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

* **What does high-level language (HLL) mean?**

In computer programming, a high-level language (HLL) is a programming language with strong abstraction from the details of the computer. Such languages are easier and more natural for people to use than machine code or assembly language, which require knowledge of details that are hidden from the user by the HLL. The difference between low-level and high-level programming languages is that Low-level languages are closer to machine code, which is the code that a computer executes directly. High-level languages are further from machine code and closer to natural languages like English.

In general, high-level languages are less efficient in terms of memory and processing power than low-level languages because the language has to provide an abstraction for the programmer. However, this trade-off is usually worth it because high-level languages make programming easier and more fun!

* **What makes python a high-level programming language?**

In a general understanding of highest-to-lowest languages, Python tops the list: Python/PHP, Ruby and Perl, Google Go, Java/C/C++, Assembly

* Python is not your straight compiled language (like Java or C) but an interpreted dynamic language that has to be run in the given system using another program instead of its local processor. When compiled, other languages turn into Assembly and run directly in the processor.
* Also, Python is a high-level programming language that is known for its ease of readability. Python’s syntax is designed to be easy to read and understand resulting in fewer coding steps for developers than imposed by Java or C++.
* Python is also portable, meaning that it can run on any platform, including Windows, Mac, Linux, and even Raspberry Pi.
* In addition, Python supports development in AI which is a rapidly growing field, and Python is a language that is well-suited for development in this area.
* **Why do we call Python as a general-purpose programming language?**

In computer software, a general-purpose programming language (GPL) is a programming language designed to be used for building software in a wide variety of application domains, across a multitude of hardware configurations and operating systems. In theory, a GPL program can run equivalently on a single processor or on a network.

The opposite of a general-purpose programming language is a domain-specific programming language, which is designed to be used within a specific area, for example, querying databases. For example, SQL was designed for querying databases.

**Q2. Why is Python called a dynamically typed language?**

Before we understand a dynamically typed language, we should learn about what typing is. Typing refers to type-checking in programming languages. In a stronglytyped language, such as Python, "1" + 2 will result in a type error since these languages don't allow for "type-coercion" (implicit conversion of data types). On the other hand, a weakly-typed language, such as Javascript, will simply output "12" as result. Type-checking can be done at two stages - Static - Data Types are checked before execution. Dynamic - Data Types are checked during execution. Python is an interpreted language, executes each statement line by line and thus type-checking is done on the fly, during execution. Hence, Python is a Dynamically Typed Language.

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

## **Pros of Python**

## **Beginner-Friendly:** 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.

### 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.

### 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.

### 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.

### 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.

### 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 into real-world applications.

### 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.

### 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**

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.
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 troubles 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?**

* 1. Machine learning / Artificial intelligence
  2. Desktop GUI
  3. Data analytics and data visualization
  4. Web development
  5. Game development
  6. Mobile app development
  7. Embedded systems
  8. Data Science

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

* + Variables are nothing but reserved memory locations to store values. It means that when you create a variable, you reserve some space in the memory. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to the variables, you can store integers, decimals or characters in these variables.
  + Python has no command for declaring a variable. A variable is created when some value is assigned to it. The value assigned to a variable determines the data type of that variable. Thus, declaring a variable in Python is very simple.
    - Just name the variable
    - Assign the required value to it
    - The data type of the variable will be automatically determined from the value assigned, we need not define it explicitly.

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

Python allows for user input.

That means we are able to ask the user for input.

Python 3 uses the input() method.

username = input("Enter username:")  
print("Username is: " + username)

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

String. Even if an integer value is entered, the input() method accepts it as a string.

**Q8. What is type casting?**

Type Casting is the method to convert the variable data type into a certain data type in order to the operation required to be performed by users. In this article, we will see the various technique for typecasting.

There can be two types of Type Casting in Python –

* Implicit Type Casting
* Explicit Type Casting

**Implicit Type Conversion**

In this, methods, Python converts data type into another data type automatically. In this process, users don’t have to involve in this process.

* Python3

|  |
| --- |
| # Python program to demonstrate  # implicit type Casting    # Python automatically converts  # a to int  a **=** 7  **print**(type(a))    # Python automatically converts  # b to float  b **=** 3.0  **print**(type(b))    # Python automatically converts  # c to float as it is a float addition  c **=** a **+** b  **print**(c)  print(type(c))    # Python automatically converts  # d to float as it is a float multiplication  d **=** a **\*** b  **print**(d)  print(type(d)) |

**Output:**

<class 'int'>

<class 'float'>

10.0

<class 'float'>

21.0

<class 'float'>

**Explicit Type Casting**

In this method, Python need user involvement to convert the variable data type into certain data type in order to the operation required.

Mainly in type casting can be done with these data type function:

* **Int() :**Int() function take float or string as an argument and return int type object.
* **float() :**float() function take int or string as an argument and return float type object.
* **str() :**str() function take float or int as an argument and return string type object.

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

Yes, we can. Using:

* a) Using split() function

# Python program to take multiple inputs from the user

a, b = input("Enter two of your lucky number: ").split()

print("First lucky number is: ", a)

print("Second lucky number is: ", b)

**Output:**

Enter two of your lucky number: 7 1

﻿First lucky number is: 7

﻿Second lucky number is: 1

* b) Using input () function

#multiple inputs in Python using input

x, y = input("Enter First Name: "), input("Enter Last Name: ")

print("First Name is: ", x)

print("Second Name is: ", y)

**Output:**

Enter First Name: FACE

Enter Last Name: Prep

First Name is: FACE

Second Name is: Prep

* c) Using map () function

#multiple inputs in Python using map

x, y = map(int, input("Enter two values: ").split())

print("First Number is: ", x)

print("Second Number is: ", y)

**Output:**

Enter two values: 7 1

﻿First Number is: 7

﻿Second Number is: 1

* d) Using List Comprehension

#multiple inputs in Python using list comprehension

x, y = [x for x in input("Enter your name and age: ").split(",")]

print("Your name is: ", x)

print("Your age is: ", y)

**Output:**

Enter your name and age: FACE Prep, 8

﻿Your name is: FACE Prep

Your age is: 8

**Q10. What are keywords?**

**Keywords** are some predefined and reserved words in python that have special meanings. Keywords are used to define the syntax of the coding. The keyword cannot be used as an identifier, function, and variable name. All the keywords in python are written in lower case except True and False. There are 33 keywords in Python 3.7

| Keywords | Description |
| --- | --- |
| and | This is a logical operator it returns true if both the operands are true else return false. |
| Or | This is also a logical operator it returns true if anyone operand is true else return false. |
| not | This is again a logical operator it returns True if the operand is false else return false. |
| if | This is used to make a conditional statement. |
| elif | Elif is a condition statement used with an if statement the elif statement is executed if the previous conditions were not true |
| else | Else is used with if and elif conditional statement the else block is executed if the given condition is not true. |
| for | This is created for a loop. |
| while | This keyword is used to create a while loop. |
| break | This is used to terminate the loop. |
| as | This is used to create an alternative. |
| def | It helps us to define functions. |
| lambda | It is used to define the anonymous function. |
| pass | This is a null statement which means it will do nothing. |
| return | It will return a value and exit the function. |
| True | This is a boolean value. |
| False | This is also a boolean value. |
| try | It makes a try-except statement. |
| with | The with keyword is used to simplify exception handling. |
| assert | This function is used for debugging purposes. Usually used to check the correctness of code |
| class | It helps us to define a class. |
| continue | It continues to the next iteration of a loop |
| del | It deletes a reference to an object. |
| except | Used with exceptions, what to do when an exception occurs |
| finally | Finally is use with exceptions, a block of code that will be executed no matter if there is an exception or not. |
| from | The form is used to import specific parts of any module. |
| global | This declares a global variable. |
| import | This is used to import a module. |
| in | It’s used to check if a value is present in a list, tuple, etc, or not. |
| is | This is used to check if the two variables are equal or not. |
| None | This is a special constant used to denote a null value or avoid. It’s important to remember, 0, any empty container(e.g empty list) do not compute to None |
| nonlocal | It’s declared a non-local variable. |
| raise | This raises an exception |
| yield | It’s ends a function and returns a generator. |

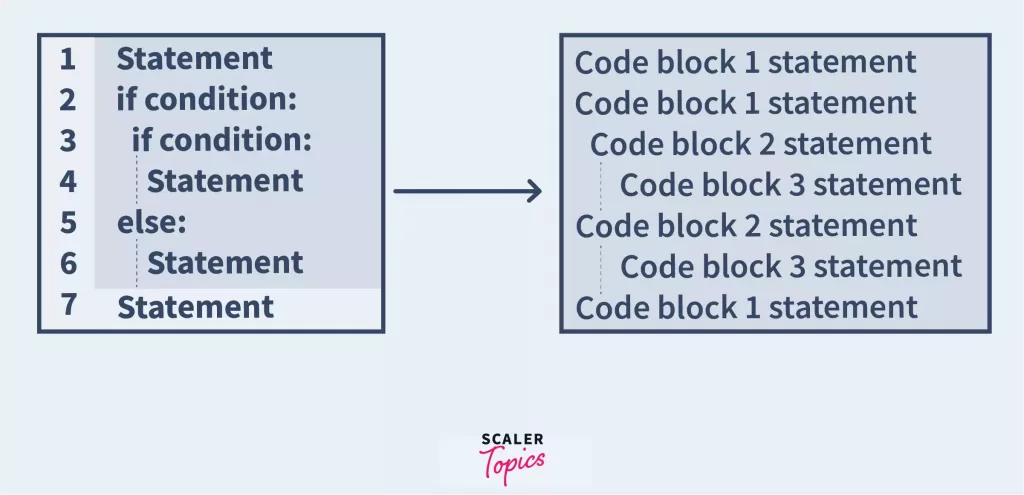
**Q11. Can we use keywords as a variable? Support your answer with reason.**

**Keywords** are some predefined and reserved words in python that have special meanings**. Keywords are case sensitive in python**. You can't use a keyword as variable name, function name or any other identifier name.

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

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. So whereas in languages like C, C++, etc. a block of code is represented by curly braces { }, in python a block is a group of statements that have the same Indentation level i.e. same number of leading whitespaces.



**Q13. How can we throw some output in Python?**

**Q14. What are operators in Python?**

Operators are used to perform operations on variables and values.

In the example below, we use the + operator to add together two values:

print(10 + 5)

## **Python Arithmetic Operators:** Arithmetic operators are used with numeric values to perform common mathematical operations:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| + | Addition | x + y |
| - | Subtraction | x - y |
| \* | Multiplication | x \* y |
| / | Division | x / y |
| % | Modulus | x % y |
| \*\* | Exponentiation | x \*\* y |
| // | Floor division | x // y |

## **Python Assignment Operators:** Assignment operators are used to assign values to variables:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| \*= | x \*= 3 | x = x \* 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| //= | x //= 3 | x = x // 3 |
| \*\*= | x \*\*= 3 | x = x \*\* 3 |
| &= | x &= 3 | x = x & 3 |
| |= | x |= 3 | x = x | 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

## **Python Comparison Operators:** Comparison operators are used to compare two values:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| == | Equal | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

## **Python Logical Operators:** Logical operators are used to combine conditional statements:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| and | Returns True if both statements are true | x < 5 and  x < 10 |
| or | Returns True if one of the statements is true | x < 5 or x < 4 |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) |

## **Python Identity Operators:** Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| is | Returns True if both variables are the same object | x is y |
| is not | Returns True if both variables are not the same object | x is not y |

## **Python Membership Operators:** Membership operators are used to test if a sequence is presented in an object:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| in | Returns True if a sequence with the specified value is present in the object | x in y |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y |

## **Python Bitwise Operators:** Bitwise operators are used to compare (binary) numbers:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Description** |
| & | AND | Sets each bit to 1 if both bits are 1 |
| | | OR | Sets each bit to 1 if one of two bits is 1 |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 |
| ~ | NOT | Inverts all the bits |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off |

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

In Python programming, you can perform division in two ways. The first one is **Float Division("/")** and the second is **Integer Division("//")** or Floor Division.

5/2 = 2.5

5//2 = 2

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

iNeuroniNeuroniNeuroniNeuron

x = "iNeuron"

print(x\*4)

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

x = int(input("Enter a number: "))

if (x%2 == 0):

print(f"Entered number {x} is even")

else:

print(f"Entered number {x} is odd")

**Q18. What are boolean operator?**

The Python Boolean type is one of Python's built-in data types. It's **used to represent the truth value of an expression**. For example, the expression 1 <= 2 is True , while the expression 0 == 1 is False .

**Q19. What will the output of the following?**

1 or 0 = 1

0 and 0 = 0

True and False and True = False (1\*0\*1)

1 or 0 or 0 = 1

**Q20. What are conditional statements in Python?**

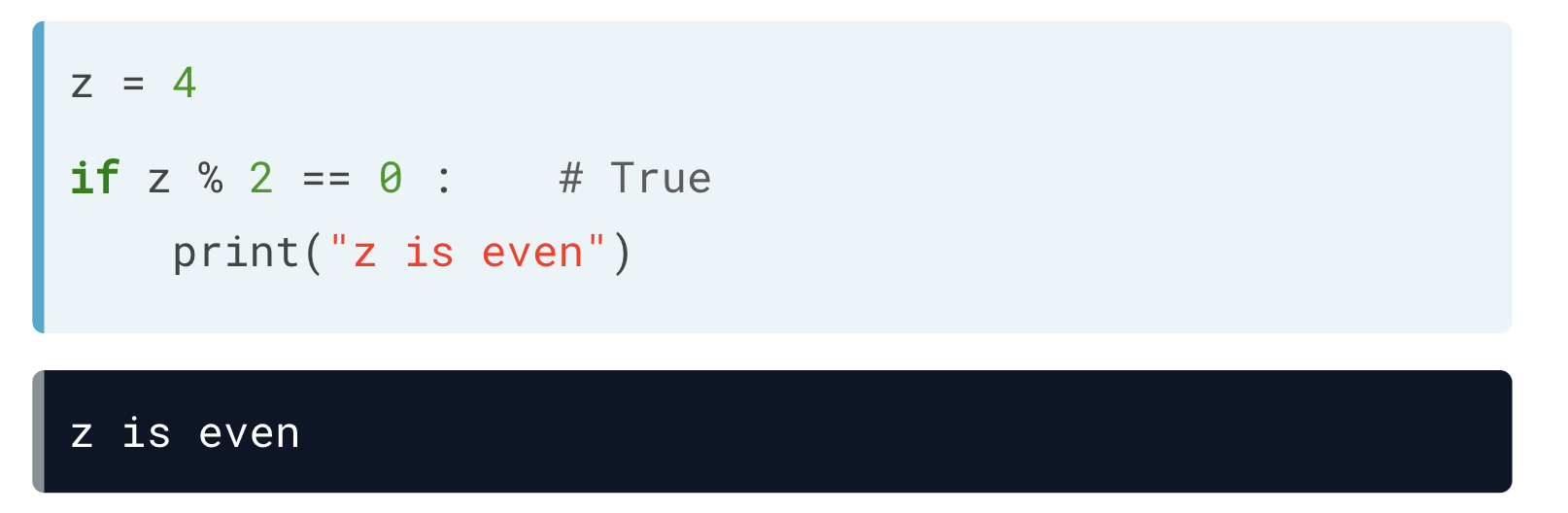
A conditional statement as the name suggests itself, is used to handle conditions in your program. These statements guide the program while making decisions based on the conditions encountered by the program.

Python has 3 key Conditional Statements that you should know:

* *if*statement
* *if-else* statement
* *if-elif-else* ladder

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

## **if condition:** The if condition is considered the simplest of the three and makes a decision based on whether the condition is true or not. If the condition is true, it prints out the indented expression. If the condition is false, it skips printing the indented expression.



## **if-else condition:** The if-else condition adds an additional step in the decision-making process compared to the simple if statement. The beginning of an if-else statement operates similar to a simple if statement; however, if the condition is false, instead of printing nothing, the indented expression under else will be printed.

x = int(input("Enter a number: "))

if (x%2 == 0):

print(f"Entered number {x} is even")

else:

print(f"Entered number {x} is odd")

## **elif condition:** The elif keyword is used in conditional statements (if statements), and is short for else if.

for i in range(-5, 5):  
  if i > 0:  
    print("YES")  
  elif i == 0:  
    print("WHATEVER")  
  else:  
    print("NO")

**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".**

age = int(input("Enter age of the person: "))

if age>= 18:

print("I can vote")

elif age<18:

print("I can't 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]

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

result = 0 # Initialize your results variable.

for i in numbers: # Loop through each element of the list.

if not i % 2: # Test for even numbers.

result += i

print(result)

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

a = int(input("Enter first number: "))

b = int(input("Enter second number: "))

c = int(input("Enter third number: "))

if (a >= b) and (a >= c):

largest = a

elif (b >= a) and (b >= c):

largest = b

else:

largest = c

print("The largest number is", largest)

**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]**

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

result = 0

for i in numbers:

if i%5==0:

if i>150:

if i>500:

print(i)

**Q26. What is a string? How can we declare string in Python?**

Strings in Python are arrays of bytes representing Unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

"Hello, World!"

**Q27. How can we access the string using its index?**

Square brackets can be used to access elements of the string.

a = "Hello, World!"  
print(a[1])

**Q28. Write a code to get the desired output of the following**

**string = "Big Data iNeuron"**

**desired\_output = "iNeuron"**

print(string[9:])

**Q29. Write a code to get the desired output of the following**

**string = "Big Data iNeuron"**

**desired\_output = "norueNi"**

print(string[-8:])

**Q30. Resverse the string given in the above question.**

print(string[::-1])

**Q31. How can you delete entire string at once?**

We can delete a entire string using del command.

a = "Hi iNuronAI"

print(a)

Hi iNuronAI

del a

print(a)

**NameError**: name 'a' is not defined

**Q32. What is escape sequence?**

Escape characters or sequences are illegal characters for Python and never get printed as part of the output. When backslash is used in Python programming, it allows the program to escape the next characters.

| **Escape Sequence** | **Meaning** |
| --- | --- |
| \’ | Single quote |
| \\’ | Double quote |
| \\ | Backslash |
| \n | Newline |
| \r | Carriage Return |
| \t | Horizontal Tab |
| \b | Backspace |
| \f | Formfeed |
| \v | Vertical Tab |
| \0 | Null Character |
| \N{Name} | Unicode character Database named lookup |
| \uxxxxxxxx | Unicode character with a 16-bit hex value |
| \Uxxxxxxxx | Unicode character with a 32-bit hex value |
| \000 | Character with octal value ooo |
| \xhh | Character with hex value hh |

**Q33. How can you print the below string?**

**'iNeuron's Big Data Course'**

print("'iNeuron's Big Data Course'")

**Q34. What is a list in Python?**

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are [Tuple](https://www.w3schools.com/python/python_tuples.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

Lists are created using square brackets:

thislist = ["apple", "banana", "cherry"]  
print(thislist)

**Q35. How can you create a list in Python?**

thislist = ["apple", "banana", "cherry"]  
print(thislist)

**Q36. How can we access the elements in a list?**

To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

**list1 = ['physics', 'chemistry', 1997, 2000];**

**list2 = [1, 2, 3, 4, 5, 6, 7 ];**

**print("list1[0]: ", list1[0])**

**print("list2[1:5]: ", list2[1:5])**

**Q37. Write a code to access the word "iNeuron" from the given list.**

**lst = [1,2,3,"Hi",[45,54, "iNeuron"], "Big Data"]**

lst = [1,2,3,"Hi",[45,54, "iNeuron"], "Big Data"]

lst[4][2]

**Q38. Take a list as an input from the user and find the length of the list.**

a = list(input("Enter numbers for your list: "))

print(a)

**Q39. Add the word "Big" in the 3rd index of the given list.**

**lst = ["Welcome", "to", "Data", "course"]**

lst = ["Welcome", "to", "Data", "course"]

lst.insert(2, "Big")

lst

**Q40. What is a tuple? How is it different from list?**

Tuples are a sequence data type containing elements of different [data types](https://www.simplilearn.com/what-is-data-article).

It comes in handy when storing a collection of items, especially if you want those items to be unchanging.

A python tuple has the following features -

* Tuples are used to store heterogeneous and homogeneous data.
* Tuples are immutable in nature.
* Tuples are ordered
* An index is used to traverse a tuple.
* Tuples are similar to lists. It also preserves the data sequence.

As tuples are immutable, they are faster than the list because they are static.

The primary difference between tuples and lists is that **tuples are immutable as opposed to lists which are mutable**. Therefore, it is possible to change a list but not a tuple.

Tuple are faster than the list because they are static.

num\_list = [1,2,3,4,5]

num\_tuple = (1,2,3,4,5)

**Q41. How can you create a tuple in Python?**

num\_tuple = (1,2,3,4,5)

**Q42. Create a tuple and try to add your name in the tuple. Are you able to do it? Support your answer with reason.**

**#create a tuple**

**tuplex = (4, 6, 2, 8, 3, 1)**

**print(tuplex)**

**#tuples are immutable, so you can not add new elements**

**#using merge of tuples with the + operator you can add an element and it will create a new tuple**

**tuplex = tuplex + ("Arka",)**

**print(tuplex)**

**Q43. Can two tuples be appended. If yes, write a code for it. If not, why?**

* We can't add elements to a tuple because of their immutable property. There's no append() or extend() method for tuples,
* We can't remove elements from a tuple, also because of their immutability. Tuples have no remove() or pop() method,

**Q44. Take a tuple as an input and print the count of elements in it.**

**tuplex = tuple(input("Enter elements for desired tuple: "))**

**len(tuplex)**

**Q45. What are sets in Python?**

Sets are used to store multiple items in a single variable. Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.asp), [Tuple](https://www.w3schools.com/python/python_tuples.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

**Q46. How can you create a set?**

thisset = {"apple", "banana", "cherry"}  
print(thisset)

**Q47. Create a set and add "iNeuron" in your set.**

thisset = {"apple", "banana", "cherry"}  
  
thisset.add("iNeuron")  
  
print(thisset)

**Q48. Try to add multiple values using add() function.**

**thisset = {"apple", "banana", "cherry"}**

**thisset.add("orange", "mango")**

**print(thisset)**

**TypeError**: set.add() takes exactly one argument (2 given)

**Q49. How is update() different from add()?**

In set we can add exactly one argument using add() function but using update() function we can add multiple arguments in that given set.

thisset = {"apple", "banana", "cherry"}  
mylist = ["kiwi", "orange"]  
  
thisset.update(mylist)  
  
print(thisset)

**Q50. What is clear() in sets?**

The clear() method removes all elements in a set.

**thisset = {"apple", "banana", "cherry"}**

**thisset.clear()**

**thisset**

set()

**Q51. What is frozen set?**

Frozen set is just an immutable version of a [Python set](https://www.programiz.com/python-programming/set) object. While elements of a set can be modified at any time, elements of the frozen set remain the same after creation.

Due to this, frozen sets can be used as keys in [Dictionary](https://www.programiz.com/python-programming/dictionary) or as elements of another set. But like sets, it is not ordered (the elements can be set at any index).

# tuple of vowels

vowels = ('a', 'e', 'i', 'o', 'u')

fSet = frozenset(vowels)

print('The frozen set is:', fSet)

print('The empty frozen set is:', frozenset())

# frozensets are immutable

fSet.add('v')

**AttributeError**: 'frozenset' object has no attribute 'add'

**Q52. How is frozen set different from set?**

Frozen set is just an immutable version of a Python set object. **While elements of a set can be modified at any time, elements of the frozen set remain the same after creation**.

**Q53. What is union() in sets? Explain via code.**

Union() method Return a set that contains all items from both sets, duplicates are excluded:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.union(y)  
  
print(z)

**Q54. What is intersection() in sets? Explain via code.**

Return a set that contains the items that exist in both set x, and set y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.intersection(y)  
  
print(z)

**Q55. What is dictionary in Python?**

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered, changeable and do not allow duplicates.

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(thisdict)

**Q56. How is dictionary different from all other data structures.**

**Q57. How can we declare a dictionary in Python?**

A Dictionary in python is declared by enclosing a comma-separated list of key-value pairs using curly braces({})

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(thisdict)

**Q58. What will the output of the following?**

**var = {}**

**print(type(var))**

<class 'dict'>

**Q59. How can we add an element in a dictionary?**

CountryCodeDict = {"India": 91, "UK" : 44 , "USA" : 1, "Spain" : 34}

print(CountryCodeDict)

CountryCodeDict.update( {'Germany' : 49} )

print(CountryCodeDict)

# Adding multiple key value pairs

CountryCodeDict.update( [('Austria', 43),('Russia',7)] )

print(CountryCodeDict)

**Q60. Create a dictionary and access all the values in that dictionary.**

We can access all the values using built in value() methode.

year\_of\_creation = {'Python': 1993, 'JavaScript': 1995, 'HTML': 1993}

print(year\_of\_creation.values())

**Q61. Create a nested dictionary and access all the element in the inner dictionary.**

people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},

2: {'name': 'Marie', 'age': '22', 'sex': 'Female'},

3: {'name': 'Luna', 'age': '24', 'sex': 'Female'},

4: {'name': 'Peter', 'age': '29', 'sex': 'Male'}}

print(people.values())

print(people.keys())

print(people)

**Q62. What is the use of get() function?**

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = car.get("model")

print(x)

output: Mustang

**Q63. What is the use of items() function?**

The items() method returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list.

The view object will reflect any changes done to the dictionary, see example below.

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = car.items()

car["year"] = 2018

print(x)

**output:** dict\_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 2018)])

**Q64. What is the use of pop() function?**

List pop in [Python](https://www.simplilearn.com/learn-the-basics-of-python-article) is a pre-defined, in-built [function](https://www.simplilearn.com/tutorials/python-tutorial/python-functions) that removes an item at the specified index from the [list](https://www.simplilearn.com/tutorials/python-tutorial/python-list). We can also use pop in Python without mentioning the [index](https://www.simplilearn.com/tutorials/python-tutorial/index-in-python) value. In such cases, the pop() function will remove the last element of the list.

**# Cars list**

**cars = ['Mercedes Benz', 'BMW', 'Jeep', 'Mahindra', 'Maserati']**

**print(cars)**

**# Using pop() and storing the return value**

**ret\_val = cars.pop(2)**

**print('The return value is:', ret\_val)**

**# Updated List**

**print('The updated list is:', cars)**

**Q65. What is the use of popitems() function?**

The popitem() method removes the item that was last inserted into the dictionary.

The removed item is the return value of the popitem() method, as a tuple.

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = car.popitem()

print(x)

**Output:** ('year', 1964)

**Q66. What is the use of keys() function?**

The keys() method returns a view object. The view object contains the keys of the dictionary, as a list. The view object will reflect any changes done to the dictionary

car = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
  
x = car.keys()  
  
car["color"] = "white"  
  
print(x)

**Output:** dict\_keys(['brand', 'model', 'year', 'color'])

**Q67. What is the use of values() function?**

The values() method returns a view object. The view object contains the values of the dictionary, as a list.

The view object will reflect any changes done to the dictionary

car = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
  
x = car.values()  
  
car["year"] = 2018  
  
print(x)

**Output:** dict\_values(['Ford', 'Mustang', 2018])

**Q68. What are loops in Python?**

A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement −



**Q69. How many type of loop are there in Python?**

Python programming language provides following types of loops to handle looping requirements.

|  |  |
| --- | --- |
| **Sr.No.** | **Loop Type & Description** |
| 1 | [while loop](https://www.tutorialspoint.com/python/python_while_loop.htm)  Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. |
| 2 | [for loop](https://www.tutorialspoint.com/python/python_for_loop.htm)  Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. |
| 3 | [nested loops](https://www.tutorialspoint.com/python/python_nested_loops.htm)  You can use one or more loop inside any another while, for or do..while loop. |

**Q70. What is the difference between for and while loops?**

|  |  |  |
| --- | --- | --- |
| **Basis of Comparison** | **For Loop** | **While Loop** |
| **Keyword** | Uses **for** keyword | Uses **while** keyword |
| **Used** | For loop is used when the number of iterations is already known. | While loop is used when the number of iterations is already Unknown. |
| **absence of condition** | The loop runs infinite times in the absence of condition | Returns the compile time error in the absence of condition |
| **Nature of Initialization** | Once done, it cannot be repeated | In the while loop, it can be repeated at every iteration. |
| **Functions** | To iterate, the range or xrange function is used. | There is no such function in the while loop. |
| **Initialization based on iteration** | To be done at the beginning of the loop. | In the while loop, it is possible to do this anywhere in the loop body. |
| **Generator Support** | Python's for loop can iterate over generators. | While loops cannot be directly iterated on Generators. |
| **Speed** | The for loop is faster than the while loop. | While loop is relatively slower as compared to for loop. |

**Q71. What is the use of continue statement?**

The continue keyword is used to end the current iteration in a for loop (or a while loop), and continues to the next iteration.

**i = 0**

**while i < 9:**

**i += 1**

**if i == 3:**

**continue**

**print(i)**

**Output:**

1

2

4

5

6

7

8

9

**Q72. What is the use of break statement?**

The break keyword is used to break out a for loop, or a while loop.

**i = 1**

**while i < 9:**

**print(i)**

**if i == 3:**

**break**

**i += 1**

**Output:**

**1**

**2**

**3**

**Q73. What is the use of pass statement?**

The pass statement is used as a placeholder for future code.

When the pass statement is executed, nothing happens, but you avoid getting an error when empty code is not allowed.

Empty code is not allowed in loops, function definitions, class definitions, or in if statements.

a = 33  
b = 200  
  
if b > a:  
  pass

**Q74. What is the use of range() function?**

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

**x = range(3, 6)**

**for n in x:**

**print(n)**

**Output:**

**3**

**4**

**5**

**Q75. How can you loop over a dictionary?**

We can loop through a dictionary by using a for loop.

When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.

* Print all key names in the dictionary, one by one:

for x in thisdict:  
  print(x)

* Print all *values* in the dictionary, one by one:

for x in thisdict:  
  print(thisdict[x])

**Coding problems**

**Q76. Write a Python program to find the factorial of a given number.**

**def factorial(x): # creating function definition**

**if x == 1:**

**return 1 # factorial 1 = 1**

**else:**

**return (x \* factorial(x-1)) # because 5!=5\*4\*3\*2\*1**

**num = int(input("Enter a number: ")) # input the number**

**F = factorial(num) # call the factorial function**

**print("The factorial is", F)**

**Q77. Write a Python program to calculate the simple interest. Formula to calculate simple interest is SI = (P*R*T)/100**

P = int(input("Enter the principle ammount: "))

R = float(input("Enter rate of interest(%): "))

T = float(input("Enter the time period: "))

SI = (P\*R\*T)/100

print(f"Simple Interest amount is {SI}")

**Q78. Write a Python program to calculate the compound interest. Formula of compound interest is A = P(1+ R/100)^t.**

**P = int(input("Enter the principle ammount: "))**

**R = float(input("Enter rate of interest(%): "))**

**t = float(input("Enter the time period: "))**

**A = P\*((1+ R/100)\*\*t)**

**print(f"Simple Interest amount is {A}")**

**Q79. Write a Python program to check if a number is prime or not.**

**# Program to check if a number is prime or not**

**# Input from the user**

**num = int(input("Enter a number: "))**

**# If number is greater than 1**

**if num > 1:**

**# Check if factor exist**

**for i in range(2,num):**

**if (num % i) == 0:**

**print(num,"is not a prime number")**

**break**

**else:**

**print(num,"is a prime number")**

**# Else if the input number is less than or equal to 1**

**else:**

**print(num,"is not a prime number")**

**Q80. Write a Python program to check Armstrong Number.**

**num = input("Enter a number")**

**armst=0**

**for i in num:**

**armst=armst+int(i)\*\*3**

**if(int(num)==armst):**

**print("number is armstrong")**

**else:**

**print("number is not a Armstrong")**

**Q81. Write a Python program to find the n-th Fibonacci Number.**

The Fibonacci Sequence is the series of numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

The next number is found by adding up the two numbers before it:

* the 2 is found by adding the two numbers before it (1+1),
* the 3 is found by adding the two numbers before it (1+2),
* the 5 is (2+3),
* and so on!

def solve(n):

if n <= 2:

return n - 1

else:

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

n = 8

print(solve(n))

**Q82. Write a Python program to interchange the first and last element in a list.**

**myList = [1, 7, 3, 90, 23, 4]**

**print("Initial List : ", myList)**

**# finding the length of list**

**length = len(myList)**

**# Swapping first and last element**

**temp = myList[0]**

**myList[0] = myList[length - 1]**

**myList[length - 1] = temp**

**print("List after Swapping : ", myList)**

**Q83. Write a Python program to swap two elements in a list.**

**def swapList(sl,pos1,pos2):**

**sl[pos1], sl[pos2] = sl[pos2], sl[pos1]**

**return sl**

**List = [9, 11, 5, 3, 6, 27, 4]**

**pos1, pos2= 3,5**

**print(List)**

**print("Swapped List: ",swapList(List,pos1-1,pos2-1))**

**Q84. Write a Python program to find N largest element from a list.**

**# Create a List**

**myList = [120, 50, 89, 170, 45, 250, 450, 340]**

**print("List = ",myList)**

**# The value of N**

**n = 4**

**# First, sort the List**

**myList.sort()**

**# Now, get the largest N integers from the list**

**print("Largest integers from the List = ",myList[-n:])**

**Q85. Write a Python program to find cumulative sum of a list.**

**a=[]**

**n= int(input("Enter the number of elements in list:"))**

**for x in range(0,n):**

**element=int(input("Enter element" + str(x+1) + ":"))**

**a.append(element)**

**b=[sum(a[0:x+1]) for x in range(0,len(a))]**

**print("The original list is: ",a)**

**print("The new list is: ",b)**

**Q86. Write a Python program to check if a string is palindrome or not.**

**x = input("Enter a string: ")**

**x\_rev = x[::-1]**

**if x == x\_rev:**

**print(f"Entered string {x} is palindrome")**

**else:**

**print(f"Entered string {x} is not palindrome")**

**Q87. Write a Python program to remove i'th element from a string.**

**def remove\_char(s, i):**

**a = s[ : i]**

**b = s[i + 1: ]**

**return a+b**

**string = "Pythonisgood"**

**# Remove ith index element**

**i = 5**

**print(remove\_char(string,i-1))**

**Q88. Write a Python program to check if a substring is present in a given string.**

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

**sub\_str= input("Enter word:")**

**if(string.find(sub\_str)==-1):**

**print("Substring not found in string!")**

**else:**

**print("Substring in string!")**

**Q89. Write a Python program to find words which are greater than given length k.**

**def word\_k(k, s):**

**# split the string where space comes**

**word = s.split(" ")**

**# iterate the loop for every word**

**for x in word:**

**# if length of current word**

**if len(x)>k:**

**# greater than k then**

**print(x)**

**k = 3**

**s = input("Enter Words: ")**

**word\_k(k, s)**

**Q90. Write a Python program to extract unique dictionary values.**

**my\_dict = {'hi' : [5,3,8, 0],**

**'there' : [22, 51, 63, 77],**

**'how' : [7, 0, 22],**

**'are' : [12, 11, 45],**

**'you' : [56, 31, 89, 90]}**

**print("The dictionary is : ")**

**print(my\_dict)**

**my\_result = list(sorted({elem for val in my\_dict.values() for elem in val}))**

**print("The unique values are : ")**

**print(my\_result)**

**Q91. Write a Python program to merge two dictionaries.**

**dict\_1 = {'X': 15, 'Y': 10, 'Z' : 12 }**

**dict\_2 = {'A': 18,'B': 20,'C' : 16 }**

**dict\_1.update(dict\_2)**

**print('Updated dictionary:')**

**print(dict\_1)**

**Q92. Write a Python program to convert a list of tuples into dictionary.**

**Input : [('Sachin', 10), ('MSD', 7), ('Kohli', 18), ('Rohit', 45)]**

**Output : {'Sachin': 10, 'MSD': 7, 'Kohli': 18, 'Rohit': 45}**

**dict\_1 = {'X': 15, 'Y': 10, 'Z' : 12 }**

**dict\_2 = {'A': 18,'B': 20,'C' : 16 }**

**dict\_1.update(dict\_2)**

**print('Updated dictionary:')**

**print(dict\_1)**

**Q93. Write a Python program to create a list of tuples from given list having number and its cube in each tuple.**

**Input: list = [9, 5, 6]**

**Output: [(9, 729), (5, 125), (6, 216)]**

**# Creating a list**

**myList = [9, 5, 6]**

**# Creating list of tuples**

**tupleList = []**

**for val in myList:**

**myTuple = (val, (val\*val\*val))**

**tupleList.append(myTuple)**

**# print the result**

**print("The list of Tuples is " , str(tupleList))**

**Q94. Write a Python program to get all combinations of 2 tuples.**

**Input : test\_tuple1 = (7, 2), test\_tuple2 = (7, 8)**

**Output : [(7, 7), (7, 8), (2, 7), (2, 8), (7, 7), (7, 2), (8, 7), (8, 2)]**

**from itertools import chain, product**

**# tuples**

**tuple1 = (7, 2)**

**tuple2 = (7, 8)**

**# original tuples**

**print("The tuple 1 : " + str(tuple1))**

**print("The tuple 2 : " + str(tuple2))**

**# All pair combinations of 2 tuples**

**result = list(chain(product(tuple1, tuple2), product(tuple2, tuple1)))**

**# result**

**print("The resultant tuple : " + str(result))**

**Q95. Write a Python program to sort a list of tuples by second item.**

**Input : [('for', 24), ('Geeks', 8), ('Geeks', 30)]**

**Output : [('Geeks', 8), ('for', 24), ('Geeks', 30)]**

**#tuple list**

**tuple = [('for', 24), ('Geeks', 8), ('Geeks', 30)]**

**print("Orignal Tuple List :" ,tuple)**

**#function**

**def Sort(tuple):**

**# Sorts in Ascending order**

**tuple.sort(key = lambda a: a[1])**

**return tuple**

**# printing the sorted list of tuples**

**print("Sorted Tuple List:" ,Sort(tuple))**

**Q96. Write a python program to print below pattern.**

**\***

**\* \***

**\* \* \***

**\* \* \* \***

**\* \* \* \* \***

**# number of rows**

**rows = 5**

**for i in range(0, rows):**

**# nested loop for each column**

**for j in range(0, i + 1):**

**# print star**

**print("\*", end=' ')**

**# new line after each row**

**print("\r")**

**Q97. Write a python program to print below pattern.**

**\***

**\*\***

**\*\*\***

**\*\*\*\***

**\*\*\*\*\***

**Q98. Write a python program to print below pattern.**

**\***

**\* \***

**\* \* \***

**\* \* \* \***

**\* \* \* \* \***

**def triangle(n):**

**# number of spaces**

**k = n - 1**

**# outer loop to handle number of rows**

**for i in range(0, n):**

**# inner loop to handle number spaces**

**# values changing acc. to requirement**

**for j in range(0, k):**

**print(end=" ")**

**# decrementing k after each loop**

**k = k - 1**

**# inner loop to handle number of columns**

**# values changing acc. to outer loop**

**for j in range(0, i+1):**

**# printing stars**

**print("\* ", end="")**

**# ending line after each row**

**print("\r")**

**# Driver Code**

**n = 5**

**triangle(n)**

**Q99. Write a python program to print below pattern.**

**1**

**1 2**

**1 2 3**

**1 2 3 4**

**1 2 3 4 5**

**rows = 5**

**for i in range(1, rows + 1):**

**for j in range(1, i + 1):**

**print(j, end=' ')**

**print('')**

**Q100. Write a python program to print below pattern.**

**A**

**B B**

**C C C**

**D D D D**

**E E E E E**

def alphapat(n):

# initializing value corresponding to 'A'

# ASCII value

num = 65

# outer loop to handle number of rows

# 5 in this case

for i in range(0, n):

# inner loop to handle number of columns

# values changing acc. to outer loop

for j in range(0, i+1):

# explicitly converting to char

ch = chr(num)

# printing char value

print(ch, end=" ")

# incrementing number

num = num + 1

# ending line after each row

print("\r")

# Driver Code

n = 5

alphapat(n)