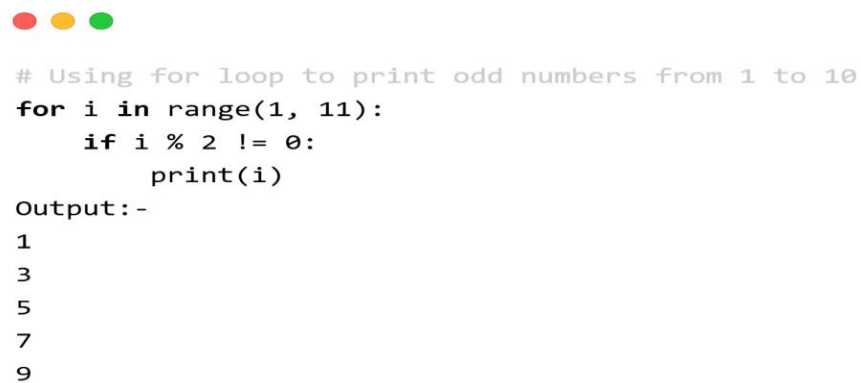
**5. Write a program to print even numbers from 1-10 using for loop?**

A code editor window with a white background and three colored window control buttons (red, yellow, green) at the top left. It contains Python code to print even numbers from 1 to 10. Below the code, the output is shown as a list of even numbers.

```
# Using for loop to print even numbers from 1 to 10
for i in range(1, 11):
    if i % 2 == 0:
        print(i)
Output:-
2
4
6
8
10
```

johndoe

**6. Write a program to print odd numbers from 1-10 using for loop?**

A code editor window with a white background and three colored window control buttons (red, yellow, green) at the top left. It contains Python code to print odd numbers from 1 to 10. Below the code, the output is shown as a list of odd numbers.

```
# Using for loop to print odd numbers from 1 to 10
for i in range(1, 11):
    if i % 2 != 0:
        print(i)
Output:-
1
3
5
7
9
```

johndoe



**7. Write a program to print sum of even numbers & sum of odd numbers from 1-10 using for loop?**

```

# Initialize sums
even_sum = 0
odd_sum = 0
# Using for loop from 1 to 10
for i in range(1, 11):
    if i % 2 == 0:
        even_sum += i # Add even numbers
    else:
        odd_sum += i # Add odd numbers
print("Sum of Even Numbers from 1 to 10:", even_sum)
print("Sum of Odd Numbers from 1 to 10:", odd_sum)
Output:-Sum of Even Numbers from 1 to 10: 30
Sum of Odd Numbers from 1 to 10: 25

```

johndoe

**8. Write program to reverse the string using for loop?**


```

name="akshitha"
for char in range(len(name)-1, -1, -1):
    print(name[char])

```

johndoe

**9. Write program to number and pick only prime number and store them in a list and print the list?**




```

# Create an empty list to store prime numbers
prime_numbers = []
# Loop through numbers from 1 to 50
for num in range(1, 51):
    if num > 1: # 1 is not a prime number
        for i in range(2, num):
            if num % i == 0:
                break # Not a prime number
        else:
            prime_numbers.append(num) # Add to list if prime
# Print the list of prime numbers
print("Prime numbers from 1 to 50 are:")
print(prime_numbers)

```

johndoe

**10. Write a program to get alphabets from a string using for loop in reverse direction and store them in empty str and find alphabets count?**



```

input_str = "abc123@#XYZ"
reverse_alphabets = ""
for char in input_str[::-1]:
    if char.isalpha():
        reverse_alphabets += char
print("Reversed Alphabets Only:", reverse_alphabets)
print("Count of Alphabets:", len(reverse_alphabets))

```

johndoe

**11. Write a program to iterate dictionary using for loop and print key: value pair?**



```
student = {  
    "name": "Akhi",  
    "age": 20,  
    "course": "Python",  
}  
for key, value in student.items():  
    print(f"{key}: {value}")
```

Output:-  
name: Akhi  
age: 20  
course: Python

johndoe

**12. Write a program to iterate a list and get only special symbols from the list using for loop?**



```
sample_list = ['A', '1', '@', 'B', '!', '3', '#', 'Z', '$']  
special_symbols = []  
for item in sample_list:  
    if not item.isalnum():  
        special_symbols.append(item)  
print("Special Symbols in the list:", special_symbols)  
Output:-Special Symbols in the list: ['@', '!', '#', '$']
```

johndoe

**13. Write a program to iterate a list in reverse order and get the alphabets in the list which are having length greater than 7?**

```

words_list = ["elephant", "sun", "butterfly", "python", "watermelon", "sky", "aeroplane"]
long_words = []
for word in words_list[::-1]:
    if word.isalpha() and len(word) > 7:
        long_words.append(word)
print(long_words)
Output:- ['aeroplane', 'watermelon', 'butterfly', 'elephant']

```

johndoe

**14. Write a program to get only prime numbers from the list using for loop?**

```

numbers = [2, 4, 5, 9, 11, 13, 15, 17, 19, 20, 23, 25, 29]
prime_numbers = []
for num in numbers:
    if num > 1:
        for i in range(2, num):
            if num % i == 0:
                break
        else:
            prime_numbers.append(num)
Print(prime_numbers)

```

johndoe

**15. Write a program to print highest digit in range 1-100 using for loop?**

```

highest_digit = 0
for num in range(1, 101):
    for digit in str(num):
        if int(digit) > highest_digit:
            highest_digit = int(digit)
        if highest_digit == 9:
            break
print(highest_digit)

```

johndoe

**16. Write a program to print the lowest digit in range -100-(-100) using for loop?**

```
min_digit = 9 # Start from the highest possible digit

for num in range(-100, 101):
    for digit in str(abs(num)):
        if digit.isdigit():
            if int(digit) < min_digit:
                min_digit = int(digit)
print(min_digit)
```

johnndoe

**17. Write a program to print the sum of alternative digits in list with for loop?**

```
# Sample list of digits
digits_list = [1, 4, 3, 7, 2, 9, 5, 8]
sum_alternative = 0
for i in range(0, len(digits_list), 2):
    sum_alternative += digits_list[i]
print(sum_alternative)
```

johnndoe

**18. Write a program to print the product of alternate str lengths in list and check the final result is a prime or not?**

```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

str_list = ["hello", "world", "python", "is", "awesome", "language"]
product = 1
for i in range(0, len(str_list), 2):
    length = len(str_list[i])
    product *= length
print(product)
if is_prime(product):
    print(product, "is a Prime Number")
else:
    print(product, "is NOT a Prime Number")
```

johnndoe

## 19. Write a program to check str is palindrome or not?

```
input_str = input("Enter a string: ")
reversed_str = input_str[::-1]
if input_str == reversed_str:
    print(f'"{input_str}" is a Palindrome')
else:
    print(f'"{input_str}" is NOT a Palindrome')
```

johndoe

## 20. Write a program to iterate a list of strs and find every str item length and concatenate all lengths to final length variable and print the final length of all str together and check it is even or odd?

```
str_list = ["hello", "world", "python", "rocks"]
final_length_str = ""
for item in str_list:
    length = len(item)
    final_length_str += str(length)
final_length_int = int(final_length_str)
print("Final concatenated length:", final_length_int)
if final_length_int % 2 == 0:
    print(final_length_int, "is EVEN")
else:
    print(final_length_int, "is ODD")
```

johndoe