Lab assignment 1.5

Task 1:

Prompt 1 : Write a python code to reverse a string without using functions.

CODE :

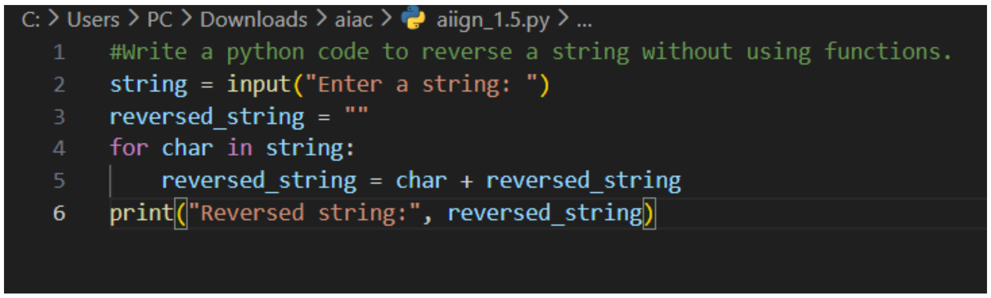
```
string = input("Enter a string: ")

reversed_string = ""

for char in string:

    reversed_string = char + reversed_string

print("Reversed string:", reversed_string)
```

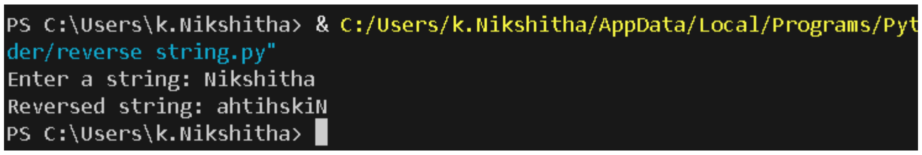


```
C: > Users > PC > Downloads > aiac > aiign_1.5.py > ...
#Write a python code to reverse a string without using functions.
string = input("Enter a string: ")
reversed_string = ""
for char in string:
    reversed_string = char + reversed_string
print("Reversed string:", reversed_string)
```

OUTPUT :

```
Enter a string: Nikshitha

Reversed string: ahtihskiN
```



```
PS C:\Users\k.Nikshitha> & C:/Users/k.Nikshitha/AppData/Local/Programs/Pyt
der/reverse string.py"
Enter a string: Nikshitha
Reversed string: ahtihskiN
PS C:\Users\k.Nikshitha>
```

TASK 2:

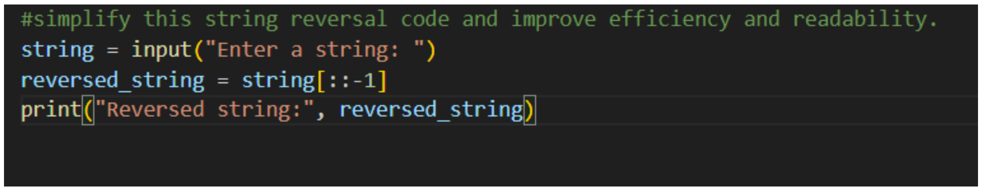
Prompt 2 : Simplify this string reversal code and improve efficiency and readability.

CODE :

```
string = input("Enter a string: ")

reversed_string = string[::-1]

print("Reversed string:", reversed_string)
```

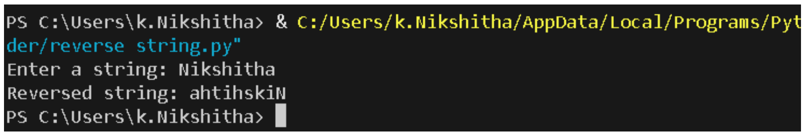


```
#simplify this string reversal code and improve efficiency and readability.
string = input("Enter a string: ")
reversed_string = string[::-1]
print("Reversed string:", reversed_string)
```

OUTPUT :

Enter a string: Nikshitha

Reversed string: ahtihskiN



```
PS C:\Users\k.Nikshitha> & C:/Users/k.Nikshitha/AppData/Local/Programs/Pyt
der/reverse string.py"
Enter a string: Nikshitha
Reversed string: ahtihskiN
PS C:\Users\k.Nikshitha> 
```

TASK 3:

Prompt 3 : Write a string reversal code in python with using functions.

CODE :

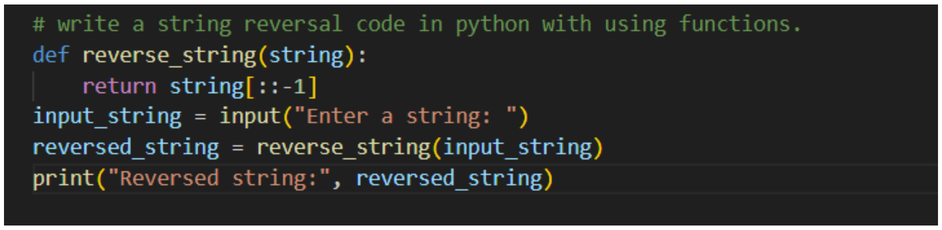
def reverse_string(string):

    return string[::-1]

input_string = input("Enter a string: ")

reversed_string = reverse_string(input_string)

print("Reversed string:", reversed_string)

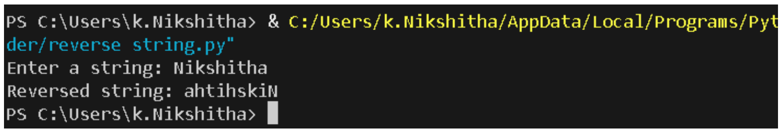


```
# write a string reversal code in python with using functions.
def reverse_string(string):
    return string[::-1]
input_string = input("Enter a string: ")
reversed_string = reverse_string(input_string)
print("Reversed string:", reversed_string)
```

OUTPUT :

Enter a string: Nikshitha

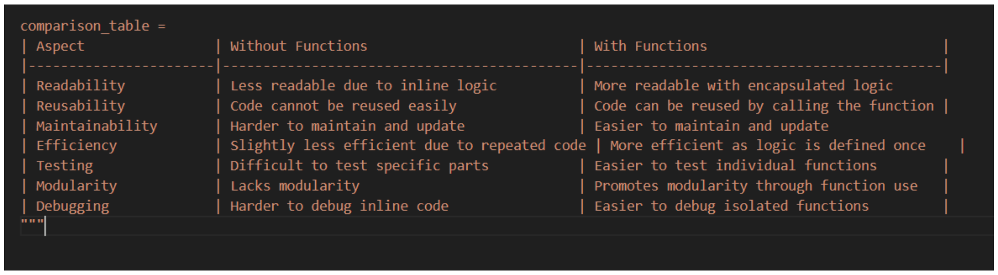
Reversed string: ahtihskiN



```
PS C:\Users\k.Nikshitha> & C:/Users/k.Nikshitha/AppData/Local/Programs/Pyt
der/reverse string.py"
Enter a string: Nikshitha
Reversed string: ahtihskiN
PS C:\Users\k.Nikshitha> 
```

TASK 4:

Prompt 4 : Analyse the code with function and without function and give a comparsion table.



```
comparison_table =
| Aspect                 | Without Functions                            | With Functions                               |
|------------------------|----------------------------------------------|----------------------------------------------|
| Readability            | Less readable due to inline logic            | More readable with encapsulated logic
| Reusability            | Code cannot be reused easily                 | Code can be reused by calling the function |
| Maintainability        | Harder to maintain and update                | Easier to maintain and update
| Efficiency             | Slightly less efficient due to repeated code | More efficient as logic is defined once     |
| Testing                | Difficult to test specific parts             | Easier to test individual functions          |
| Modularity             | Lacks modularity                             | Promotes modularity through function use     |
| Debugging              | Harder to debug inline code                  | Easier to debug isolated functions           |
"""
```

TASK 5:

Prompt 5 : Give different approaches to reverse a string like a loop based and built in or slicing based.

CODE :

# Loop-based approach

string = input("Enter a string: ")

reversed_string = ""

for char in string:

    reversed_string = char + reversed_string

print("Reversed string:", reversed_string)

# Built-in function approach

string = input("Enter a string: ")

reversed_string = ''.join(reversed(string))

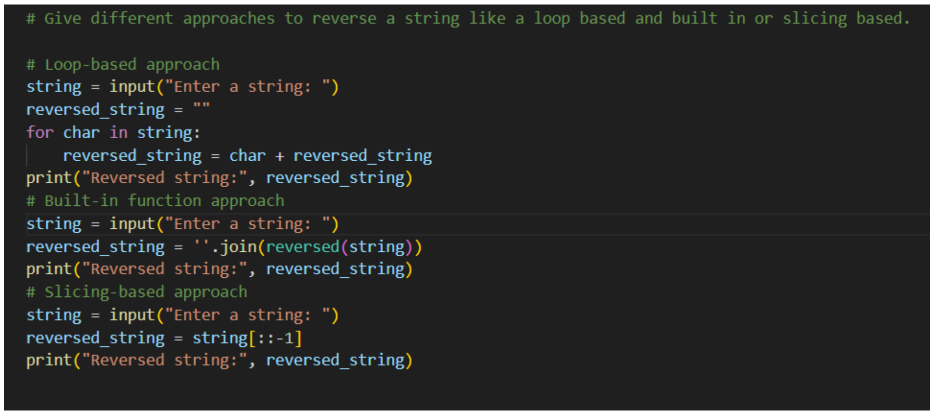
print("Reversed string:", reversed_string)

# Slicing-based approach

string = input("Enter a string: ")

reversed_string = string[::-1]

print("Reversed string:", reversed_string)



```
# Give different approaches to reverse a string like a loop based and built in or slicing based.

# Loop-based approach
string = input("Enter a string: ")
reversed_string = ""
for char in string:
    reversed_string = char + reversed_string
print("Reversed string:", reversed_string)
# Built-in function approach
string = input("Enter a string: ")
reversed_string = ''.join(reversed(string))
print("Reversed string:", reversed_string)
# Slicing-based approach
string = input("Enter a string: ")
reversed_string = string[::-1]
print("Reversed string:", reversed_string)
```

OUTPUT :

Enter a string: Nikshitha

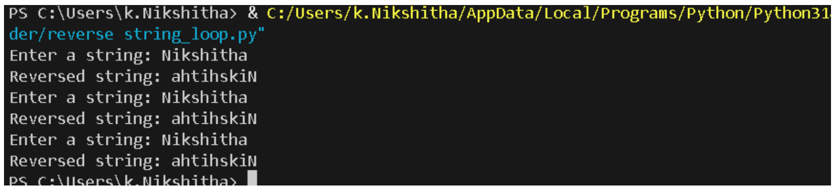
Reversed string: ahtihskiN

Enter a string: Nikshitha

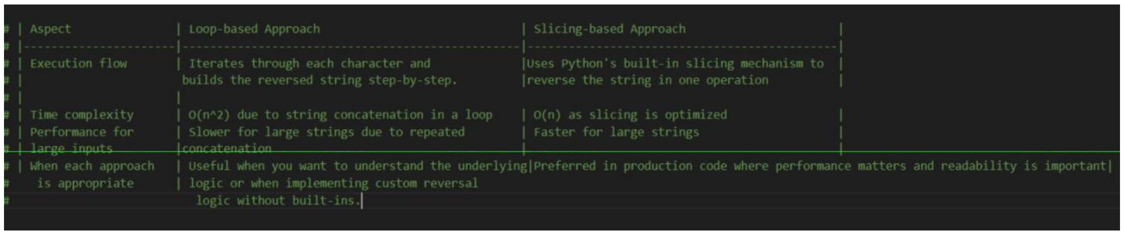
Reversed string: ahtihskiN

Enter a string: Nikshitha

Reversed string: ahtihskiN



```
PS C:\Users\k.Nikshitha> & C:/Users/k.Nikshitha/AppData/Local/Programs/Python/Python31
der/reverse string_loop.py"
Enter a string: Nikshitha
Reversed string: ahtihskiN
Enter a string: Nikshitha
Reversed string: ahtihskiN
Enter a string: Nikshitha
Reversed string: ahtihskiN
PS C:\Users\k.Nikshitha> 
```



```
# | Aspect               | Loop-based Approach                              | Slicing-based Approach                       |
# |----------------------|--------------------------------------------------|----------------------------------------------|
# | Execution flow       | Iterates through each character and              |Uses Python's built-in slicing mechanism to  |
# |                      builds the reversed string step-by-step.          |reverse the string in one operation          |
# |                      |
# | Time complexity      | O(n^2) due to string concatenation in a loop     | O(n) as slicing is optimized                 |
# | Performance for      | Slower for large strings due to repeated         | Faster for large strings                     |
# | large inputs         |concatenation                                     |                                              |
# | When each approach   | Useful when you want to understand the underlying|Preferred in production code where performance matters and readability is important|
#    is appropriate      | logic or when implementing custom reversal
#                          logic without built-ins.
```