

# DATA STRUCTURE AND ALGORITHM



## LAB REPORT

<b>Name</b>	<b>AREEBA FAROOQ</b>
<b>Registration Number</b>	<b>200901058</b>
<b>Batch &amp; Section</b>	<b>BSCS 01 ( SECTION A)</b>
<b>Instructor's Name</b>	<b>Sir Nadeem</b>

**DATE:23-10-2021**

## **TASK 1**

**Write a program to let the user enter a string of his own choice.  
Check whether the given string is a palindrome or not using stack**

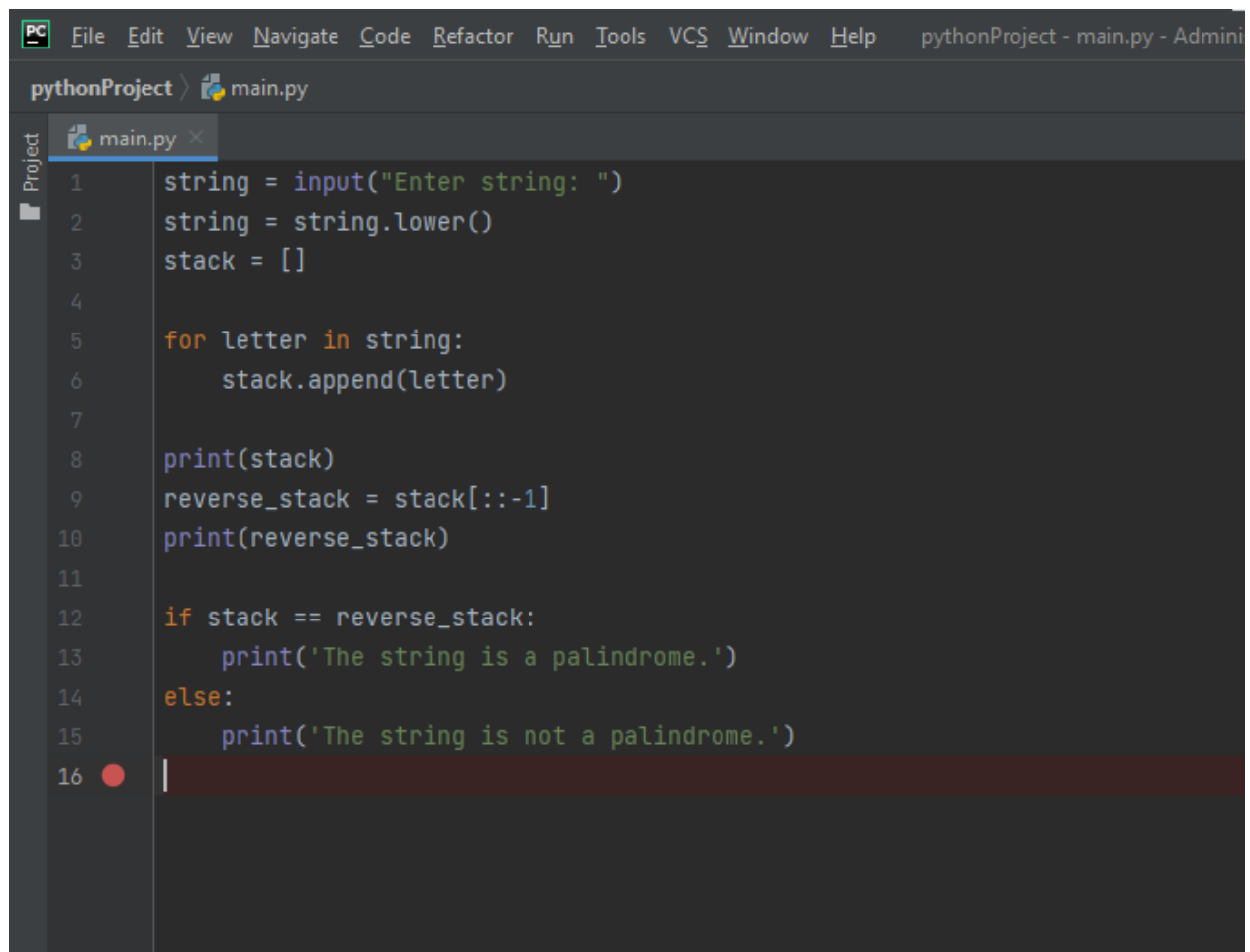
## **SOLUTION**

```
string = input("Enter string: ")
string = string.lower()
stack = []

for letter in string:
    stack.append(letter)

print(stack)
reverse_stack = stack[::-1]
print(reverse_stack)

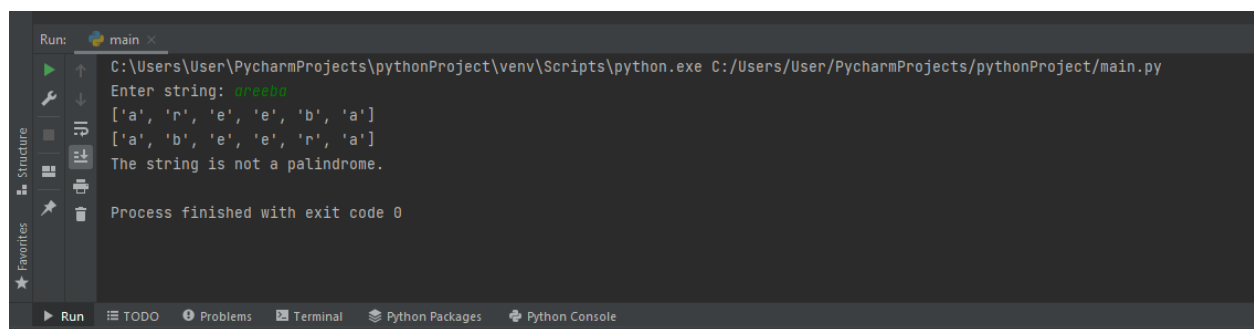
if stack == reverse_stack:
    print('The string is a palindrome.')
else:
    print('The string is not a palindrome.')
```



The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar indicates the project is 'pythonProject' and the file is 'main.py'. The left sidebar shows the 'Project' view with 'main.py' selected. The main editor area displays the following Python code:

```
1 string = input("Enter string: ")
2 string = string.lower()
3 stack = []
4
5 for letter in string:
6     stack.append(letter)
7
8 print(stack)
9 reverse_stack = stack[::-1]
10 print(reverse_stack)
11
12 if stack == reverse_stack:
13     print('The string is a palindrome.')
14 else:
15     print('The string is not a palindrome.')
16
```

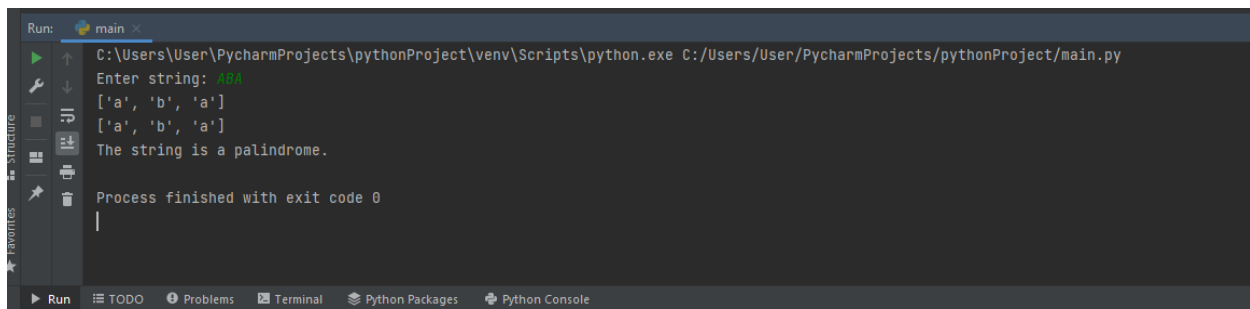
## OUTPUT



The screenshot shows the PyCharm Run console. The title bar indicates the run configuration is 'main'. The console output is as follows:

```
C:\Users\User\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/User/PycharmProjects/pythonProject/main.py
Enter string: areece
['a', 'r', 'e', 'e', 'b', 'a']
['a', 'b', 'e', 'e', 'r', 'a']
The string is not a palindrome.
Process finished with exit code 0
```

The bottom of the console shows tabs for Run, TODO, Problems, Terminal, Python Packages, and Python Console.



## TASK 2

**Write a program to check the balanced parenthesis in the expression or not using stack**

## SOLUTION

```
open_brackets = ["[", "{", "("]
close_brackets = ["]", "}", ")"]
```

```
def check(expression):
    """
    This function checks the balanced brackets in entered expression.
    """
    stack_list = []
    for i in expression:
        if i in open_brackets:
            stack_list.append(i)
        elif i in close_brackets:
            index_position = close_brackets.index(i)
            if ((len(stack_list) > 0) and
                (open_brackets[index_position] == stack_list[len(stack_list) - 1])):
                stack_list.pop()
            else:
                return "Unbalanced"
    if len(stack_list) == 0:
```

```

        return "Balanced"

    else:

        return "Unbalanced"

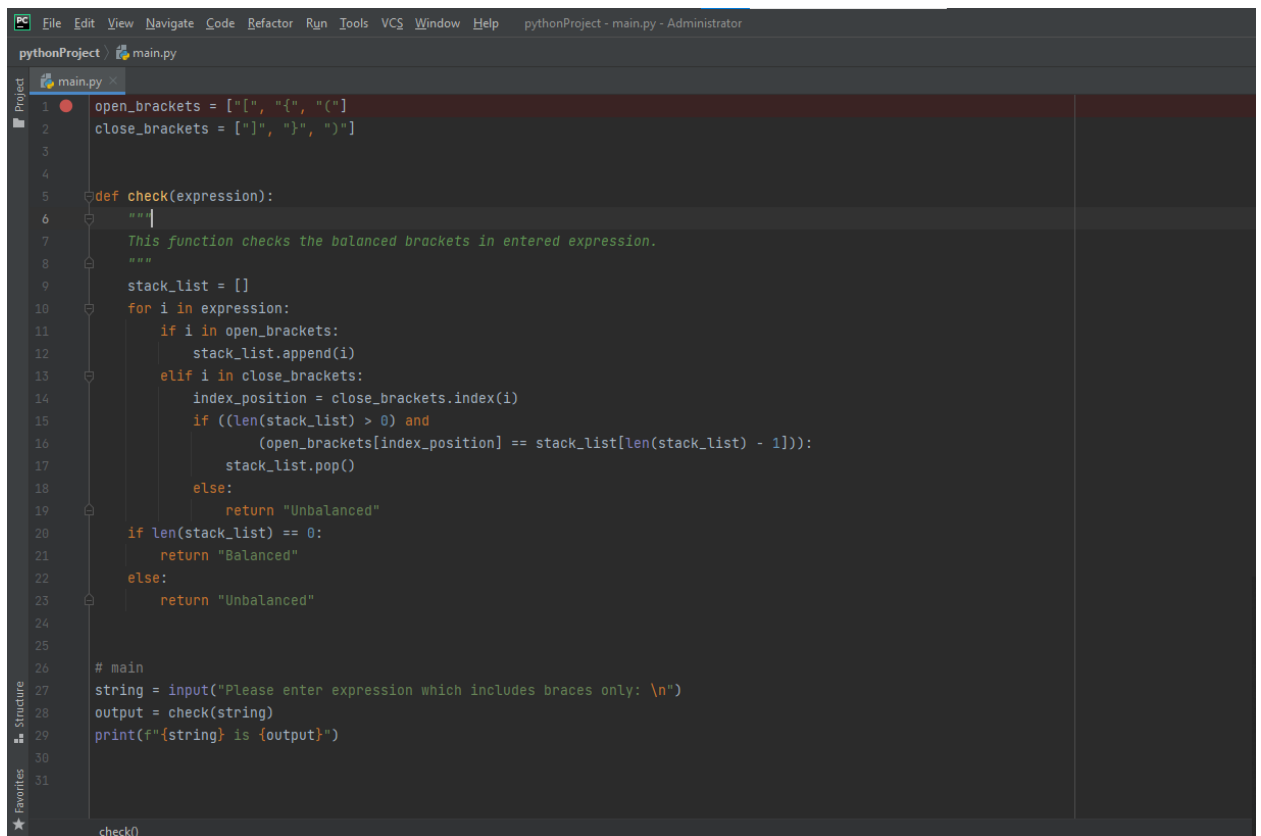
# main

string = input("Please enter expression which includes braces only: \n")

output = check(string)

print(f"{string} is {output}")

```



The screenshot shows a code editor with a dark theme. The file is named 'main.py' and is part of a project called 'pythonProject'. The code implements a function 'check(expression)' that uses a stack to verify if a string of brackets is balanced. The stack is initialized as an empty list. For each character in the expression, if it's an opening bracket, it's pushed onto the stack. If it's a closing bracket, the function checks if the stack is empty or if the top element is the corresponding opening bracket. If not, it returns 'Unbalanced'. If the stack is empty at the end, it returns 'Balanced'. The main block prompts the user for an expression and prints the result.

```

1  open_brackets = ["{", "{", "("]
2  close_brackets = ["}", "}", ")"]
3
4
5  def check(expression):
6      """
7      This function checks the balanced brackets in entered expression.
8      """
9      stack_list = []
10     for i in expression:
11         if i in open_brackets:
12             stack_list.append(i)
13         elif i in close_brackets:
14             index_position = close_brackets.index(i)
15             if ((len(stack_list) > 0) and
16                 (open_brackets[index_position] == stack_list[len(stack_list) - 1])):
17                 stack_list.pop()
18             else:
19                 return "Unbalanced"
20     if len(stack_list) == 0:
21         return "Balanced"
22     else:
23         return "Unbalanced"
24
25
26 # main
27 string = input("Please enter expression which includes braces only: \n")
28 output = check(string)
29 print(f"{string} is {output}")
30
31

```

## OUTPUT

```
Run: main x
C:\Users\User\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/User/PycharmProjects/pythonProject/main.py
Please enter expression which includes braces only:
{()}}
{()}} is Unbalanced
Process finished with exit code 0
```

```
Run: main x
C:\Users\User\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/User/PycharmProjects/pythonProject/main.py
Please enter expression which includes braces only:
{()}{}}
{()}{}} is Balanced
Process finished with exit code 0
```