

1. Write a Python program to calculate the length of a string.

The screenshot shows a Python code editor with the following code:

```
1 s="Leelanjn"
2 print(len(s))
```

The output window shows:

STDIN

Input for the program (Optional)

Output:

9

2. Write a Python program to count the number of characters (character frequency) in a string.

Sample String : google.com

Expected Result : {'o': 3, 'g': 2, '.': 1, 'e': 1, 'l': 1, 'm': 1, 'c': 1}

The screenshot shows a Python code editor with the following code:

```
1 s="google.com"
2 freq={}
3 for i in s:
4     if i in freq:
5         freq[i]+=1
6     else:
7         freq[i]=1
8 print(freq)
```

The output window shows:

STDIN

Input for the program (Optional)

Output:

{'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}

11 ms | 9.6 MB

3. Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string.

Sample String : 'thisisniceone'

Expected Result : 'thne"

Sample String : 'ab'

Expected Result : 'abab'

Sample String : 'f'

Expected Result : Empty String

The screenshot shows a Python code editor with the following code:

```
1 s='thisisniceone'
2 for i in s:
3     if len(s)<2:
4         print("")
5     print(s[:2]+s[-2:])
```

The output window shows:

STDIN

Input for the program (Optional)

Output:

thne

10 ms | 9.6 MB

4. Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '\$', except the first char itself.

Sample String : 'restart'

Expected Result : 'resta\$t'

main.py

```
44bw8q3yn ⚡  
1  
2  
3  
4 s='restart'  
5 first=s[0]  
6 print(first + s[1:].replace(first,"$"))  
7  
8  
9  
10  
11
```

STDIN
Input for the program (Optional)

Output:
resta\$t

5. Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.

Sample String : 'abc', 'xyz'

Expected Result : 'xyc abz'

main.py

```
44bw8q3yn ⚡  
1  
2  
3 s='abc'  
4 f='xyz'  
5 r=f[:2]+s[2:] + " "+s[:2]+f[2:]  
6 print(r)  
7  
8 |  
9  
10  
11
```

STDIN
Input for the program (Optional)

Output:
xyc abz

6. Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

Sample String : 'abc'

Expected Result : 'abcing'

Sample String : 'string'

Expected Result : 'stringly'

main.py

```
44beea2h8 ⚡  
1  
2 s="abc"  
3 if len(s)<3:  
4     print(s)  
5 elif s[-3:]== "ing":  
6     print(s+"ly")  
7 else:  
8     print(s+"ing")  
9  
10  
11
```

STDIN
Input for the program (Optional)

Output:
abcing

10 ms | 9.6 MB

7. Write a Python program to find the first appearance of the substring 'not' and 'poor' from a given string, if 'not' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.

Sample String : 'The lyrics is not that poor!'

'The lyrics is poor!'

Expected Result : 'The lyrics is good!'

'The lyrics is poor!'

The screenshot shows a code editor window with a dark theme. On the left is the code file 'main.py' containing the following Python code:

```
1 def not_poor(s):
2     not_pos = s.find("not")
3     poor_pos = s.find("poor")
4
5     if not_pos != -1 and poor_pos != -1 and not_pos < poor_pos:
6         return s[:not_pos] + "good" + s[poor_pos+4:]
7     return s
8
9 print(not_poor("The lyrics is not that poor!"))
10 print(not_poor("The lyrics is poor!"))
11
```

On the right side of the editor, there are three sections: 'STDIN', 'Input for the program (Optional)', and 'Output'. The 'Output' section displays the results of running the code with the provided sample strings.

8. Write a Python function that takes a list of words and returns the length of the longest one.

The screenshot shows a code editor window with a dark theme. On the left is the code file 'main.py' containing the following Python code:

```
1 def longest_word(words):
2     return max(len(word) for word in words)
3
4 print(longest_word(["Python", "Django", "Programming"]))
5
```

On the right side of the editor, there are three sections: 'STDIN', 'Input for the program (Optional)', and 'Output'. The 'Output' section displays the result of running the code with the provided list of words.

9. Write a Python program to remove the nth index character from a nonempty string.

The screenshot shows a code editor window with a dark theme. On the left is the code file 'main.py' containing the following Python code:

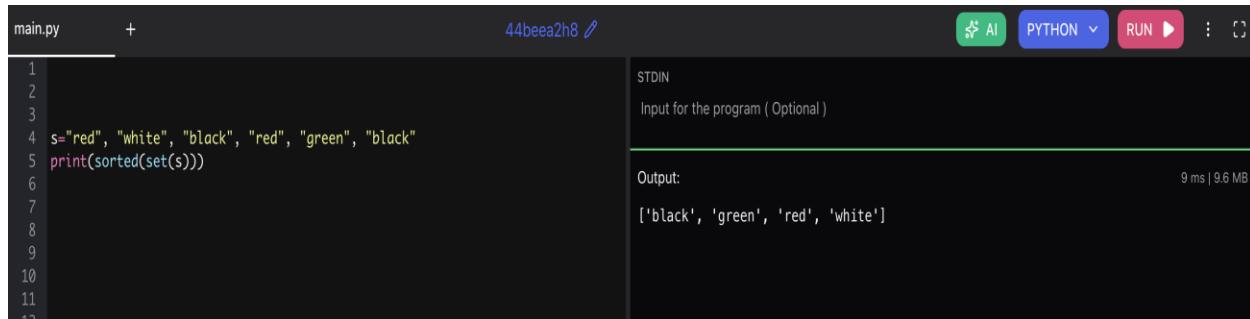
```
1 s = "Python"
2 n = 2
3 print(s[:n] + s[n+1:])
4
```

On the right side of the editor, there are three sections: 'STDIN', 'Input for the program (Optional)', and 'Output'. The 'Output' section displays the result of running the code with the provided string and index value.

10. Write a Python program that accepts a comma separated sequence of words as input and prints the unique words in sorted form (alphanumerically).

Sample Words : red, white, black, red, green, black

Expected Result : black, green, red, white



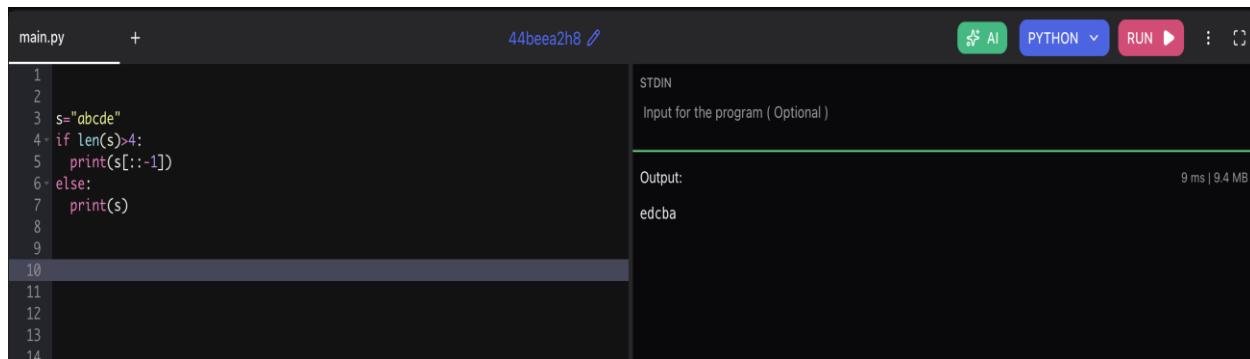
```
main.py + 44beea2h8 ⚡ AI PYTHON RUN ⏪ ⏹
```

```
1
2
3
4 s="red", "white", "black", "red", "green", "black"
5 print(sorted(set(s)))
6
7
8
9
10
11
12
```

STDIN
Input for the program (Optional)

Output: ['black', 'green', 'red', 'white'] 9 ms | 9.6 MB

11. Write a Python function to reverses a string if it's length is a multiple of 4.



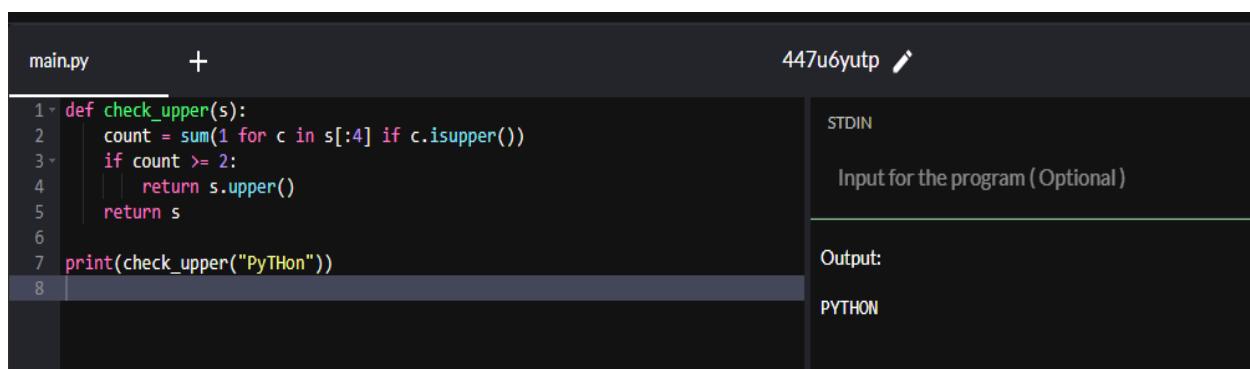
```
main.py + 44beea2h8 ⚡ AI PYTHON RUN ⏪ ⏹
```

```
1
2
3 s="abcde"
4 if len(s)>4:
5     print(s[::-1])
6 else:
7     print(s)
8
9
10
11
12
13
14
```

STDIN
Input for the program (Optional)

Output: edcba 9 ms | 9.4 MB

12. Write a Python function to convert a given string to all uppercase if it contains at least 2 uppercase characters in the first 4 characters.



```
main.py + 447u6yutp 🖊
```

```
1 def check_upper(s):
2     count = sum(1 for c in s[:4] if c.isupper())
3     if count >= 2:
4         return s.upper()
5     return s
6
7 print(check_upper("PyTHON"))
8
```

STDIN
Input for the program (Optional)

Output: PYTHON

13. Write a Python program to check whether a string starts with specified characters.

The screenshot shows a Python code editor interface. The code in `main.py` is:

```
1 s="Python Programming"
2 if s[:6]=="Python":
3     print(True)
4 else:
5     print(False)
6
7
8
9
10
11
```

The status bar at the top right shows the hash `44beea2h8`, an AI icon, a Python dropdown, and a Run button. The output section shows "Output: True" and "9 ms | 9.6 MB".

14. Write a Python program to print the following floating numbers upto 2 decimal places.

3.1415926

The screenshot shows a Python code editor interface. The code in `main.py` is:

```
1 num = 3.1415926
2 print(f"{num:.2f}")
3
```

The status bar at the top right shows the hash `447u6yutp`, an AI icon, a Python dropdown, and a Run button. The output section shows "Output: 3.14" and "9 ms | 9.6 MB".

15. Write a Python program to count repeated characters in a string.

Sample string: 'thequickbrownfoxjumpsoverthelazydog'

Expected output :

o 4
e 3
u 2
h 2
r 2
t 2

The screenshot shows a Python code editor interface. The code in `main.py` is:

```
1
2 s= 'thequickbrownfoxjumpsoverthelazydog'
3 freq={}
4 for i in s:
5     if i in freq:
6         freq[i]+=1
7     else:
8         freq[i]=1
9 for i,j in freq.items():
10    if j>1:
11        print(i,j)
12
13
14
15
16
```

The status bar at the top right shows the hash `44beea2h8`, an AI icon, a Python dropdown, and a Run button. The output section shows "Output: t 2 h 2 e 3 u 2 r 2 o 4" and "9 ms | 9.6 MB".

16. Write a Python program to print the index of the character in a string.

```
main.py + 44bw8q3yn ⚖
1 s="leelajan"
2 for i,j in enumerate(s):
3     print(i,j)
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Output:

```
0 l
1 e
2 e
3 l
4 a
5 n
6 j
7 a
8 n
```

17. Write a Python program to convert a string in a list.

```
main.py + 447u6yutp ⚖
1 s = "Python"
2 print(list(s))
3
```

Output:

```
['P', 'y', 't', 'h', 'o', 'n']
```

18. Write a Python program to swap comma and dot in a string.

Sample string: "32.054,23"

Expected Output: "32,054.23"

```
main.py + 447u6yutp ⚖
1 s = "32.054,23"
2 s = s.replace('.','#').replace(',','.').replace('#','')
3 print(s)
4
```

Output:

```
32,054.23
```

19. Write a Python program to find smallest and largest word in a given string.

```
main.py + 447u6yutp ⚖
1 s = "Python is very easy language"
2 words = s.split()
3
4 print("Smallest:", min(words, key=len))
5 print("Largest:", max(words, key=len))
6
```

Output:

```
Smallest: is
Largest: language
```

20. Write a Python program to remove all consecutive duplicates of a given string.

The screenshot shows a code editor interface with two tabs: "main.py" and "447u6yutp". The "main.py" tab contains the following Python code:

```
1 print(list(set("aaabbbccaad")))
```

The "447u6yutp" tab shows the execution results:

STDIN

Input for the program (Optional)

Output:

```
[ 'a', 'b', 'd', 'c']
```