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

Python is an object-oriented, high-level programming language. Object-oriented means this language is based around objects (such as data) rather than functions, and high-level means it's easy for humans to understand.

Q2. Why is Python called a dynamically typed language?

Python is both a strongly typed and a dynamically typed language. Strong typing means that variables do have a type and that the type matters when performing operations on a variable. Dynamic typing means that the type of the variable is determined only during runtime

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

Pros	Cons
Beginner-friendly	Issues with design
Large community	Slower than compiled languages
Flexible and extensible	Security
Embeddable	Work environment
Highly scalable	Python's Memory Consumption and Garbage Collection
Machine learning	Python is dynamically typed

Q4. In what all domains can we use Python?
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 variables and how can we declare them?

Variable: 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. But the data itself is still contained within the object.
Python has no command for declaring a variable. A variable is created the moment you first assign a value to it.

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

Python user input from the keyboard can be read using the input() built-in function. The input from the user is read as a string and can be assigned to a variable. After entering the value from the keyboard, we have to press the “Enter” button. Then the input() function reads the value entered by the user.

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

By default, input returns a string. So the name and age will be stored as strings.

Q8. What is typecasting?

Type Casting is the method to convert the variable data type into a certain data type in order to perform the operation required to be performed by users.

Q9. Can we take more than one input from the user using a single input() function? If yes, how? If not, why?

Yes. We can use split(), input(), and map() functions to take more than one input from the user using a single input() function.

Q10. What are keywords?

Python keywords are special reserved words that have specific meanings and purposes and can't be used for anything but those specific purposes. These keywords are always available—you'll never have to import them into your code.

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

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, or variable name

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

Indentation refers to the spaces at the beginning of a code line. Whereas in other programming languages the indentation in code is for readability only, the indentation in Python is very important. Python uses indentation to indicate a block of code.
Uses:
Used to help to make the code look pretty
Used to create a group of statements

Q13. How can we throw some output in Python?

The basic way to do output is the print statement. To end the printed line with a newline, add a print statement without any objects. This will print to any object that implements write(), which includes file objects.

Q14. What are operators in Python?

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. A sequence of operands and operators, like a + b - 5 , is called an expression. Python supports many operators for combining data objects into expressions.

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

/ operator is float division
// is an integer division or floor division.
Float Division("/"): Divides left-hand operand by right-hand operand. The division works in Python the way it's mathematically defined.
Floor Division("//"): The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored , i.e., rounded away from zero (towards negative infinity).

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

```
print('iNeuron' *4)
```

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

```
number = int(input("Enter the number"))
if number%2 == 0:
    print( number, 'is even')

else:
    print( number ,'is odd')
```

:

Q18. What are boolean operators?

The Python Boolean type is one of Python's built-in data types. It's used to represent the true 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 be?
1 or 0

0 and 0

True and False and True

1 or 0 or 0

```
1 or 0

0 and 0

True and False and True

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.

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

if statement
If the condition following the keyword is evaluated as true, the block of code will execute. Note that parentheses are not used before and after the condition check as in other languages.

```
x = 5

if x > 4:
  print("The condition was true!") #this statement executes
```

else statement
You can optionally add an else response that will execute if the condition is false

```
if not True:
  print('If statement will execute!')
else:
  print('Else statement will execute!')
```

elif statement
Multiple conditions can be checked by including one or more elif checks after your initial if statement. Just keep in mind that only one condition will execute

```
z = 7

if z > 8:
  print("I won't print!") #this statement does not execute
elif z > 5:
  print("I will!") #this statement will execute
elif z > 6:
  print("I also won't print!") #this statement does not execute
else:
  print("Neither will I!") #this statement does not execute
```

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

```
age = int(input('Enter your age : '))
if age>= 18:
    print(' I can vote')

else:
    print('I cant 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]

[Preview] README.md main.py M ×

main.py > ...

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

def even_sum(numbers):
    result = 0
    for x in numbers:
        if x % 2 == 0:
            result += x
    return result

print(even_sum(numbers))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
392
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
a=int(input("enter a number: "))
b=int(input("enter a number: "))
c=int(input("enter a number: "))
if a>b and a>c:
    print("a is the greatest number")
elif b>a and b>c:
    print("b is the greatest number")
elif c>a and c>b:
    print("c is the greatest number")
else:
    print("all the numbers are equal")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
enter a number: 5
enter a number: 7
enter a number: 8
c is the greatest number
abc@7db98ad07b3a:~/workspace$
```

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]

```
[Preview] README.md    main.py M ×
main.py > i
lst = [12, 75, 150, 180, 145, 525, 50]

for i in lst:
    if i%5==0 and i<=500:
        print(i)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
75
150
180
145
50
abc@7db98ad07b3a:~/workspace$
```

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

Like many other popular programming languages, 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. Square brackets can be used to access elements of the string.

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

You can access the characters in a string by referring to its index number inside square brackets

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

```
[5]: string = "Big Data iNeuron"

[7]: string.split(' ')[-1]

[7]: 'iNeuron'
```

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

```
[5]: string = "Big Data iNeuron"

[8]: string.split(' ')[-1][::-1]

[8]: 'norueNi'
```

Q30. Reverse the string given in the above question.

```
[5]: string = "Big Data iNeuron"

[9]: string[::-1]

[9]: 'norueNi ataD giB'
```

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

In Python, you can use the replace() and translate() methods to specify which characters you want to remove from the string and return a new modified string result. It is important to remember that the original string will not be altered because strings are immutable

Q32. What is an escape sequence?

An escape sequence is a sequence of characters that, when used inside a character or string, does not represent itself but is converted into another character or series of characters.

Q33. How can you print the below string?

```
[10]: txt = print("iNeuron's Big Data Course")

      iNeuron's Big Data Course
```

Q34. What is a list in Python?

List. 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, Set, and Dictionary, all with different qualities and usage.

Q35. How can you create a list in Python?

In Python, a list is created by placing elements inside square brackets [ ], separated by commas. A list can have any number of items and they may be of different types (integer, float, string, etc.). A list can also have another list as an item. This is called a nested list.

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

The syntax for accessing the elements of a list is the same as the syntax for accessing the characters of a string. We use the index operator ([ ] – not to be confused with an empty list). The expression inside the brackets specifies the index. Remember that the indices start at 0.

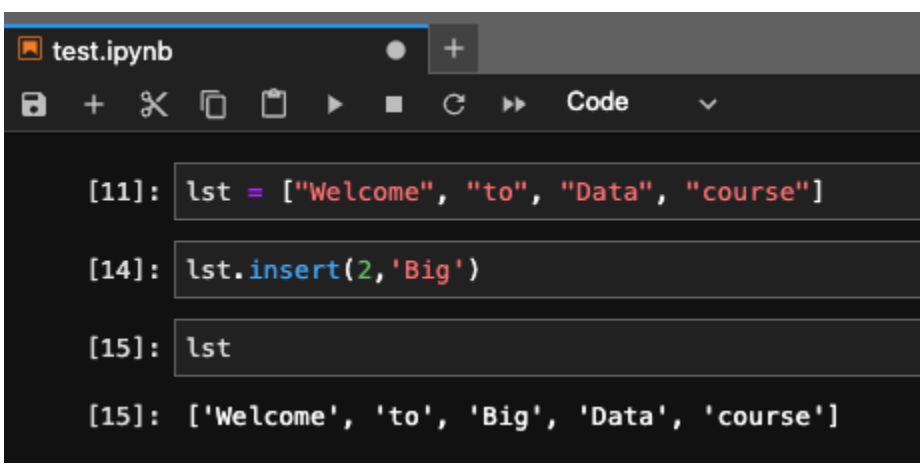
Q37. Write a code to access the word "iNeuron" from the given list.
lst = [1,2,3,"Hi",[45,54, "iNeuron"], "Big Data"]

Answer: lst[4][2]

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

Answer: len(lst)

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



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

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. The contents of a tuple cannot change once they have been created in Python due to the immutability of tuples

Q41. How can you create a tuple in Python?

A tuple is created by placing all the items (elements) inside parentheses () , separated by commas. The parentheses are optional, however, it is a good practice to use them. A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).

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

```
[21]: sample_tup = ('this','is','my','name')

[22]: tup_list = list(sample_tup)

[23]: tup_list.append('Gokul')

[24]: tuple(tup_list)

[24]: ('this', 'is', 'my', 'name', 'Gokul')
```

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

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

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

```
[Preview] README.md    main.py M ×
main.py > ...
1  user = tuple(input("Please enter what ever you want seperated by comma: "))
2  print(len(user))

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
Please enter what ever you want seperated by comma: gokul,123,#$%,__
16
abc@7db98ad07b3a:~/workspace$
```

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, Tuple, and Dictionary, all with different qualities and usage. A set is a collection which is unordered, unchangeable*, and unindexed.

Q46. How can you create a set?

A set is created by placing all the items (elements) inside curly braces {} , separated by comma, or by using the built-in set() function. It can have any number of items and they may be of different types (integer, float, tuple, string etc.).

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

```
[25]: my_set = {'I','like'}

[29]: my_set.add("iNeuron")

[27]: my_set

[27]: {'I', 'iNeuron', 'like'}
```

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

```
[31]: my_set = {'I','like'}

[32]: my_set.add("iNeuron","Krish","banglore")

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [32], line 1
----> 1 my_set.add("iNeuron","Krish","banglore")

TypeError: set.add() takes exactly one argument (3 given)

[34]: lis = ("iNeuron","Krish","banglore")

[36]: my_set.update(lis)

[37]: my_set

[37]: {'I', 'Krish', 'banglore', 'iNeuron', 'like'}
```

Q49. How is update() different from add()?
We use the add() method to add a single value to a set. We use update() method to add sequence values to a set

Q50. What is clear() in sets?
The clear() method removes all elements from a Set object.

Q51. What is a frozen set?

Frozenset is similar to set in Python, except that frozensets are immutable, which implies that once generated, elements from the frozenset cannot be added or removed. This function accepts any iterable object as input and transforms it into an immutable object.

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

Frozen set is an immutable version of set.

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

The Python set union() method returns a new set with distinct elements from all the sets.

Code:

```
main.py M ×
main.py > ...
A = {2, 3, 5}
B = {1, 3, 5}

# compute union between A and B
print('A U B = ', A.union(B))


# Output: A U B =  {1, 2, 3, 5}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

abc@56bc1a6ba0af:~/workspace$ /bin/python /config/workspace/main.py
A U B =  {1, 2, 3, 5}
abc@56bc1a6ba0af:~/workspace$
```

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

The intersection() method returns a new set with elements that are common to all sets.

```python
A = {2, 3, 5}
B = {1, 3, 5}

# compute intersection between A and B
print(A.intersection(B))


# Output: {3, 5}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@56bc1a6ba0af:~/workspace$ /bin/python /config/workspace/main.py
{3, 5}
abc@56bc1a6ba0af:~/workspace$
```

Q55. What is a dictionary in Python?

Dictionaries are Python's implementation of a data structure that is more generally known as an associative array. A dictionary consists of a collection of key-value pairs. Each key-value pair maps the key to its associated value.

Q57. How can we declare a dictionary in Python?

```python
# Empty dictionary
dict1 = dict()
dict2 = {}

# Dictionary with elements
dict3 = {'name':'Vivek', 'age':23, 'city':'Pune'}
```

Q58. What will be the output of the following?

```
main.py M ×
main.py > ...
var = {}
print(type(var))

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

abc@56bc1a6ba0af:~/workspace$ /bin/python /config/workspace/main.py
<class 'dict'>
abc@56bc1a6ba0af:~/workspace$
```

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

We can make use of the built-in function append() to add elements to the keys in the dictionary. To add an element using append() to the dictionary, we have first to find the key to which we need to append to.

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

```
main.py M ×
main.py > key
my_dictionary = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

for key, value in my_dictionary.items():
    print(key, value)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

abc@56bc1a6ba0af:~/workspace$ /bin/python /config/workspace/main.py
name John
age 30
city New York
abc@56bc1a6ba0af:~/workspace$
```

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

```python
d = {'dict1': {'a': 1, 'b': 2, 'c': 3}, 'dict2': {'d': 4, 'e': 5, 'f': 6}}

for key1, val1 in d.items():
    for key2, val2 in val1.items():
        print(key1, key2, val2)
```

```
abc@56bc1a6ba0af:~/workspace$ /bin/python /config/workspace/main.py
dict1 a 1
dict1 b 2
dict1 c 3
dict2 d 4
dict2 e 5
dict2 f 6
abc@56bc1a6ba0af:~/workspace$
```

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

The get() method returns the value of the item with the specified key.

Q63. What is the use of the 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.

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

The pop() method removes the item at the given index from the list and returns the removed item.

main.py M ×

main.py > ...

```python
# create a list of prime numbers
prime_numbers = [2, 3, 5, 7]

# remove the element at index 2
removed_element = prime_numbers.pop(2)

print('Removed Element:', removed_element)
print('Updated List:', prime_numbers)

# Output:
# Removed Element: 5
# Updated List: [2, 3, 7]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@56bc1a6ba0af:~/workspace$ /bin/python /config/workspace/main.py
Removed Element: 5
Updated List: [2, 3, 7]
abc@56bc1a6ba0af:~/workspace$
```

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

Python dictionary popitem() method removes the last inserted key-value pair from the dictionary and returns it as a tuple.

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.

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.

Q68. What are loops in Python?

Loops in Python are used to execute a block of code repeatedly until the condition is True.

Q69. How many types of loop are there in Python?

There are two types of loops in Python, for and while.

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

The for and while loops are both conditional statements. A for loop is a single-line command that will be executed repeatedly. While loops can be single-lined or contain multiple commands for a single condition. Both the for loop and the while loop are important in computer languages for obtaining results.

Q71. What is the use of the 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.

Q72. What is the use of a break statement?

'Break' in Python is a loop control statement. It is used to control the sequence of the loop. Suppose you want to terminate a loop and skip to the next code after the loop; break will help you do that. A typical scenario of using the Break in Python is when an external condition triggers the loop's termination.

Q73. What is the use of a 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.

Q74. What is the use of the 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.

Q75. How can you loop over a dictionary?

You can loop through a dictionary by using a for loop. When looping through a dictionary, the return values are the keys of the dictionary, but there are methods to return the values as well.

```
main.py M ×
main.py > k
dict1 = {'name':'Gokul', 'age':23, 'city':'CBE', 'skills': {'language':'Python', 'database':'MySQL'}}
for k,v in dict1.items():
    print('Key:', k, 'Value:', v)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

abc@4a4a2a95a5f3:~/workspace$ /bin/python /config/workspace/main.py
Key: name Value: Gokul
Key: age Value: 23
Key: city Value: CBE
Key: skills Value: {'language': 'Python', 'database': 'MySQL'}
abc@4a4a2a95a5f3:~/workspace$
```

CODING PROBLEMS

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

```
main.py M ×
main.py > ...
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
    print(" Factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1,num + 1):
        factorial = factorial*i
    print("The factorial of",num,"is",factorial)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

abc@95bf7581873c:~/workspace$ /bin/python /config/workspace/main.py
Enter a number: 2
The factorial of 2 is 2
abc@95bf7581873c:~/workspace$
```

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

```python
def simple_interest(p,t,r):
    print('The principal is', p)
    print('The time period is', t)
    print('The rate of interest is',r)

    si = (p * t * r)/100

    print('The Simple Interest is', si)
    return si

simple_interest(10, 6, 8)
```

```
abc@95bf7581873c:~/workspace$ /bin/python /config/workspace/main.py
abc@95bf7581873c:~/workspace$ /bin/python /config/workspace/main.py
The principal is 10
The time period is 6
The rate of interest is 8
The Simple Interest is 4.8
abc@95bf7581873c:~/workspace$
```

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

```python
def compound_interest(principle, rate, time):

    # Calculates compound interest
    Amount = principle * (pow((1 + rate / 100), time))
    CI = Amount - principle
    print("Compound interest is", CI)

compound_interest(10000, 10.25, 5)
```

```
abc@95bf7581873c:~/workspace$ /bin/python /config/workspace/main.py
Compound interest is 6288.946267774416
abc@95bf7581873c:~/workspace$
```

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

```python
num = int(input("Please enter a number: "))
if num > 1:
    for i in range(2, int(num/2)+1):
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

```
abc@95bf7581873c:~/workspace$ /bin/python /config/workspace/main.py
Please enter a number: 3
3 is a prime number
abc@95bf7581873c:~/workspace$ /bin/python /config/workspace/main.py
Please enter a number: 4
4 is not a prime number
abc@95bf7581873c:~/workspace$
```

Q80. Write a Python program to check Armstrong Number.

```python
num = int(input("Enter a number: "))

sum = 0

temp = num
while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

if num == sum:
    print(num,"is an Armstrong number")
else:
    print(num,"is not an Armstrong number")
```

```
abc@95bf7581873c:~/workspace$ /bin/python /config/workspace/main.py
Enter a number: 663
663 is not an Armstrong number
abc@95bf7581873c:~/workspace$
```

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

main.py M ×

main.py > ...

```python
def Fibonacci(n):
    if n<= 0:
        print("Incorrect input")
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)

print(Fibonacci(10))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
34
abc@859679fdeb61:~/workspace$
```

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

main.py M ×

main.py > ...

```python
def swapList(newList):
    size = len(newList)
    temp = newList[0]
    newList[0] = newList[size - 1]
    newList[size - 1] = temp
    return newList

newList = [12, 35, 9, 56, 24]

print(swapList(newList))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
[24, 35, 9, 56, 12]
abc@859679fdeb61:~/workspace$
```

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

```python
def swapPositions(list, pos1, pos2):

    list[pos1], list[pos2] = list[pos2], list[pos1]
    return list

List = [23, 65, 19, 90]
pos1, pos2  = 1, 3

print(swapPositions(List, pos1-1, pos2-1))
```

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
[19, 65, 23, 90]
abc@859679fdeb61:~/workspace$
```

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

```python
def largenum(list):
    list.sort()
    print("The largest num in the list is: ", list[-1])

list = [1,2,3,4,5]

largenum(list)
```

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
The largest num in the list is:  5
abc@859679fdeb61:~/workspace$
```

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

```python
def sumlist(list):
    sumed_list = sum(list)
    print("The sum of your list is: ", sumed_list)

list = [1,2,3,4,5]

sumlist(list)
```

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
The sum of your list is:  15
abc@859679fdeb61:~/workspace$
```

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

```python
text = input("Please enter your word to check is that's a palindrome: ")

def pal(text):
    if text[::-1] == text:
        print("The word you have entered is a palindrome!!!")
    else:
        print("This is not a palindrome :(")

pal(text)
```

```
abc@859679fdeb61:~/workspace$
```

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

```python
def remove_char(s, i):
    a = s[ : i]
    b = s[i + 1: ]

    return a+b

string = "MyNameIsGokul"
i = 5
print(remove_char(string,i-1))
```

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
MyNaeIsGokul
abc@859679fdeb61:~/workspace$
```

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

```python
text = input("Please enter a sentence: ")
text1 = input(("Please enter a word: "))

def sub_check(text, text1):
    if text1 in text:
        print("Yes, the word is present in the sentence!!")
    else:
        print("Nope, the word is not present")

sub_check(text, text1)
```

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
Please enter a sentence: Gokul is a good boy
Please enter a word: Gokul
Yes, the word is present in the sentence!!
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
Please enter a sentence: Gokul is a good boy!
Please enter a word: Ragul
Nope, the word is not present
abc@859679fdeb61:~/workspace$
```

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

```python
text = input('Please enter the words you want to check: ')
k = int(input("Please enter the len of the words: "))

def words_greater(text, k):
    w_count = text.split(' ')
    for i in w_count:
        if len(i) > k:
            print(i)
        else:
            pass

words_greater(text, k)
```

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
Please enter the words you want to check: This is my name Gokul
Please enter the len of the words: 4
Gokul
abc@859679fdeb61:~/workspace$
```

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

```python
dictionary = {'Gokul':'Basketball', 'Max': 'F1', 'Lewis': 'Surffing', 'LeBron': 'Basketball'}

def uniq_val(dictionary):
    for val in set(dictionary.values()):
        print(val.title())

uniq_val(dictionary)
```

```
abc@859679fdeb61:~/workspace$ /bin/python /config/workspace/main.py
Basketball
F1
Surffing
abc@859679fdeb61:~/workspace$
```

Q91. Write a Python program to merge two dictionaries.

[Preview] README.md main.py M

main.py > ...

```python

d1 = {'a': 10, 'b': 8}
d2 = {'d': 6, 'c': 4}

d = d1.copy()
d.update(d2)
print(d)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
{'a': 10, 'b': 8, 'd': 6, 'c': 4}
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
lis = [('Sachin', 10), ('MSD', 7), ('Kohli', 18), ('Rohit', 45)]

out = dict(lis)

print(out)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
{'Sachin': 10, 'MSD': 7, 'Kohli': 18, 'Rohit': 45}
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
list = [9, 5, 6]

result = [(x,x**3) for x in list]
print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
[(9, 729), (5, 125), (6, 216)]
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
test_tuple1 = (7, 2)
test_tuple2 = (7, 8)

result = [(x,y) for x in test_tuple1 for y in test_tuple2]
result = result + [(x,y) for x in test_tuple2 for y in test_tuple1]

print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
[(7, 7), (7, 8), (2, 7), (2, 8), (7, 7), (7, 2), (8, 7), (8, 2)]
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
def short_tup(tup):
    tup.sort(key = lambda x: x[1])
    return(tup)

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

print(short_tup(tup))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
[('Geeks', 8), ('for', 24), ('Geeks', 30)]
abc@7db98ad07b3a:~/workspace$
```

Q96. Write a python program to print below pattern.

[Preview] README.md main.py M

main.py > ...

```python
n = int(input("Enter the number of rows: "))
for i in range(0, n):
        for j in range(0, i + 1):
            print("* ", end="")
        print()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
Enter the number of rows: 6
*
* *
* * *
* * * *
* * * * *
* * * * * *
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
rows = int(input("Enter the number of rows: "))
k = 2 * rows - 2
for i in range(0, rows):
    for j in range(0, k):
        print(end=" ")
    k = k - 2
    for j in range(0, i + 1):
        print("* ", end="")
    print("")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
Enter the number of rows: 5
        *
      * *
    * * *
  * * * *
* * * * *
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
n = int(input("Enter the number of rows: "))
m = (2 * n) - 2
for i in range(0, n):
    for j in range(0, m):
        print(end=" ")
    m = m - 1
    for j in range(0, i + 1):
        print("* ", end=' ')
    print(" ")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
Enter the number of rows: 5
        *
       * *
      * * *
     * * * *
    * * * * *
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
rows = int(input("Enter the number of rows: "))
for i in range(rows+1):
    for j in range(i):
        print(i, end=" ")
    print(" ")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
Enter the number of rows: 5

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
abc@7db98ad07b3a:~/workspace$
```

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

[Preview] README.md main.py M

main.py > ...

```python
str1 = "ABCDE"
x = ""
for i in str1:
    x += i
    print(x)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
abc@7db98ad07b3a:~/workspace$ /bin/python /config/workspace/main.py
A
AB
ABC
ABCD
ABCDE
abc@7db98ad07b3a:~/workspace$
```