



Department of Computing Technologies

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC207J

Course Name: Advanced Programming Practice

Experiment No	2
Title of Experiment	To complete all the 13 problems in Jupyter environment
Name of the Student	Abhijay Rajvansh (RA2011003010398)
Date of Experiment	28 - 03 - 2022

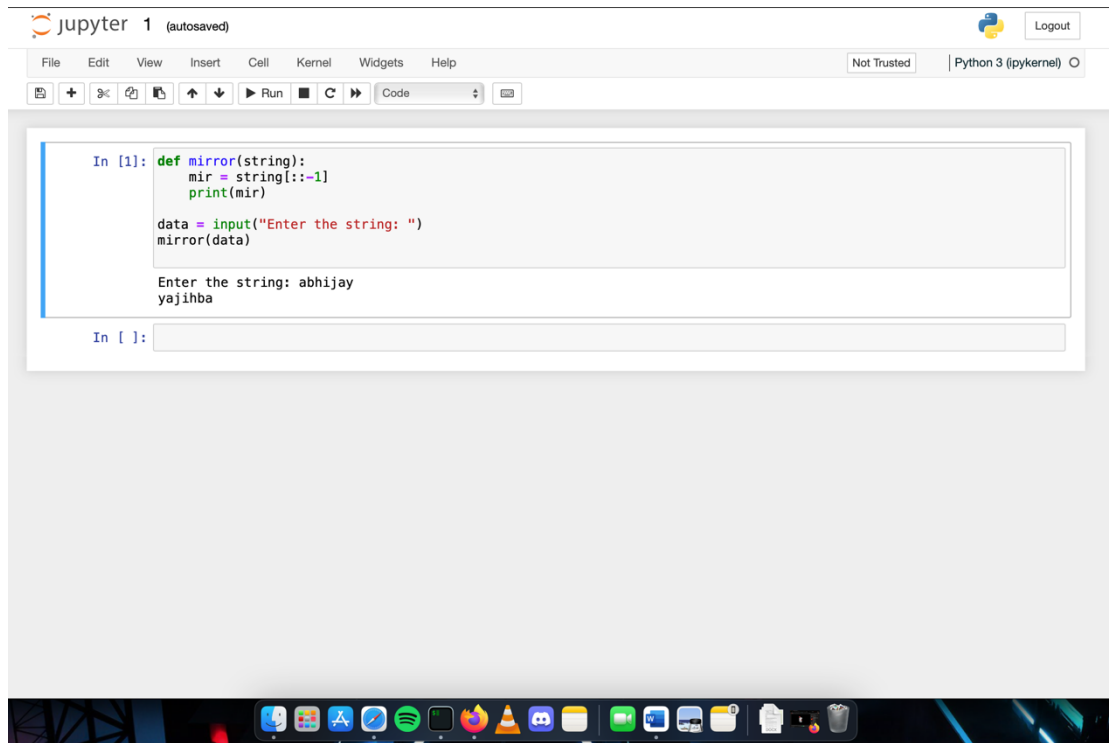
Staff Signature with date

1. AIM: Given a string, find its mirroring image

Code:

```
def mirror(string):  
    mir = string[::-1]  
    print(mir)
```

```
data = input("Enter the string: ")  
mirror(data)
```



Result: Python program to mirror a given string was completed.

2. AIM: Check if two strings are Rotationally Equivalent

Sample Output

string 1 is : srmist

string 2 is : tsrmis

Are two strings Rotationally equal ? : True

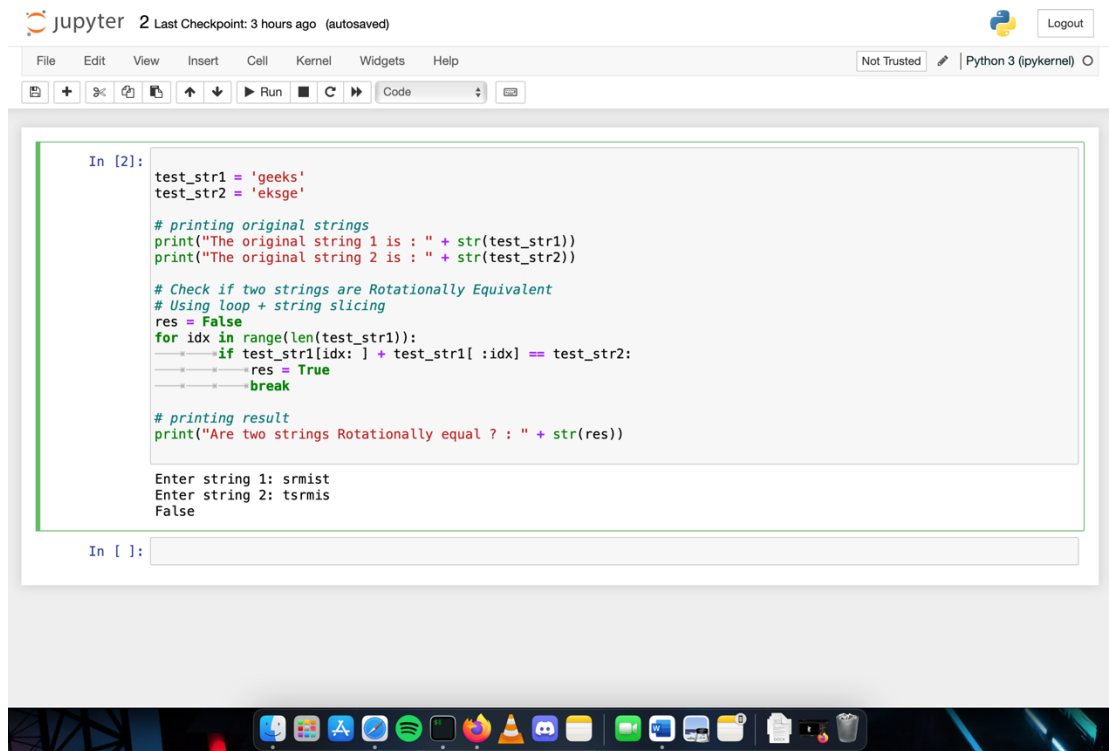
Code:

```
test_str1 = 'geeks'
test_str2 = 'eksge'

# printing original strings
print("The original string 1 is : " + str(test_str1))
print("The original string 2 is : " + str(test_str2))

# Check if two strings are Rotationally Equivalent
# Using loop + string slicing
res = False
for idx in range(len(test_str1)):
    if test_str1[idx: ] + test_str1[ :idx] == test_str2:
        res = True
        break

# printing result
print("Are two strings Rotationally equal ? : " + str(res))
```



Jupyter 2 Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [2]:
test_str1 = 'geeks'
test_str2 = 'eksge'

# printing original strings
print("The original string 1 is : " + str(test_str1))
print("The original string 2 is : " + str(test_str2))

# Check if two strings are Rotationally Equivalent
# Using loop + string slicing
res = False
for idx in range(len(test_str1)):
    if test_str1[idx: ] + test_str1[ :idx] == test_str2:
        res = True
        break

# printing result
print("Are two strings Rotationally equal ? : " + str(res))

Enter string 1: srmist
Enter string 2: tsrmis
False
```

In []:

Result: Python Program to check if two strings are Rotationally Equivalent was completed.

3. AIM: Given a number n, the task is to generate a random binary string of length n.

CODE:

```
import random

n = int(input("Enter the length: "))

list1 = [0, 1]
s = ""

for i in range(n):
    choice = (random.choice(list1))
    s = s + str(choice)
    # print(choice)

print("Binary String :", s)
```

The screenshot displays a Jupyter Notebook environment. At the top, the header shows 'Home Page - Select or create a notebook' and '3 - Jupyter Notebook'. The Jupyter logo and version '3 (autosaved)' are on the left, and a 'Logout' button is on the right. Below the header is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A toolbar contains icons for file operations, a 'Run' button, and a 'Code' dropdown. The main area shows a code cell with the following Python code:

```
In [18]: import random

n = int(input("Enter the length: "))

list1 = [0, 1]
s = ""

for i in range(n):
    choice = (random.choice(list1))
    s = s + str(choice)
    # print(choice)

print("Binary String : ", s)
```

Below the code, the output of the program is shown:

```
Enter the length: 5
Binary String : 00110
```

At the bottom of the code cell, there is an input prompt 'In []:' followed by a text box.

The bottom of the image shows a macOS dock with various application icons.

RESULT: Python program to generate a random binary string of length n was completed.

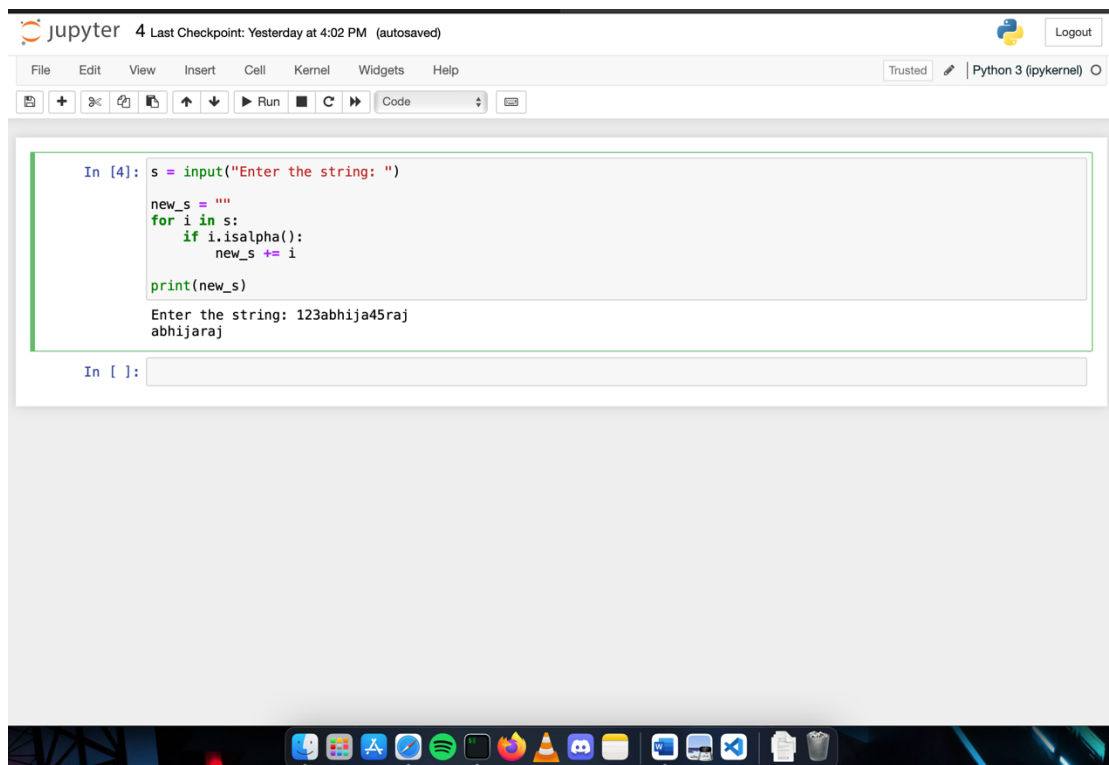
4. AIM: Given a string, remove punctuation and any special characters

CODE

```
s = input("Enter the string: ")

new_s = ""
for i in s:
    if i.isalpha():
        new_s += i

print(new_s)
```

RESULT: Python program to remove punctuation and any special characters for a given string was completed.

5. AIM: Write a Python program to compute element-wise sum of given tuples.

Input

(11, 2, 3, 14)

(13, 5, 22, 10)

(12, 2, 3, 10)

Output

(36, 9, 28, 34)

CODE:

```
import math
```

```
#Input
```

```
t1 = (11, 2, 3, 14)
```

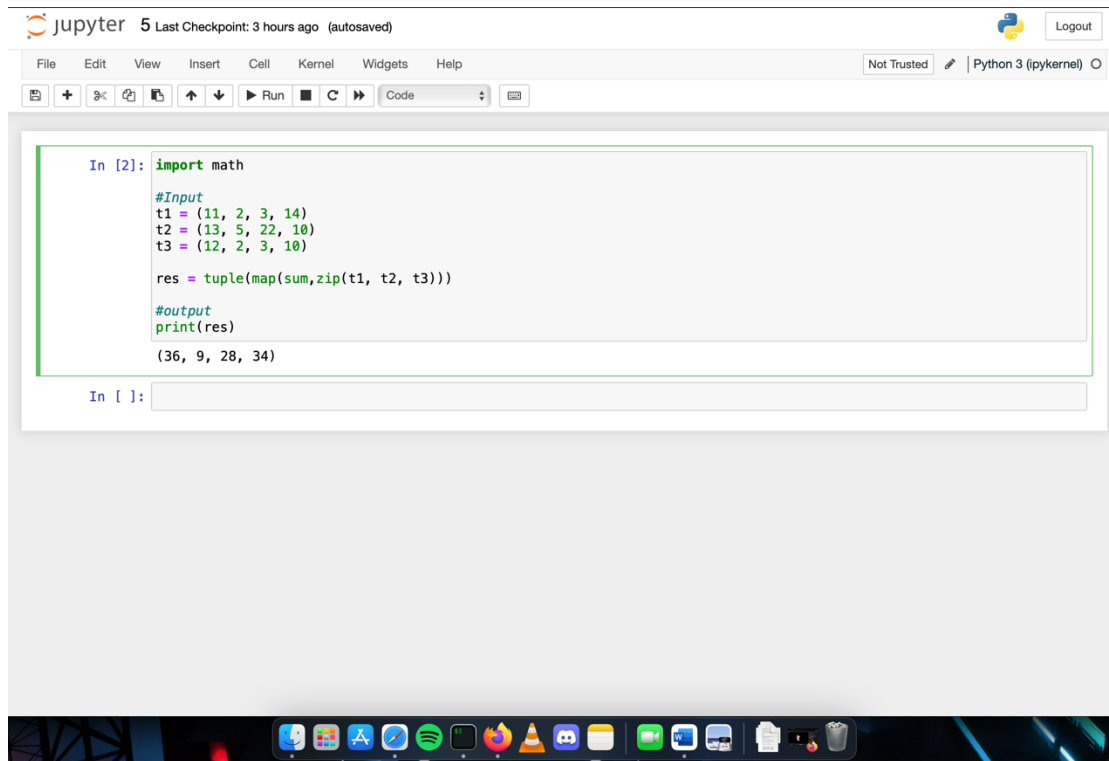
```
t2 = (13, 5, 22, 10)
```

```
t3 = (12, 2, 3, 10)
```

```
res = tuple(map(sum,zip(t1, t2, t3)))
```

```
#output
```

```
print(res)
```



The image shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "5 Last Checkpoint: 3 hours ago (autosaved)". On the right, there is a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar, it says "Not Trusted" and "Python 3 (ipykernel)". Below the menu bar is a toolbar with various icons for file operations, cell execution, and output management. The main area of the notebook contains a code cell with the following Python code:

```
In [2]: import math

#Input
t1 = (11, 2, 3, 14)
t2 = (13, 5, 22, 10)
t3 = (12, 2, 3, 10)

res = tuple(map(sum,zip(t1, t2, t3)))

#output
print(res)

(36, 9, 28, 34)
```

Below the code cell, there is an input prompt "In []:" followed by an empty text box. At the bottom of the image, a macOS dock is visible with various application icons.

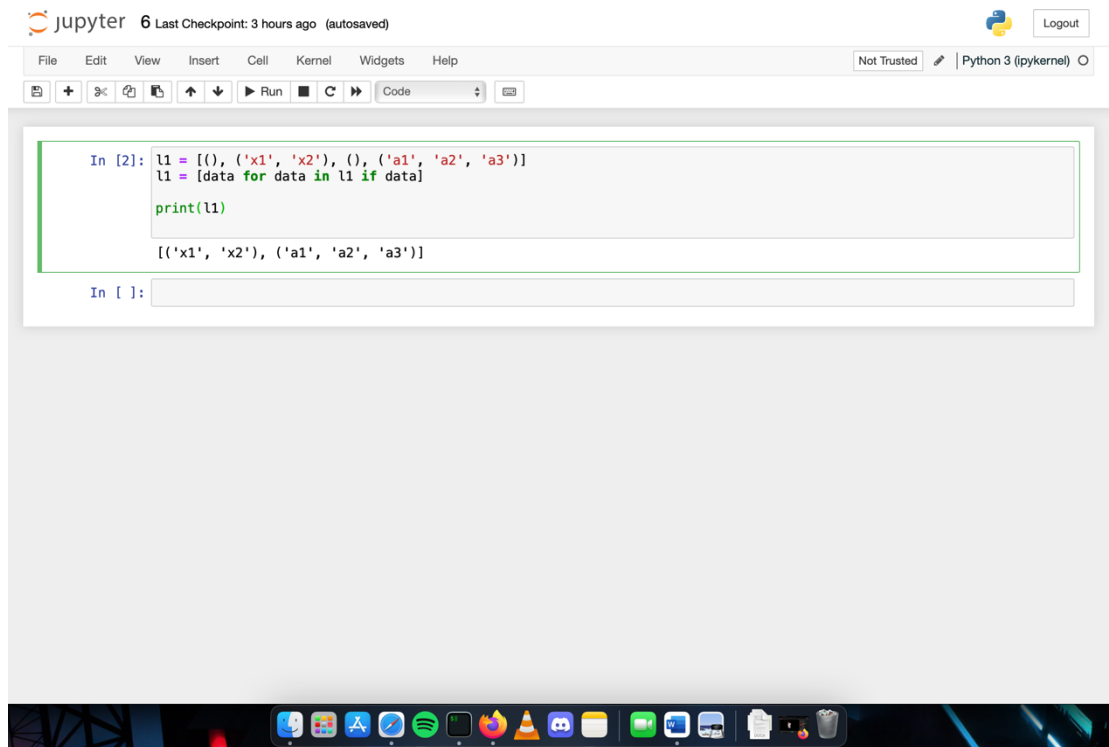
Result: Python program to compute element-wise sum of given tuples was completed.

6. AIM: Write a Python program to remove an empty tuple(s) from a list of tuples.

CODE:

```
l1 = [(), ('x1', 'x2'), (), ('a1', 'a2', 'a3')]
l1 = [data for data in l1 if data]
```

```
print(l1)
```



The image shows a Jupyter Notebook interface. At the top, the Jupyter logo is on the left, followed by the text "6 Last Checkpoint: 3 hours ago (autosaved)". On the right, there is a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar, it says "Not Trusted" and "Python 3 (ipykernel)". Below the menu bar is a toolbar with icons for saving, adding a new cell, undo, redo, running, and other actions. The main area contains a code cell with the following Python code:

```
In [2]: l1 = [(), ('x1', 'x2'), (), ('a1', 'a2', 'a3')]
l1 = [data for data in l1 if data]
print(l1)

[('x1', 'x2'), ('a1', 'a2', 'a3')]

In [ ]:
```

The code defines a list `l1` containing four elements: an empty tuple, a tuple with two strings, another empty tuple, and a tuple with three strings. It then uses a list comprehension to create a new list containing only the non-empty tuples. The result is printed, showing that the empty tuples have been removed.

Result: Python program to remove an empty tuple(s) from a list of tuples was completed.

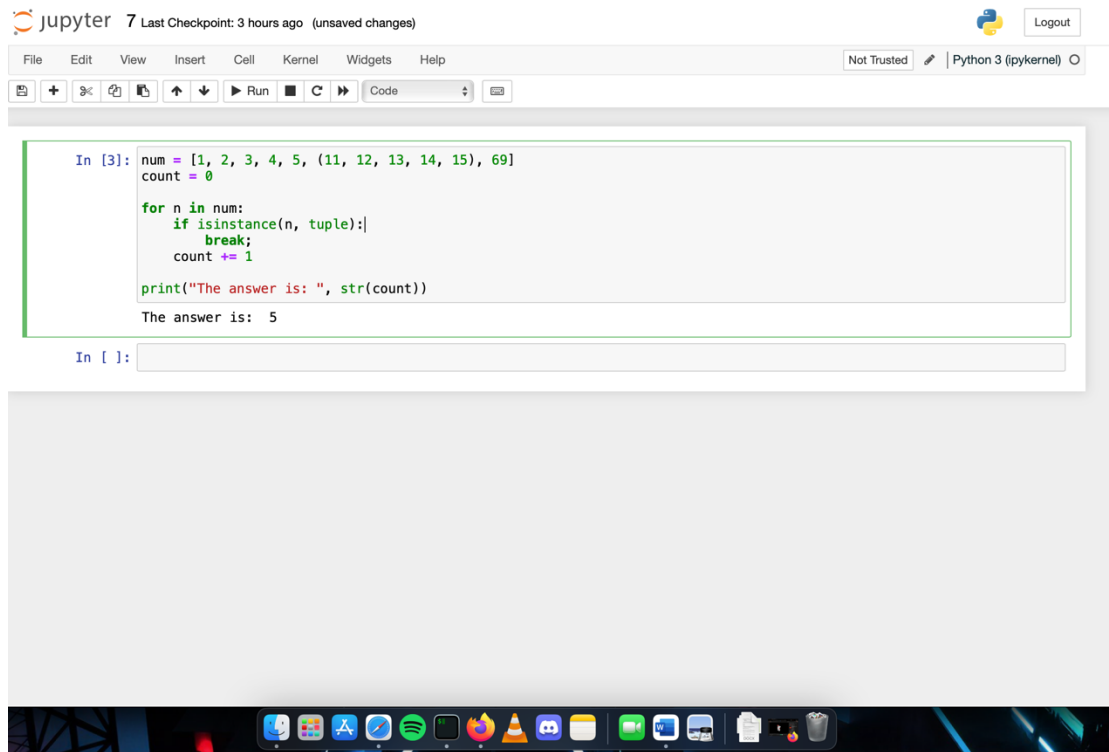
7. AIM: Write a Python program to count the elements in a list until an element is a tuple.

CODE:

```
num = [1, 2, 3, 4, 5, (11, 12, 13, 14, 15), 69]  
count = 0
```

```
for n in num:  
    if isinstance(n, tuple):  
        break;  
    count += 1
```

```
print(count)
```



The image shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "7 Last Checkpoint: 3 hours ago (unsaved changes)". On the right, there is a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar is a "Not Trusted" warning and a "Python 3 (pykernel)" label. Below the menu bar is a toolbar with icons for saving, undo, redo, and running code. The main area of the notebook contains a code cell with the following Python code:

```
In [3]: num = [1, 2, 3, 4, 5, (11, 12, 13, 14, 15), 69]
count = 0

for n in num:
    if isinstance(n, tuple):
        break;
    count += 1

print("The answer is: ", str(count))
```

The output of the code cell is "The answer is: 5". Below the code cell is an empty input field labeled "In []:". The bottom of the image shows a macOS dock with various application icons.

Result: Python program to count the elements in a list until an element is a tuple was completed.

8. AIM: Write a Python program to Convert Tuple Matrix to Tuple List

CODE:

```
l1 = [[(5, 6), (17, 8)], [(1, 3), (16, 17)], [(0, 4), (10, 11)]]
```

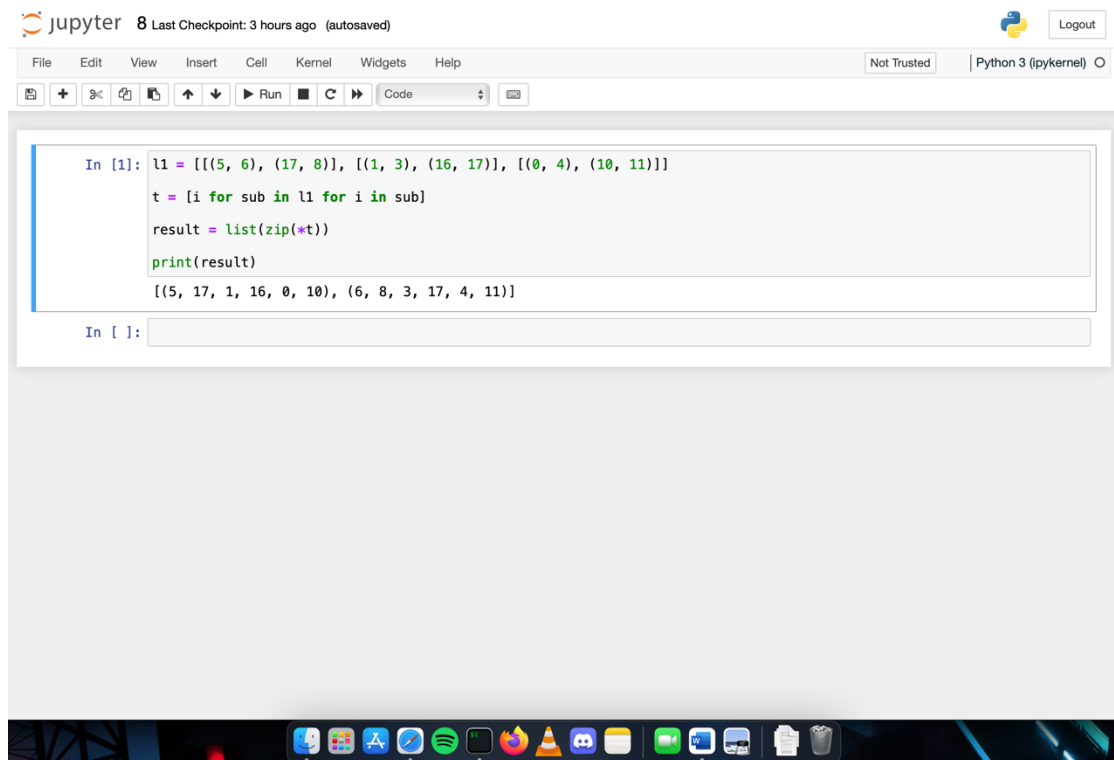
```
t = [i for sub in l1 for i in sub]
```

```
result = list(zip(*t))
```

```
print(result)
```

Sample Input : [[(9, 51), (7, 9)], [(11, 1), (22, 19)]]

Output : [(9, 7, 11, 22), (51, 9, 1, 19)]



The image shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "8 Last Checkpoint: 3 hours ago (autosaved)". On the right, there is a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Not Trusted" and "Python 3 (pykernel)" labels. Below the menu bar is a toolbar with icons for file operations, cell navigation, and execution. The main area contains a code cell with the following Python code:

```
In [1]: l1 = [[(5, 6), (17, 8)], [(1, 3), (16, 17)], [(0, 4), (10, 11)]]
        t = [i for sub in l1 for i in sub]
        result = list(zip(*t))
        print(result)
        [(5, 17, 1, 16, 0, 10), (6, 8, 3, 17, 4, 11)]
```

Below the code cell is an input prompt "In []:" followed by an empty text box. At the bottom of the image is a macOS dock with various application icons.

Result: Python program to Convert Tuple Matrix to Tuple List was completed.

9. AIM: Write a Python program to count unique values in the list.

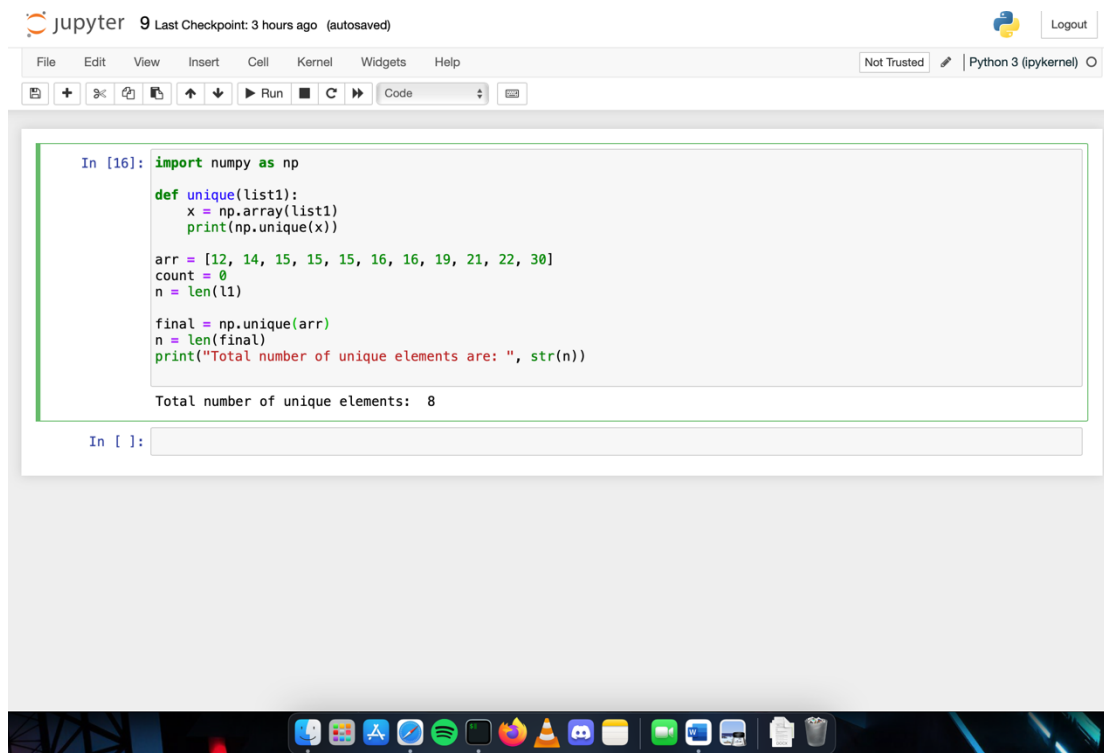
CODE:

```
import numpy as np

def unique(list1):
    x = np.array(list1)
    print(np.unique(x))

arr = [12, 14, 15, 15, 15, 16, 16, 19, 21, 22, 30]
count = 0
n = len(arr)

final = np.unique(arr)
n = len(final)
print("Total number of unique elements are: ", str(n))
```



The image shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "9 Last Checkpoint: 3 hours ago (autosaved)". On the right, there is a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar is a status bar showing "Not Trusted", a Python logo, and "Python 3 (ipykernel)". Below the menu bar is a toolbar with icons for file operations, cell execution, and other notebook functions. The main area of the notebook contains a code cell with the following Python code:

```
In [16]: import numpy as np

def unique(list1):
    x = np.array(list1)
    print(np.unique(x))

arr = [12, 14, 15, 15, 15, 16, 16, 19, 21, 22, 30]
count = 0
n = len(l1)

final = np.unique(arr)
n = len(final)
print("Total number of unique elements are: ", str(n))

Total number of unique elements: 8
```

Below the code cell, there is an input prompt "In []:" followed by an empty text box. At the bottom of the image, a macOS dock is visible with various application icons.

Result: Python program to count unique values in the list was completed.

10. AIM: Python Program to print all Possible Combinations from the three Digits

CODE:

```
def ncr(L):  
    for i in range(3):  
        for j in range(3):  
            for k in range(3):  
                if (i!=j and j!=k and i!=k):  
                    print(L[i], L[j], L[k])  
  
a = int(input("Enter element A: "))  
b = int(input("Enter element B: "))  
c = int(input("Enter element C: "))  
  
print("Following are the all possible permutations:-")  
ncr([a, b, c])
```

The screenshot shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "10 Last Checkpoint: 3 hours ago (autosaved)". On the right, there is a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Not Trusted" and "Python 3 (ipykernel)". Below the menu bar is a toolbar with icons for saving, adding, undo, redo, running, and other functions. The main area of the notebook contains a code cell with the following Python code:

```
In [3]: def ncr(L):
        for i in range(3):
            for j in range(3):
                for k in range(3):
                    if (i!=j and j!=k and i!=k):
                        print(L[i], L[j], L[k])

        a = int(input("Enter element A: "))
        b = int(input("Enter element B: "))
        c = int(input("Enter element C: "))

        print("Following are the all possible permutations:-")
        ncr([a, b, c])
```

The output of the code is displayed below the code cell:

```
Enter element A: 2
Enter element B: 3
Enter element C: 7
Following are the all possible permutations:-
2 3 7
2 7 3
3 2 7
3 7 2
7 2 3
7 3 2
```

At the bottom of the notebook, there is an input prompt "In []:" followed by a text box. The entire notebook interface is set against a dark background with a grid pattern.

RESULT: Python program to print all Possible Combinations from the three Digits was completed.

11. AIM: Write a Python program (using function) to print the even numbers from a given list.

Note : function type - with arguments but no return values

CODE:

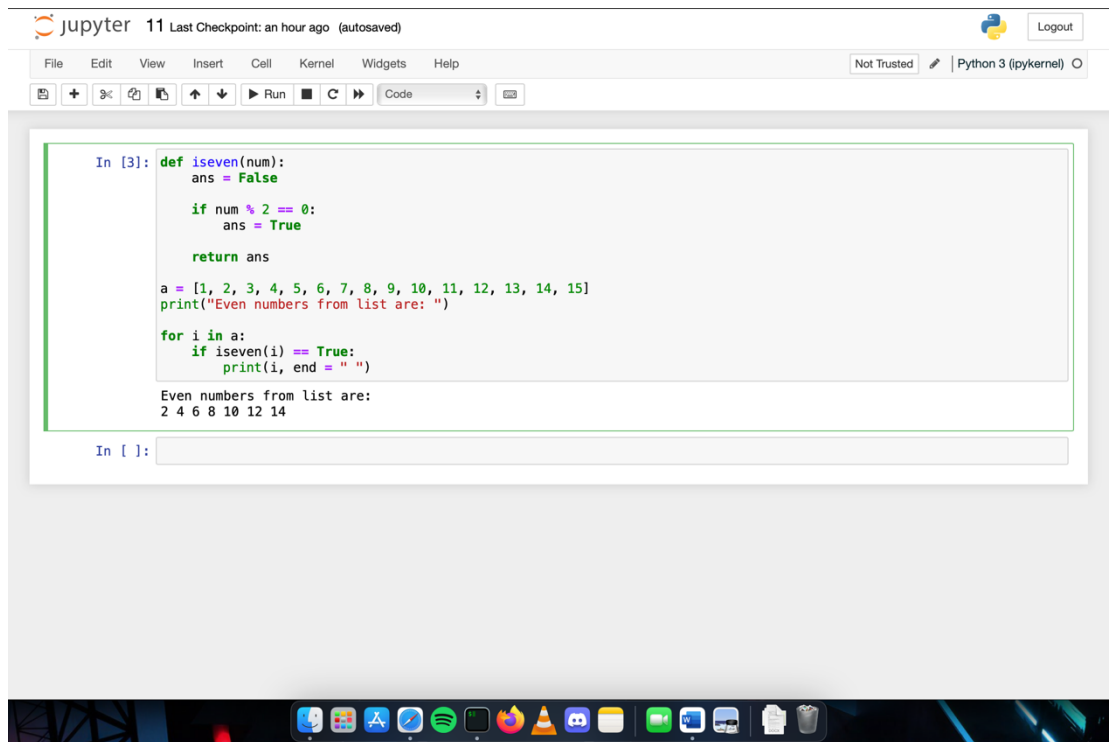
```
def iseven(num):
    ans = False

    if num % 2 == 0:
        ans = True

    return ans

a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
print("Even numbers from list are: ")

for i in a:
    if iseven(i) == True:
        print(i, end = " ")
```



The image shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "jupyter 11 Last Checkpoint: an hour ago (autosaved)". On the right, there is a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar is a "Not Trusted" warning and a "Python 3 (ipykernel)" label. Below the menu bar is a toolbar with icons for file operations, cell navigation, and execution. The main area contains a code cell with the following Python code:

```
In [3]: def iseven(num):  
        ans = False  
  
        if num % 2 == 0:  
            ans = True  
  
        return ans  
  
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]  
print("Even numbers from list are: ")  
  
for i in a:  
    if iseven(i) == True:  
        print(i, end = " ")  
  
Even numbers from list are:  
2 4 6 8 10 12 14
```

Below the code cell is an input prompt "In []:" followed by an empty text box. At the bottom of the image, a macOS dock is visible with various application icons.

RESULT: Python program (using function) to print the even numbers from a given list was completed.

12. AIM: Write a Python function (using function) that checks whether a passed string is palindrome or not.

Note : function type - No arguments with return values

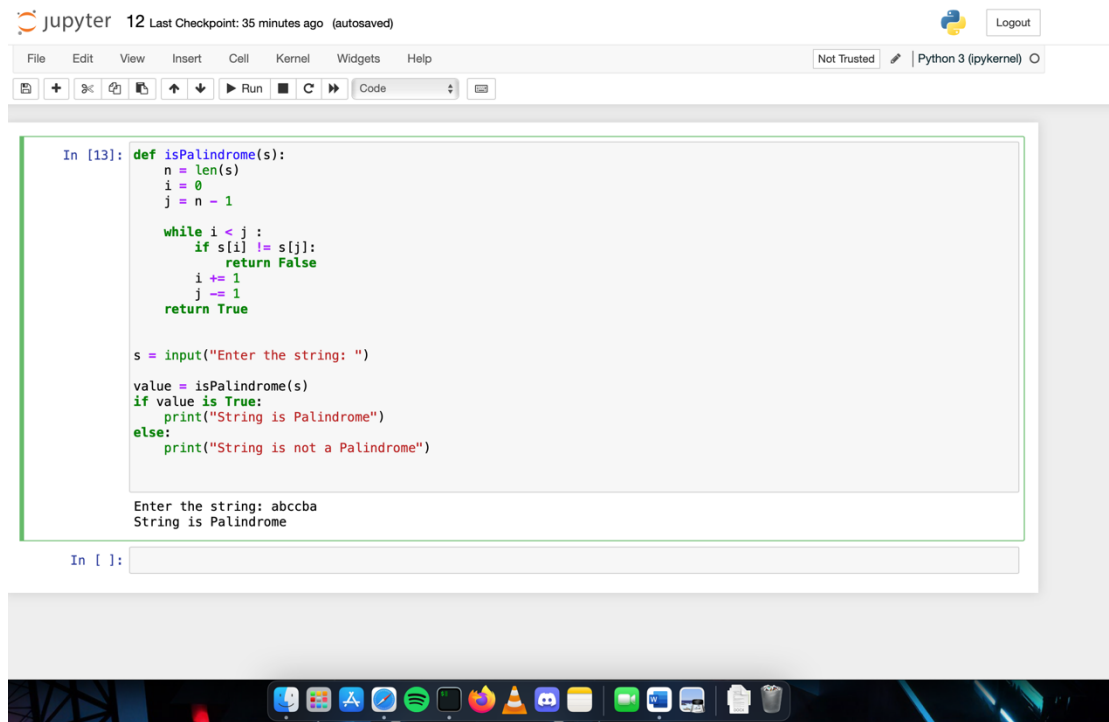
CODE:

```
def isPalindrome(s):
    n = len(s)
    i = 0
    j = n - 1

    while i < j :
        if s[i] != s[j]:
            return False
        i += 1
        j -= 1
    return True

s = input("Enter the string: ")

value = isPalindrome(s)
if value is True:
    print("String is Palindrome")
else:
    print("String is not a Palindrome")
```

The image shows a Jupyter Notebook interface. At the top, the Jupyter logo and text "12 Last Checkpoint: 35 minutes ago (autosaved)" are visible. The top bar includes a "Logout" button and a "Python 3 (ipykernel)" label. The main area contains a code cell with the following Python code:

```
In [13]: def isPalindrome(s):
          n = len(s)
          i = 0
          j = n - 1

          while i < j :
              if s[i] != s[j]:
                  return False
              i += 1
              j -= 1
          return True

          s = input("Enter the string: ")

          value = isPalindrome(s)
          if value is True:
              print("String is Palindrome")
          else:
              print("String is not a Palindrome")
```

Below the code cell, the output of the program is displayed:

```
Enter the string: abccba
String is Palindrome
```

The bottom of the image shows a macOS dock with various application icons.

RESULT: Python program to function (using function) that checks whether a passed string is palindrome or not was completed.

13. AIM: Write a Python function (using function) that checks whether a given number is prime or not

Note : function type - with arguments with return values

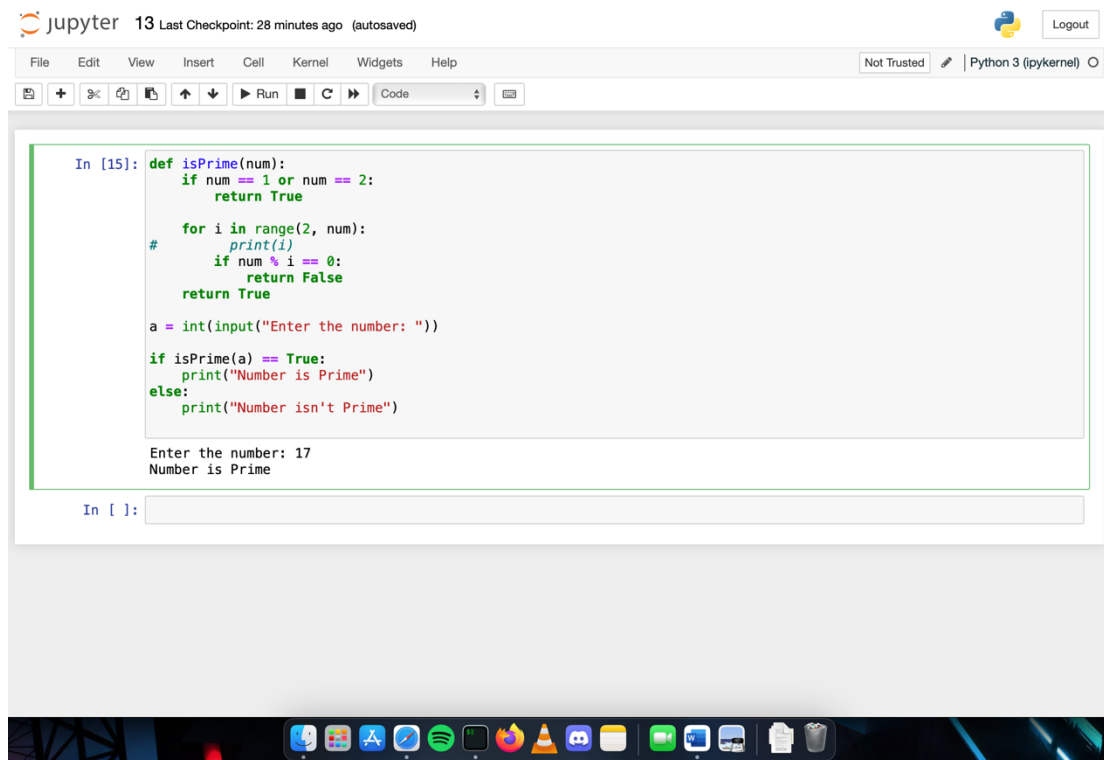
CODE:

```
def isPrime(num):
    if num == 1 or num == 2:
        return True

    for i in range(2, num):
        # print(i)
        if num % i == 0:
            return False
    return True

a = int(input("Enter the number: "))

if isPrime(a) == True:
    print("Number is Prime")
else:
    print("Number isn't Prime")
```



Jupyter 13 Last Checkpoint: 28 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [15]: def isPrime(num):
          if num == 1 or num == 2:
              return True

          for i in range(2, num):
              # print(i)
              if num % i == 0:
                  return False
          return True

          a = int(input("Enter the number: "))

          if isPrime(a) == True:
              print("Number is Prime")
          else:
              print("Number isn't Prime")

          Enter the number: 17
          Number is Prime
```

In []:

RESULT: Python program to function (using function) that checks whether a given number is prime or not was completed.