Q1 Why do we call Python as a general purpose and high-level programming language?

Ans:- Python call as a general purpose because:-

it is designed to be used in a range of applications, including **data science**, **software and web development**, **automation**, and generally getting stuff done.

Python call as a high-level programming language because:-

**Python is easy to utilize, powerful, and versatile, making it a great cull for beginners and experts kindred. Python’s readability makes it a great first programming language — it sanctions you to think like a programmer and not waste time with confusing syntax. Python is great for backend web development, data analysis, artificial intelligence, and scientific computing.**

Q2. Why is Python called a dynamically typed language?

Ans:- Python don't have any problem even if we don't declare the type of variable. It states the kind of variable in the runtime of the program. Python also take cares of the memory management which is crucial in programming. So, Python is a dynamically typed language. Let's see one example.

## assigning a value to a variable

x = [1, 2, 3]

## x is a list here

print(type(x))

## reassigning a value to the 'x'

x = True

## x is a bool here

print(type(x))

## we can also redefine 'x' as many times as we want

Output

<class 'list'>

<class 'bool'>

As we see, we didn't declare the type of variable in the program. Python will automatically recognize the type of variable with the help of the value in the runtime.

Q3. List some pros and cons of Python programming language?

### Ans:- Pros of python programming language

### 1. Beginner-Friendly

Python is an excellent platform for beginners who want to step into coding. This programming language is easy to learn, understand and code as it does not include too many technicalities.

Even the developers who are new to coding and do not have experience in it find it beginner-friendly. It is a verbose language because reading python is very similar to reading English.

The language does not need curly braces to define the blocks deep into the technicalities. However, indentation is mandatory, which further aids in the readability of code.

### 2. Large Community

The Python community boasts many coders, developers, professionals, and students on the same platform, helping one another and bringing the best out of it.

This feature makes Python one of the best platforms for developers as they can seek help from experienced and knowledgeable developers.

### 3. Flexible and Extensible

Python is extremely flexible and can be extended to other languages. Developers can write code in C  and C++ and build new features in the dynamically-typed language.

### 4. Extensive Libraries

Python features an extensive set of libraries and contains code for various purposes like documentation-generation, regular expressions, web-browsers, unit-testing, CGI, databases, image manipulation, etc. Hence, it eliminates the need to write the complete code manually.

### 5. Embeddable

Python is extensible, and most of its codes can even be written in other languages such as C++. This lets us add scripting capabilities to our code in the other language.

### 6. Highly Scalable

Python is scalable, and many apps are in the market to prove it. Instagram and Pinterest are two of the most popular apps, receiving millions of user requests successfully.

Instagram, Pinterest, and Facebook are some of the most popular applications in everyday use that outshine the power of Python.

### 7. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the [**Internet Of Things**](https://www.pixelcrayons.com/blog/what-are-pros-and-cons-of-internet-of-things/).

Python is a way to connect the language with the real world. With Python and Raspberry Pi, the future for IoT is bright. Raspberry is a popular device used to integrate **[IoT technologies](https://www.pixelcrayons.com/blog/top-8-emerging-iot-technologies/" \t "_blank)**into real-world applications.

### 8. Machine Learning

Python is an excellent choice when your business needs web projects to be integrated with Machine Learning algorithms. Web Projects that require Python use it because it provides rich libraries, making it easier for the developers to work with data.

### 9. Portable

Python is portable, which means it can be run on any other platform. Here, you need to code only once, and you can run it anywhere. This is called WORA (Write Once Run Anywhere). This makes it easier for the developers to work with Python as they do not need to make changes to it in case they want to run it on another platform.

**Cons of python programming language**

### 1. Issues with Design

Python developers sometimes have to deal with complicated designs. Therefore, highly efficient and experienced developers are preferred over beginners.

### 2. Slower than Compiled Languages

Python is slow compared to other non-compiled languages as it requires a lot of computational power. So, this is the thing that you need to look at before choosing Python.

### 3. Security

Python is not 100% secure. You need to take the necessary steps to ensure the code’s security. However, performing the right QA testing can fix this concern.

### 4. Work Environment

Before working with Python, you must set up a work environment. Many trouble can arise while setting up the environment, affecting beginners’ programming experience.

So, it would be better to go for Python IDEs and set up the environment with the help of official guides.

### 5. Python’s Memory Consumption and Garbage Collection

Python language comes with high memory usage, and the high memory consumption should be tackled carefully during the project. It uses reference counting in its garbage collection, which often leads to potential memory losses.

### 6. Python is Dynamically Typed

Developers working in Data Science and Machine Learning projects prefer statically-typed language over dynamically typed language. The reason is that the statically typed language potentially minimizes a lot of errors and bugs in the system.

### 7. Multithreading in Python

Multithreading in Python is not exactly true multithreading due to its global interpreter lock (GIL). The multithreading model in Python does not have threads that run at the same time. Here, one thread can hold the GIL at one time, which clearly implies that this is not the exact multithreading.

Python developers need to use a different implementation of Python such as Jython, IronPython, PyPy, or C extension to achieve true multithreading.

Q4. In what all domains can we use Python?

Ans:- Employing python allows the user to work on multiple domains ranging from**Data Science, Machine Learning, Deep Learning, Artificial Intelligence, Scientific Computing Scripting, Networking, Game Development to Web Development**.

Q5. What are variable and how can we declare them?

Ans:- A Python variable is **a symbolic name that is a reference or pointer to an object**. Once an object is assigned to a variable, you can refer to the object by that name. In Python, variables are **a symbolic name that is a reference or pointer to an object**. The variables are used to denote objects by that name.

|  |
| --- |
| Python is a dynamic-typed language, which means we don't need to mention the variable type or declare before using it. It makes to Python the most efficient and easy to use language. Every variable is treated as an object in Python.  Before declaring a variable, we must follow the given rules.   * The first character of the variable can be an alphabet or (\_) underscore. * Special characters (@, #, %, ^, &, \*) should not be used in variable name. * Variable names are case sensitive. For example - age and AGE are two different variables. * Reserve words cannot be declared as variables.   Let's understand the declaration of a few basic variables. Numbers [Python](https://www.javatpoint.com/python-tutorial) supports three types of numbers - integer, floating point numbers, and complex. We can declare a variable with any length, there is no restriction declares any length of the variable. Use the following syntax to declare the number type variables.  **Example -**   1. num = 25 2. **print**("The type of a", type(num)) 3. **print**(num) 5. float\_num = 12.50 6. **print**("The type of b", type(float\_num)) 7. **print**(float\_num) 9. c = 2 + 5j 10. **print**("The type of c", type(c)) 11. **print**("c is a complex number", isinstance(1 + 3j, complex))   **Output:**  The type of a <class 'int'>  25  The type of b <class 'float'>  12.5  The type of c <class 'complex'>  c is a complex number True Strings The string is the sequence of Unicode characters. It is declared using single quotes, double quotes, or triple quotes. Let's understand the following example.  **Example -**   1. str\_var = 'JavaTpoint' 2. **print**(str\_var) 3. **print**(type(str\_var)) 5. str\_var1 = "JavaTpoint" 6. **print**(str\_var1) 7. **print**(type(str\_var1)) 9. str\_var3 = '''''This is string 10. using the triple 11. Quotes''' 12. **print**(str\_var3) 13. **print**(type(str\_var1))   **Output:**  JavaTpoint  <class 'str'>  JavaTpoint  <class 'str'>  This is string  using the triple  Quotes  <class 'str'> Multiple Assignments **1. Assigning multiple values to multiple variables**  We can assign the more than one variable simultaneously on the same line. For example -   1. a, b = 5, 4 2. **print**(a,b)   **Output:**  5 4  Values are printed in the given order.  **2. Assign a single value to the multiple variables**  We can assign the single value to the multiple variables on the same line. Consider the following example.  **Example -**   1. a=b=c="JavaTpoint" 2. **print**(a) 3. **print**(b) 4. **print**(c)   **Output:**  JavaTpoint  JavaTpoint  JavaTpoint |

Q6. How can we take an input from the user in Python?

Ans:- In Python, Using the input() function, we take input from a user, and using the print() function, we display output on the screen. Using the input() function, users can give any information to the application in the strings or numbers format.

In Python 3, we have built-in function to handle input from a user

input(prompt): To accept input from a user

In Python 2,we can use the following two functions:

1. input([prompt])
2. raw\_input([prompt])

The input() function reads a line entered on a console or screen by an input device such as a keyboard, converts it into a string.

# Python Input: Take Input from User

Updated on: April 8, 2021 | [40 Comments](https://pynative.com/python-input-function-get-user-input/#llc_comments)

In Python, Using the input() function, we take input from a user, and using the print() function, we display output on the screen. Using the input() function, users can give any information to the application in the strings or numbers format.

## Python Input() function

In Python 3, we have the following two built-in functions to handle input from a user and system.

1. input(prompt): To accept input from a user.
2. print(): To display output on the console/screen.

In Python 2,we can use the following two functions:

1. input([prompt])
2. raw\_input([prompt])

The input() function reads a line entered on a console or screen by an input device such as a keyboard, converts it into a string. As a new developer, It is essential to understand what is input in Python.

**The input is a value provided by the system or user**. For example, suppose you want to calculate the addition of two numbers on the calculator, you need to provide two numbers to the calculator. In that case, those two number is nothing but an input provided by the user to a calculator program.

### Python Example to Accept Input From a User

Let see how to accept employee information from a user.

* First, ask employee name, salary, and company name from the user
* Next, we will assign the input provided by the user to the [variables](https://pynative.com/python-variables/)
* Finally, we will use the print() function to display those variables on the screen.

# take three values from user

name = **input**("Enter Employee Name: ")

salary = **input**("Enter salary: ")

company = **input**("Enter Company name: ")

# Display all values on screen

**print**("\n")

**print**("Printing Employee Details")

**print**("Name", "Salary", "Company")

**print**(name, salary, company)

**Output**:

Enter Employee Name: Jessa

Enter salary: 8000

Enter Company name: Google

Printing Employee Details

Name Salary Company

Jessa 8000 Google

Q7. What is the default datatype of the value that has been taken as an input using input() function?

Ans:- In Python, we implement the input() function to get user input. The input function translates whatever you give it as input into a string. Even if an integer value is entered, the input() method accepts it as a string.

Syntax: input(prompt)

Parameter:

Prompt: (optional) The string to write to standard output (typically the screen) without a newline.

By default, it returns a string object.

**Hence, the input() function by default returns the value as string data type​.**

Q8. What is type casting?

Ans:- **The conversion of one data type into the other data type** is known as type casting in python or type conversion in python. Python supports a wide variety of functions or methods like: int(), float(), str(), ord(), hex(), oct(), tuple(), set(), list(), dict(), etc.

# How to do the type casting in python ??

# int() , float(), str(), bool()

int\_var = int\_var + 10 # int\_var = 10 + 10 and in next step int\_var = 20

casted\_int\_var = float(int\_var)

casted\_float\_var = int(float\_var)

print("Int to Float Typecasting for int\_var = ",casted\_int\_var)

print("Float to Int Typecasting for float\_var = ",casted\_float\_var)

Q9. Can we take more than one input from the user using single input() function? If yes, how? If no, why?

Ans:- Yes, we can take multiple inputs from the user using the input() function in Python.

Using split() function: You can ask the user to enter multiple values separated by a space or a comma, and then use the split() function to split the input string into a list of values. For example:

values = input("Enter multiple values separated by space: ").split()

print(values)

Q10. What are keywords?

Ans:- **Keywords** are reserved words in Python that are used to trigger specific tasks. We cannot use a keyword as a variable name, function name, or any other identifier since these keywords each hold a special meaning. All of the keywords in Python are lowercase.

The table below shows the list of keywords and their meanings:

| **Keyword** | **Meaning** |
| --- | --- |
| print | Prints to console. |
| while | Controls the flow of the code by starting a loop. |
| for | Iterates over items of a collection in the order that they appear. |
| continue | Interrupts the current cycle. The remaining part of the current iteration will be skipped and the program will move on to the next iteration. |
| break | Exits the (loop) cycle. |
| if | Determines which statements are going to be executed. |
| elif | Stands for “else if”. If the first test evaluates to False, then this keyword checks for the next one. |
| else | Is optional. The statement after the else keyword is executed when the if condition is False. |
| is | Tests for object identity. |
| not | Negates a Boolean value. |
| and | All conditions in a Boolean expression must be met. |
| or | At least one condition must be met. |
| import | Imports other modules into a Python script. |
| as | Gives a module a different alias. |
| from | Imports a specific variable, class, or function from a module. |
| def | Creates a new user-defined function. |
| return | Exits the function and returns a value if needed. |
| lambda | Creates a new anonymous function. |
| global | Accesses variables defined outside functions. |
| try | Specifies exception handlers. |
| except | Catches the exception and executes codes. |
| raise | Creates a user-defined exception. |
| del | Deletes an object. |
| pass | Tells the program to do nothing. Used in if-else statements to make the loop do nothing upon encountering certain conditions. |
| assert | Used for debugging purposes. |
| class | Creates a new user-defined object. |
| exec | Executes Python code dynamically. |
| yield | Used with generators. |

Q11. Can we use keywords as a variable? Support your answer with reason.

Ans:- Yes we can but we shouldn't as it will override the properties of the keyword.

.

Q12. What is indentation? What's the use of indentaion in Python?

Ans:- Indentation in Python is simply the spaces at the beginning of a code line. In Python, the indentation is an **essential** and **mandatory** concept that should be followed when writing a python code; otherwise, the python interpreter throws IndentationError. Indentation is the leading whitespace (spaces or/and tabs) before any statement in [Python](https://www.scaler.com/topics/python/). The reason why indentation is important in python is that the indentation serves another purpose other than code readability. Python treats the statements with the same indentation level (statements with an equal number of whitespaces before them) as a single code block. In python a block is a group of statements that have the same Indentation level i.e same number of leading whitespaces.

The identification of a block of code in Python is done using Indentation.  in Python, it is done using the spaces or tabs, which is known as indentation and also it is generally known as 4 space rule in Pep8 documentation of rules for styling and designing the code for Python. Let us consider an example of this.

#### Example #1

**Code:**

n = 10

if n>5:

print "n is greater than 5"

else:

print "n is not greater than 5"

**Output:**

n is greater than 5

In the above block of code, the indentation is used after the “if” and “else” statement so that the Python interpreter can [**execute the print statement**](https://www.educba.com/print-statement-in-python/) and gives the proper output; else if it is not properly indented, then it would throw us an error which is seen in below output. The print statement (“ n is greater than 5” ) and the print statement ( “ n is not greater than 5” )are two different clocks of code. To indicate these blocks of code, Python uses indentation at the beginning of each line of the block with the same number of spaces that are 4 spaces.

Q13. How can we throw some output in Python?

Ans:- Using print() function.

We can throw some output in python using print() function. For example

# Function in python for Output

# Function does what ? They might or minght not accept some parameter

print("Hello World !!!!")

Q14. What are operators in Python?

Ans:- In Python, operators are **special symbols that designate that some sort of computation should be performed**. The values that an operator acts on are called operands. Here is an example: >>> >>> a = 10 >>> b = 20 >>> a + b 30. In this case, the + operator adds the operands a and b together.

# Numerical operators in Python

# + for addition

# - for substractions

# \* for multiplication

# / for float division

# // for integer division

# \*\* for power calculation

# % for Modulus

x = 5

y = 3

print("Addition of x + y = ", x+y)

print("Substraction of x - y = ", x-y)

print("Multiplication of x \* y = ", x\*y)

print("Float Division of x / y = ", x/y)

print("Integer Divison of x // y = ", x//y)

print("Modulus of x % y = ", x%y)

print("Power of y on x i.e; x \*\* y = ", x\*\*y)

# concat operation for string

full\_name = str\_data + " " + "mishra"

print("Full name = ", full\_name)

# if we can use - as well ? IT will not work

minus\_str = "shashank" - "mishra"

print("Minus str = ", minus\_str)

multiply\_str = "shashank"\*"mishra"

print("Multiply str = ", multiply\_str)

power\_str = "shashank"\*\*"mishra"

print("Power str = ", power\_str)

power\_str = "shashank" \*\* 3

print("Power str = ", power\_str)

multiply\_numeric\_str = "shashank"\*5

print("Multiply numeric str = ", multiply\_numeric\_str)

# Assignment operators

# = , x = 5

# += , x += 5 -> x = x + 5

# -= , x -= 5 -> x = x - 5

# \*= , x \*= 5 -> x = x \* 5

# /= , x /= 5 -> x = x / 5

# //= , x //= 5 -> x = x // 5

# %= , x %= 5 -> x = x % 5

# Comparison Operators ( we compare operand values)

# == , Equals to condition , x == y

# != , Not Equals to condition , x != y

# > , Greater than condition , x > y

# < , Less than condition , x < y

# >= , Greater than and Equals to condition , x >= y

# <= , Less than and Equals to condition , x <= y

a = 10

b = 5

print("Result of a == b , ", a == b)

print("Result of a != b , ", a != b)

print("Result of a > b , ", a > b)

print("Result of a < b , ", a < b)

print("Result of a >= b , ", a >= b)

print("Result of a <= b , ", a <= b)

# logical operators in Python ( Logical check will happen for expression result)

# and -> Returns true if both statements are true

# or -> Returns true if one of statements are true

# not -> Reverse the result, returns false if the result is true

m = 10

n = 8

print("m>10 and n<10 Result " , m>10 and n<10) # False and True -> False

print("m>20 or n<10 Result " , m>10 or n<10) # False or True ->

print("not(m>20 and n<10) Result " , not(m>10 and n<10))

# not(False and True) -> not(False) -> True

Q15. What is difference between / and // operators?

Ans:- / is used for float division and // is used of floor (integer) division.

'/' is the division operator.

'//' is the floor division operator.

**Explanation:**

Python supports different types of operators:

They are arithmetic operators, logical operators, assignment operators, etc.

'/' and '//' belong to the arithmetic operators.

'/' is used for the normal division of two numbers.

'//' is used to obtain the smallest integer nearest to the quotient obtained by dividing two numbers.

Let us see an example to understand this.

x = 15

y = 3

print(x / y)   #This prints output as 5

print(x // y)  #This prints output as 5

a = 19

b = 4

print(a // b)  #This prints output as 4

print(a / b)  #This prints output as 4.75

So, if the quotient obtained by dividing two numbers is not an integer, then operators '/' and '//' will return different answers.

**'/' is the division operator. '//' is the floor division operator.**

/ for float division

// for integer division

x = 5

y = 3

print("Float Division of x / y = ", x/y)

print("Integer Divison of x // y = ", x//y)

Q16. Write a code that gives following as an output.

```

iNeuroniNeuroniNeuroniNeuron

```

Ans:- print("ineuron"\*4)

Q17. Write a code to take a number as an input from the user and check if the number is odd or even.

Ans:-

num = int (input (“Enter any number to test whether it is odd or even: “)

if (num % 2) == 0:

print (“The number is even”)

else:

print (“The provided number is odd”)

Output:

Enter any number to test whether it is odd or even:

887

887 is odd.

The program above only accepts integers as input. However, float type numbers can also be used to check whether they are even or odd. The program will work in the same way.

Q18. What are boolean operator?

Ans:- Boolean Operators are those that result in the Boolean values of True and False. These include and, or and not. While and & or require 2 operands, not is a unary operator. Boolean operators are most commonly used in arithmetic computations and logical comparisons. True , False , not , and , or are the only built-in Python Boolean operators.

# logical operators in Python ( Logical check will happen for expression result)

# and -> Returns true if both statements are true

# or -> Returns true if one of statements are true

# not -> Reverse the result, returns false if the result is true

For example

m = 10

n = 8

print("m>10 and n<10 Result " , m>10 and n<10) # False and True -> False

print("m>20 or n<10 Result " , m>10 or n<10) # False or True ->

print("not(m>20 and n<10) Result " , not(m>10 and n<10))

# not(False and True) -> not(False) -> True

Q19. What will the output of the following?

```

1 or 0

0 and 0

True and False and True

1 or 0 or 0

```

Ans:- 1.True and False and True->False

m = 10

n = 8

q=5

print("m==10 and n>10 and q<=5 Result " , m==10 and n>10 and q<=5)

Output: m==10 and n>10 and q<=5  Result  False

2. 1 or 0

a = 1

b = 0

print("a | b =", a | b)

Output: a | b = 1

3. 0 and 0

a = 0

b = 0

print("a & b =", a & b)

Output: a & b = 0

4. 1 or 0 or 0

a=1

b=0

c=0

print("a | b | c =", a | b | c)

Output: a | b | c = 1

 Output of the following code will be

1 or 0 -> 1

0 and 0 -> 0

True and False and True -> False

1 or 0 or 0 -> 1

Q20. What are conditional statements in Python?

Ans:- As the name suggests, conditional statements are responsible for handling different conditions in your programs such as If condition in Python. They are present in every programming language, including Python. These statements help your program by forming decisions based on the conditions encountered by the same. In large projects we have to control the flow of execution of our program and we want to execute some set of statements only if the given condition is satisfied, and a different set of statements when it’s not satisfied.Typically, there are three types of conditional statements, namely,

* Python If Statement
* Python If Else statement and
* If-elif-else statements
* # How to use If-Else in Python
* x = 10
* y = 5
* if x==y:
* print("Yes, X is Equals to Y !!")
* else:
* print("No, X is not Equals to Y !!")
* # Is it mandatory to use else block with if?
* a = 50
* if a==50:
* print("Yes, A is Equals to 50 !!")
* print("Bye !!")
* a = 40
* if a==50:
* print("Yes, A is Equals to 50 !!")
* print("Bye !!")
* # Nested if-else condition
* marks = 54
* if marks>=90:
* print("Grade A+")
* elif marks>=80 and marks<90:
* print("Grade A")
* elif marks>=70 and marks<80:
* print("Grade B+")
* elif marks>=60 and marks<70:
* print("Grade B")
* else:
* print("Grade C")

Q21. What is use of 'if', 'elif' and 'else' keywords?

Ans:- if is the first condition check for the condition.

if "if" is False then elif's condition is checked.

else is checked when all the upper condition fails.

# How to use If-Else in Python

x = 10

y = 5

if x==y:

print("Yes, X is Equals to Y !!")

else:

print("No, X is not Equals to Y !!")

# Is it mandatory to use else block with if?

a = 50

if a==50:

print("Yes, A is Equals to 50 !!")

print("Bye !!")

a = 40

if a==50:

print("Yes, A is Equals to 50 !!")

print("Bye !!")

# Nested if-else condition

marks = 54

if marks>=90:

print("Grade A+")

elif marks>=80 and marks<90:

print("Grade A")

elif marks>=70 and marks<80:

print("Grade B+")

elif marks>=60 and marks<70:

print("Grade B")

else:

print("Grade C")

Q22. Write a code to take the age of person as an input and if age >= 18 display "I can vote". If age is < 18 display "I can't vote".

Ans:- age = int(input("Enter you age: "))

if age >= 18:

print("I can vote")

else:

print("I can't vote")

Output: Enter your age: 20

I can vote

or

age = int(input("Enter your age: "))

if age >= 18:

  print("I can vote")

elif age<18:

  print("I cant vote")

else:

   print("bye")

Output: Enter your age: 20

I can vote

Q23. Write a code that displays the sum of all the even numbers from the given list.

```

numbers = [12, 75, 150, 180, 145, 525, 50]

```

Ans:- numbers = [12, 75, 150, 180, 145, 525, 50]

add = 0

for num in numbers:

if num%2 == 0:

add = add+num

else:

continue

print(add)

Output: 392

Q24. Write a code to take 3 numbers as an input from the user and display the greatest no as output.

Ans:- x, y, z = input("Enter 3 numbers seprated by comma: ").split(",")

if int(x) > int(y) and int(x) > int(z):

print(f"{x} is greatest")

elif int(y) > int(z):

print(f"{y} is greatest")

else:

print(f"{z} is greatest")

Output: Enter 3 numbers seprated by comma: 2,6,8

8 is greatest

Q25. Write a program to display only those numbers from a list that satisfy the following conditions

- The number must be divisible by five

- If the number is greater than 150, then skip it and move to the next number

- If the number is greater than 500, then stop the loop

```

numbers = [12, 75, 150, 180, 145, 525, 50]

```

Ans:- numbers = [12, 75, 150, 180, 145, 525, 50]

lst = []

for num in numbers:

if num > 150:

if num > 500:

break

elif num%5==0:

lst.append(num)

print(lst)

Output: [75, 150, 145]