The Super 50 Questions of Python made by Shobhit

Q1. What is python and who had created.
Python is a general purpose, interpreted, high level programming language which supports procedural, functional and object oriented programming

Guido Van Rossum created Python

02 What are tokens
Tokens are the small valid units of any programming language. Python supports 4 types of tokens: Keywords, Identifiers, Literals and Operators.

Q3. What are keywords
Python reserves a set of 33 keywords that provides special language functionality. The reserved word cannot be used as variable names, function names or identifiers.

Q4. What are identifiers
Identifiers represent the programmable entities such as user-defined names, variables, modules, and other objects.

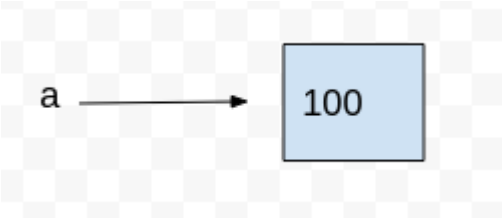
An identifier can be a sequence of lower case, upper case, integers or a combination of any. The identifier name should start with the lowercase or uppercase but must not start with digits). The identifier name should not be a reserved word, and, only underscore (_) is allowed to use as a special character in identifier names.

The length of the identifier name should not be more than 79 characters.

Q5. What is variable and constant

As the name suggests, **Variable** means which can vary or which is not fixed. Variable is the first identifier which we encounter while starting coding. When we write **a = 100,** we perform variable assignment. A constant value 100 is assigned to a variable **a.** Technically, it means that a black box is created somewhere in memory-lane, and this box is called **Object.** It holds the value 100 (a constant). This object is accessed by a reference, called **Variable.**

A constant is a type of variable that holds values, which are immutable (cannot be changed). Constants are declared and assigned to a variable, like 100 above.



Once created, **a** can be used in a statement or expression, and its value will be substituted and later, if the value of **a** is changed and used again, the new value will be substituted instead. That's why it's called a Variable.

Q6. Rules for writing or assigning a variable name

Variable names in Python can be of any length (**without space**), can consist of uppercase and lowercase letters (A-Z, a-z), digits (0-9), and the underscore character (_). Most important rule, although a variable name can contain digits, the first character of a variable name cannot be a digit. Special symbols such as **!, @, #, $, % etc.** cannot be used. Case sensitivity matters. Python is case- sensitive. The commonly used methods of writing a multi-word variable name are:

- **Camel Case:** Second and subsequent words are capitalized, to make word boundaries easier to see.
    - Example: pythonProgrammingUniverse
- **Pascal Case:** Identical to Camel Case, except the first word is also capitalised.
    - Example: PythonProgrammingUniverse
- **Snake Case:** Words are separated by underscores.
    - Example: python_programming_universe

So, which case to use? No hard and fast rule. Depending on your need, you can choose any case. OK, before moving on, let's type our first built- in function. Let's assign a variable and see how to display it and check it.

```
a = 100
print(a)
print(type(a))
```

```
100
<class 'int'>
```

Q7. Give the various data types of python
Numbers: Python supports 4 types of numeric data.

- int (signed integers like 10, 2, 29, etc.)
- long (long integers used for a higher range of values like 908090800L, -0x1929292L, etc.)
- float (float is used to store floating point numbers like 1.9, 9.902, 15.2, etc.)
- complex (complex numbers like 2.14j, 2.0 + 2.3j, etc.)

String: The string is a sequence of characters represented in the quotation marks.

List: Lists are like arrays in C. The list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

Tuple: A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

Dictionary: Dictionary is an ordered set of a key-value pair of items.

There is nothing to worry about. List, Tuple, Dictionary and String will later be dealt in detail. For the time being, read it and just keep the name stored in your memory. We will use **type ()** more frequently when we will study about these later.

Q8. What are literals

Literals are defined as data that is given in a variable or constant. Python has 6 literals: String, Numeric, Boolean, Collection, Set and Dictionaries.

Q9. What is the difference between statically typed language and dynamically typed language

Python automatically understands that it is an integer. We don't have to mention or write like **int i = 5.** This is termed as **Dynamically Typed Language.** If you are familiar with languages like C, C++ or Java, we have to write like this, **int i = 5,** and hence they are called **Statically Typed Languages.**

Means **dynamically typed** language automatically understand the datatype of the variable without explicit intervention by user. Like for python we write a=5 and for JavaScript we write var a = "hello".

In case of **Statically typed** language it is needed to mention the data type by user before declaring the variable. Languages like C, C++, Java, etc are such type of language.

Q10. What is type conversion

The process of converting one datatype to another is called **Type Conversion** or **Type Casting.**

Q11. What are the various operators in python

Python divides the operators in the following 7 groups:

- **Arithmetic operators**
Operator	Meaning	Example
+	Add two operands or unary plus	x+y
-	Subtract right operand from the left or unary minus	x-y
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)
- **Assignment operators**

Operator	Example	Equivatent to
=	x = 5	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
**=	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**

Operator	Meaning	Example
>	Greater that - True if left operand is greater than the right	x > y
<	Less that - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

- **Logical operators**

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

- **Membership operators**

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

- **Bitwise operators**

Operator	Meaning	Example
&	Bitwise AND	x& y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x>> 2 = 2 (0000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000)

- **Augmented Assignment Operator**

Augmented Assignment		Standard Assignment
`a += 5`	is equivalent to	`a = a + 5`
`a /= 10`	is equivalent to	`a = a / 10`
`a ^= b`	is equivalent to	`a = a ^ b`

Q12. What is the difference between identity and equality operator.
Identity operators determine whether the given operands have the same identity—that is, if they refer to the same object. This operator is different from the **equality operator**, which means the two operands refer to objects that contain the same data but are not necessarily the same object.

Q13. What is operator precedence
By operator precedence it means which operator should be evaluated first. When two operators share an operand, the operator with the higher *precedence* will operate first. Python follows the standard precedence rule of **PEMDAS (**Parenthesis- Exponents- Multiplication- Division- Addition- Subtraction**)**, with **Parenthesis** having the highest and **Subtraction** having the lowest priority.

Q14. What are escape characters in python
In Python strings, the backslash **** is a special character, also called the escape character. It is used in representing certain whitespace characters

\t tab
\n new line
\' single quote
\" double quote

\\ backslash

Q14. What are the various nested conditions for if
If inside if inside if
If - else condition
If - else - if - else
If - elif - else condition

Q15 What is "Multiple Assignment in Single Line" feature
Python provides an interesting one line feature to assign values. If on LHS there are n- variables separated by commas and there are n- values on RHS separated by commas, then, each value will be assigned to each variable respectively position wise.

Q16. What are the features of while - else loop? when it can used ?
When the condition inside while statement is becomes false then else block is executed.

When execution encounters breaks and loop terminates then else part would not execute.

Looks like if(while(conditon)) else (statement)
It can be used to enter the **password** for limited trials means if all the trials (iterations) are done then user prompted that "you had attempted 3 wrong input"

Q17. What is the basic use of for loop
For loop is frequently used to traverse the data structures like list, tuple, or dictionary and to iterate the statements or a part of the program several times.

Means difference comes in traversing in sequence instead of using condition to perform iteration. Mainly for loop use membership operator **"in"** to perform traversing.

Q18. What are the special conditions for for-loop
1. Printing the element in the same line (by using end=" ")
2. Adding the commas between the element but no comma at last (use of _sequence[i] with if())
3. Use of range() method where range(a,b) makes a sequence where b is not included.

Q19. What are the famous programs for loop
1. Factorial finding

2. Fabonnici series
3. Finding prime number
4. Sum of digits
5. Armstrong number
6. Sum of digits
7. Astrics patterns
8. Number patters

Q21. Give the features of List

1. Contains elements of different data types
2. Declared with [ ]
3. Can perform slicing
4. Mutable
5. Multi dimensional list allowed
6. Order matters
7. Perform the nesting
8.

Q20. Give the features of Tuples

1. Can contain element of different date types (generally same type is preferred)
2. Declared with ( ) <it is optional>
3. Can perform slicing
4. Immutable
5. Order matters

Q20. Give the features of Set

1. Can contain the elements of same data type
2. Declared with { }
3. Duplicate value automatically eliminated
4. Order not matters
5. Immutable

Q20. Give the features of String

1. Immutable
2. Ordered
3. Enclosed in ‘ ‘ / “ “ / ‘’’ ‘’’ / “”” “””
4. Can perform slicing

Q20. Give the features of Dictionaries

1. Dictionaries are unordered
2. Declared with { “key”:”value”}

3.

Q21. What are modules in python

modules are special libraries holding special functions. To use those special functions we need to import them via their modules.

Q22. What is the difference between shallow copy and deep copy

We can see that original list is modified, as well as new list is updated. This copying method is called **Shallow Copy**. What if we want that any change in the updated list should not modify the original list? This method of copying is called **Deep Copying**.

Q23. Give an example showing values enclosed in parenthesis ( ) ara not always needed to be tuples

```
>>> new_name = ("Robert Downey Jr.")
>>> type(new_name)
<class 'str'>
```

But if we want a tuple having single element, then we can insert comma " , " in following form.

```
>>> new_name = ("Robert Downey Jr.",)
>>> type(new_name)
<class 'tuple'>
```

Q24. What is tuple packing and unpacking

Tuple packing simply means assign the values (which is tuples) to the variable. We know values (tuples) are right side and variable in at left side of " = " sign.

Example of tuple packing

```
>>> avengers = ('Tony', 'Steve', 'Natasha', 'Bruce')
>>> avengers
('Tony', 'Steve', 'Natasha', 'Bruce')
>>> type(avengers)
<class 'tuple'>
```

Unpacking generally means assigning multiple values to multiple variables at the same time. Example of pure unpacking is following

```
a, b = 4, 5
print(a, b)
```

```
4 5
```

One more condition for unpacking is following. You can remember the condition when we used * (asterisk) to manage when there was less variable on the left side of = .

```
*a, b, c= 4, 5, 6, 7, 8, 9, 10
print(a, b, c)
```

```
[4, 5, 6, 7, 8] 9 10
```

Example of tuple unpacking

```
>>> (Iron_Man, Captain_America, Black_Widow, Hulk) = avengers
>>> Iron_Man
'Tony'
>>> Captain_America
'Steve'
>>> Black_Widow
'Natasha'
>>> Hulk
'Bruce'
```

Note Application of tuple unpacking comes when we return more then one values from a function.

```
import math
def circle(rad):
    cir = 2 * math.pi * rad
    area = math.pi * rad * rad
    return (cir, area)

rad = int(input("Enter the radius value:"))
print("The circumference and area are:",circle(rad))
```

Q25. Name some functions and methods of List [ ]

A function is something that you can apply on a construct and get a result, while a method is what you can do to it and change it. Following are the built-in python list methods.

Functions ()

1. **len ()**
2. **max ()**
3. **min ()**
4. **sum ()**
5. **sorted ()**
6. **list ()**
7. **any ()**
8. **all ()**

Methods

1. **append ()**
2. **extend ()**
3. **insert ()**
4. **remove ()**
5. **pop ()**

6. **clear ()**
7. **index ()**
8. **count ()**
9. **sort() and sorted ()**
10. **reverse ()**

Q26. What is enumerate function

```
heros = ['Wade','The Thing','Mr. Fantastic']
for x, y in enumerate (heros, 2):
    print(x, y)
```

```
2 Wade
3 The Thing
4 Mr. Fantastic
```

Enumerate function enables two variables to iterate over a List [ ] such that one variable stores the element value while other variable stores corresponding number (index by default), we can start counting of number from any desired value by providing one more parameter to enumerate function.(default starts from 0)

Q27. What is dictionary

Dictionary is a data type which has a **key** and a **value**.The value is associated with its key. Unlike strings, lists and tuples which are sequence types, because they are ordered and index is used to access the values,but in a Dictionary, **key is used to access the values**. Keys are mapped to the values and hence, sometimes, Dictionaries are also termed as **Mapped Type**. The keys are immutable. It can be int or string but cannot be a list.

Q28. What are the various operations performed on dictionary

1. Creating empty dictionary
2. Entering value in dictionary
3. Printing whole dictionary
4. Accessing values using keys
5. Del for deleting
6. Inserting multiple keys having same value
7. Count no of element using len()
8.

Q29. What if two elements of dictionary having same keys

During declaration of dictionary if we provide 2 elements such both have same key then the last one will be included.

```
>>> avengers = {1:'Tony', 2:'Steve', 3:'Thanos', 3:'Thor'}
>>> avengers
{1: 'Tony', 2: 'Steve', 3: 'Thor'}
```

Q30. What are some of dictionary methods

1. keys()
2. values()
3. items()
4. In method (same membership operator)
5. get()
6. fromkeys()
7. update({ })
8. pop()
9. sorted(dict.keys())
10. sorted(dict.items())
11.

Q31 How defaultdict() comes, what is KeyError what is its use

From the collections module. It is used to prevent the **KeyError** when requested key is not found. It prompts default value. Using defaultdict() is little bit tricky. Just do this step before entering the element to dictionary by

From collections import defaultdict

_dict_name = defaultdict('lambda' : "_your_default_value ");

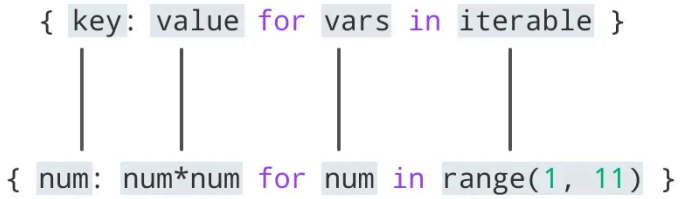
Q32 How to convert list into dictionary

Simply we need two lists. We need to join them through zip() and convert them into dictionary through dict().

```
>>>
>>> non_hero_name = ['Tony', 'Steve', 'Bruce']
>>> hero_name = ['IronMan', 'Captain America', 'Hulk']
>>> avengers = dict(zip(non_hero_name,hero_name))
>>> avengers
{'Tony': 'IronMan', 'Steve': 'Captain America', 'Bruce': 'Hulk'
}
>>>
```

Q33 What is dictionary comprehension

Dictionary comprehension comes when we want to create a dictionary via several iterations (using for loop) but all this is done in a single line.



```
names = ['Tony','Steve','Bruce']
heros = ['Iron Man','Captain America','Hulk']
dict_com = {reel_name:super_hero_name for reel_name, super_hero_name in zip(names,heros)}
print(dict_com)
```

```
{'Tony': 'Iron Man', 'Steve': 'Captain America', 'Bruce': 'Hulk'}
```

Dictionary Comprehension is little bit tricky to understand but ask several time in interviews.

Q34 Give some set functions

1. set()

Q35 What is the basic difference between set and list

In set, there is no repetition of element nor ordering matters while in case of list duplicates can exist and the elements have corresponding indexes (ordering)

Q36. What can be various operations done on set

1. Finding the length
2. Using membership operator
3. union() or **|**
4. intersection() or **&**
5. difference() or **-**
6. symmetric_difference() or **^**
7. isdisjoint()
8. issubset() or **<=**
9. Proper subset() or **<**
10. issuperset() or **>=**
11. Proper superset() or **>**
12. add()
13. remove() and discard() //discard for preventing KeyError if element not there
14. pop()
15. clear()
16.

Q37. Does union and " | " the same thing?

There is a slight difference in method and operator function. The **union ()** method converts the argument into a set and then performs the union operation and can perform the union in more than two sets. Whereas " | " is not able to do so.

Q38. What is Augmented Assignment Operators and Methods Ignore this question if you wish.

Union, intersection, difference, and symmetric difference operators mentioned above have an augmented assignment form and they can be used to modify a set. There is a method for each augmented assignment operator as well.

a.update (b) or **a |= b :** This will add those elements of **b** to **a**, which are not present in set **a**, and then update the set **a**.

a.intersection_update(b) or **a &= b [&c…] :** This will retain only those elements of **b** and **a**, which are common in both the sets and then update the set **a**.

a.difference_update(b) or **a -= b [|c…] :** This will remove those elements of **b** which are common with **a**, and then update the set **a**.

a.symmetric_difference_update(b) or **a ^= b [|c…] :** This will retain those elements which are either present in the set **a** or **b**, but not in both the sets, and then update the set **a**.
In simple language we perform the operation between two sets and the result is stored in either of those two set (mostly left ones)

Q39. What is Frozen Sets

As the name suggests, sets which are immutable or cannot change are called frozen sets. To make a set frozen, the **frozenset** command is used. It is made through frozenset().

```
a = frozenset(['Tony','Bruce','Peter'])
print(a)
print(type(a))
```

```
frozenset({'Tony', 'Bruce', 'Peter'})
<class 'frozenset'>
```

Q40. Give a syntax of set Comprehension

_set_name = { y(x) for x in _sequence_name }

```
num = [4,3,2,1,5,4,3,2,1,5]
set_com = {n*n for n in num}
print(set_com)
```

```
{1, 4, 9, 16, 25}
```

Q41. What is string in python? What are the ways to create a string? Why are there so many ways to create the string?

Python string is a sequence of **Unicode characters**. It has a built-in class **"str"** to handle string operation. Even a **single** character is a string.

```
>>> avengers = 'I am Iron Man'
>>> type(avengers)
<class 'str'>
>>> avengers = "I am Iron Man"
>>> type(avengers)
<class 'str'>
>>> avengers = '''I am Iron Man'''
>>> type(avengers)
<class 'str'>
```

It might happen that we want to display the text which already contains the quotes. Further if we use the same quote style to make the string then those quotes clash and the result may be unexpected. Thus there are many styles to enclose the string so that we can use desired style quotes in our text without clashing.

Q42. What are the other uses of triple quotes?

1. Using multiple lines to create string
2. Multi line comment

Q43. Give the names of some String Functions

1. **len ()**
2. **isdigit ()**
3. **isalnum ()**
4. **isalpha ()**
5. **upper ()**
6. **lower ()**
7. **isspace ()**
8. **startswith ()**
9. **endswith ()**
10. **find ()**
11. **replace ()**
12. **split ()**
13. **join ()**
14. **strip ()**
15. **swapcase ()**
16. **title ()**
17. **capitalize()**
18. **ord ()**
19. **chr ()**
20. **isprintable ()**
21. **format()**
22. **zfill()**
23. **center ()**
24. **rjust ()**
25. **rstrip ()**

26. **ljust ()**
27. **lstrip ()**
28. **partition (<sep>)**
29. **rpartition (<sep>)**
30. **rsplit (sep = None, maxsplit = -1)**
31. **split (sep = None, maxsplit = -1)**
32. **splitlines ([<keepends>])**
33.

Q44. What is the use of space and * for string concatenation.
Space can be used for string concatenation like following

```
avenger = 'Tony' ' Stark'
print(avenger)
```

```
Tony Stark
```

Asterisk (*) can be used concatenate the string by its repetition.

```
>>> my_data = '10'
>>> my_data * 2
'1010'
```

Q45. What is function in python
Functions are **blocks of codes** which execute when it is called by its name. Data is assigned into a function as **parameters**, which is executed as **instructed** in the body of the function and **returns** data as result. Functions are used when a set of instructions has to be used **again and again** repeatedly.

Q46. What are the types of function in python

- Built-in functions, such as help (), min (), print (), etc..
- User-Defined Functions (UDFs), made by users.
- Anonymous functions, also called lambda functions which are not declared with the standard **def** keyword and it also does not return any value.

Q47. What is method in python
A method refers to a function which is **part of a class**. You can access it with an **instance** or **object** of the class. All methods **are functions**, but **not all functions** are methods.

Q48. What is the difference between Argument and Parameter
Argument are the values which are provided to the function at the instant when the function is called.
Parameters are the local variables declared at function header or function signature which will be executed in the function body.

Q49. What is First Class Function and Higher Order Function

First Class Function When a function is assigned to a variable and that variable performs all the operation of the corresponding function, it is known as First Class Function.

```
def sq(n):
    return n * n
f = sq
print(sq)
print(f(8))
```

```
<function sq at 0x05097780>
64
```

Higher Order Function is a function which uses another function as argument and use it in its body and call that function (argument wala) from there.

```
def sq(n):
    return n * n

def my_map_func(my_func, list_of_arg):
    result = []
    for i in list_of_arg:
        result.append(my_func(i))
    return result

squares = my_map_func(sq, [4,3,2,1])
print(squares)
```

```
[16, 9, 4, 1]
```

Q50. What is the logic of finding HCF and LCM

Consider the two number 12 and 36. LCM will come in between the iteration i = 36 to 12*36.(increment will be done in the multiple of 36). In between the iteration i will be LCM if i%12==0;

For finding the HCF we know that HCF*LCM=product of two numbers.

Q51. What are the four types of arguments

- **Default arguments**
  - We give default values during function declaration, when no argument is passed then default values is used.
- **Required arguments**
  - These are normal arguments whose values are compulsory to pass
- **Keyword arguments**
  - When the order of arguments is not known then arguments are passed along with the names.
- **Variable number of arguments or Arbitrary arguments**

- When no of values passed is not known then we put * with parameter (*name) where it accepts n no of arguments in the form of tuples.
- When we use ** with parameters(**item) then it accepts n no of arguments in the form of dictionaries.

```
def n_values(*a):
    return a
print(n_values(1,2,3))
print(type(n_values()))
```

```
(1, 2, 3)
<class 'tuple'>
```

```
def n_values(**b):
    return b
print(n_values(x=1,y=2,z=3))
print(type(n_values()))
```

```
{'x': 1, 'y': 2, 'z': 3}
<class 'dict'>
```

Note Please regularly practise the various functions of python in order to acquire the girp on python. User Abhishek sir website for reference to practise.