**NAME – SAARA ANAND**

**REG NO – 21BCE8156**

**SLOT – L55+L56**

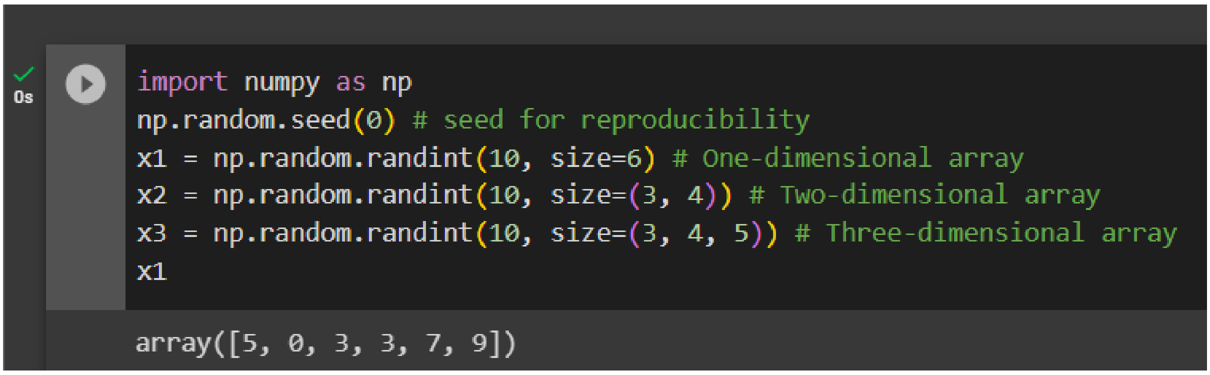
**FDA LAB ASSIGNMENT 8**

Basics of NumPy Arrays

NumPy Array Attributes:

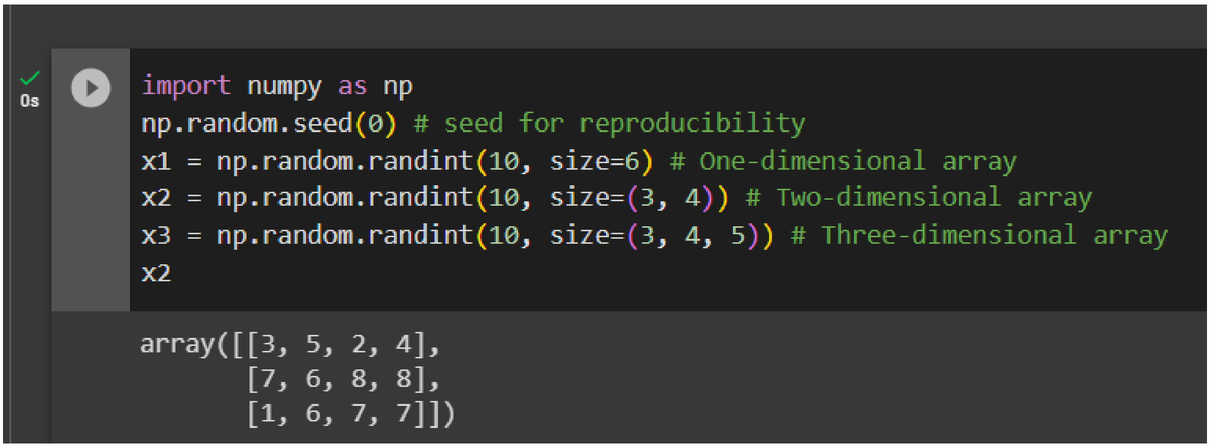
1)

```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
```



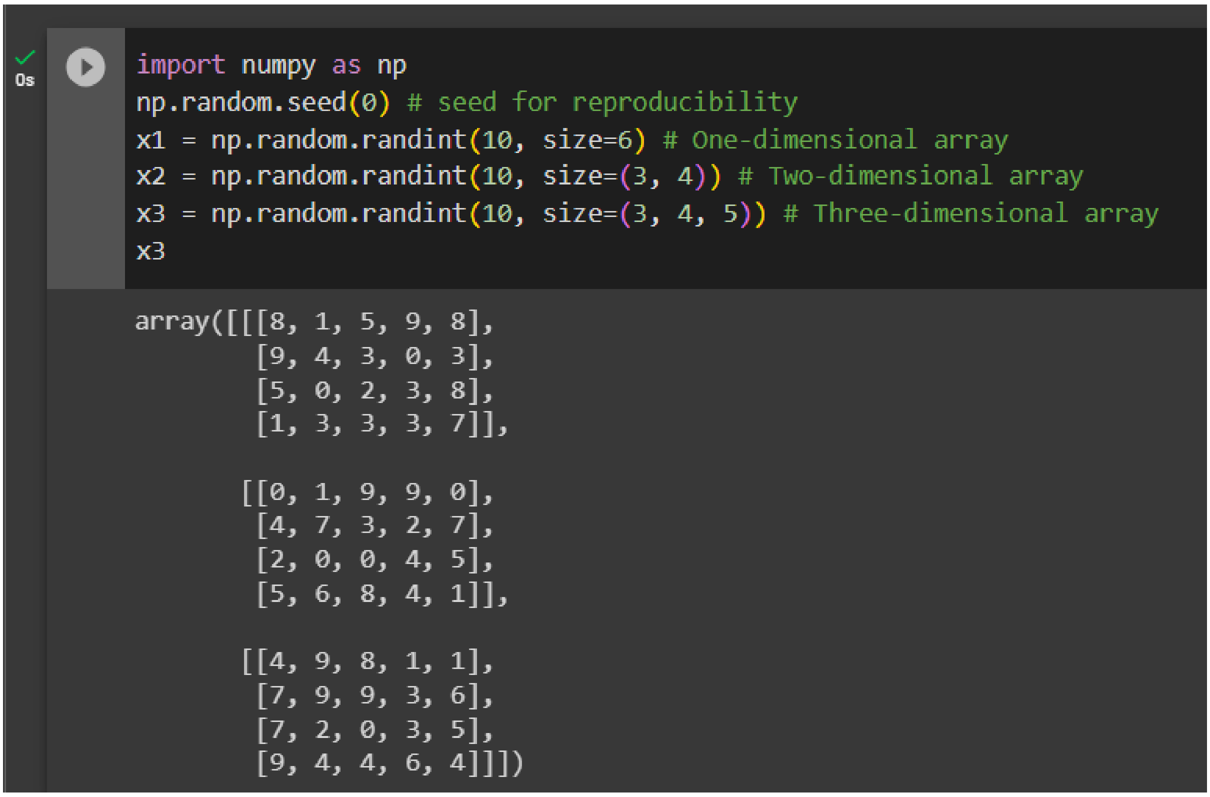
```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x1
```

```
array([5, 0, 3, 3, 7, 9])
```



```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x2
```

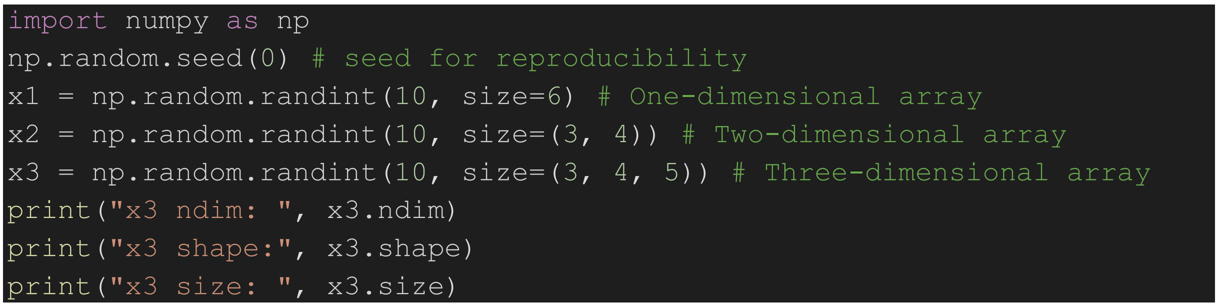
```
array([[3, 5, 2, 4],
       [7, 6, 8, 8],
       [1, 6, 7, 7]])
```



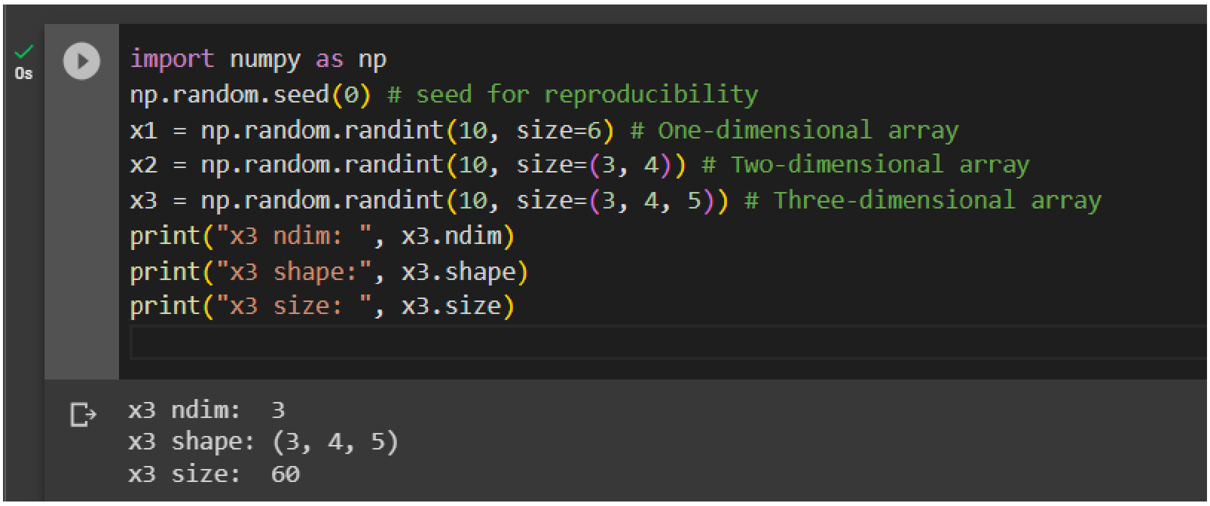
```python
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x3
```

```
array([[[8, 1, 5, 9, 8],
        [9, 4, 3, 0, 3],
        [5, 0, 2, 3, 8],
        [1, 3, 3, 3, 7]],

       [[0, 1, 9, 9, 0],
        [4, 7, 3, 2, 7],
        [2, 0, 0, 4, 5],
        [5, 6, 8, 4, 1]],

       [[4, 9, 8, 1, 1],
        [7, 9, 9, 3, 6],
        [7, 2, 0, 3, 5],
        [9, 4, 4, 6, 4]]])
```

2)



```python
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
print("x3 ndim: ", x3.ndim)
print("x3 shape:", x3.shape)
print("x3 size: ", x3.size)
```

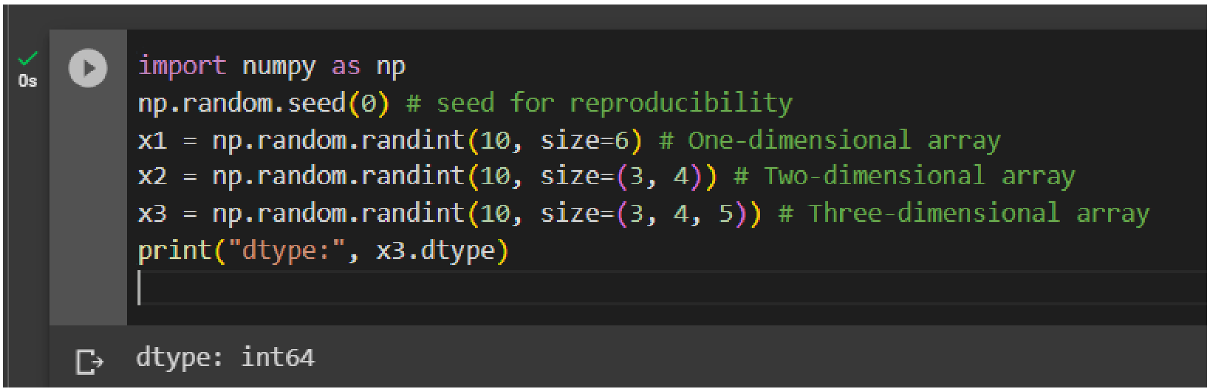


```python
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
print("x3 ndim: ", x3.ndim)
print("x3 shape:", x3.shape)
print("x3 size: ", x3.size)
```

```
x3 ndim:  3
x3 shape: (3, 4, 5)
x3 size:  60
```

3)

```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
print("dtype:", x3.dtype)
```

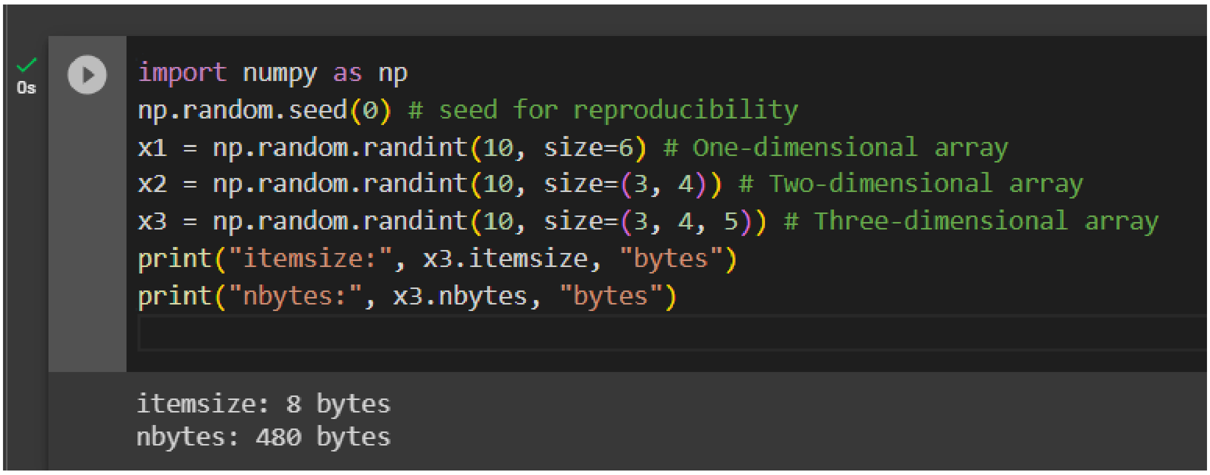


```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
print("dtype:", x3.dtype)
```

```
dtype: int64
```

4)

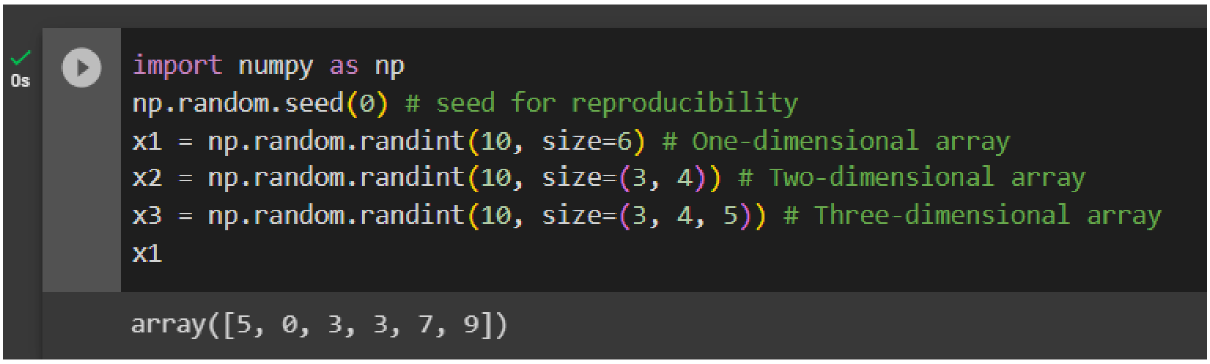
```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
print("itemsize:", x3.itemsize, "bytes")
print("nbytes:", x3.nbytes, "bytes")
```



```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
print("itemsize:", x3.itemsize, "bytes")
print("nbytes:", x3.nbytes, "bytes")
```

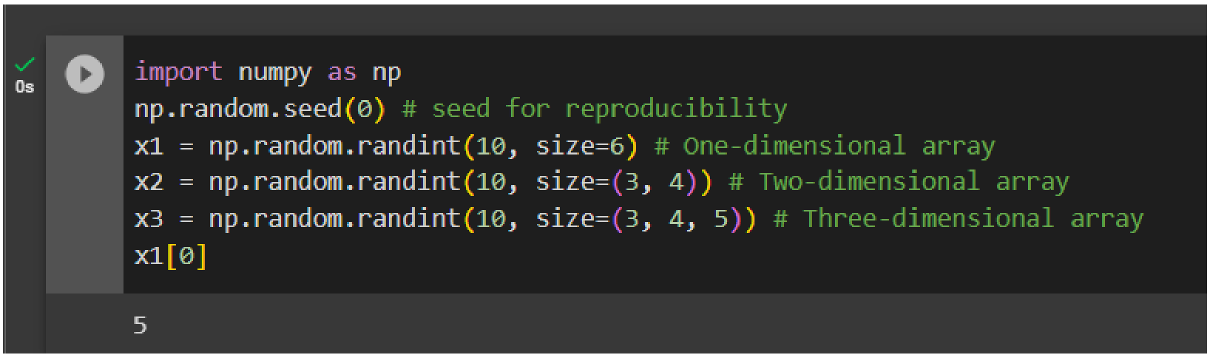
```
itemsize: 8 bytes
nbytes: 480 bytes
```

# Array Indexing-



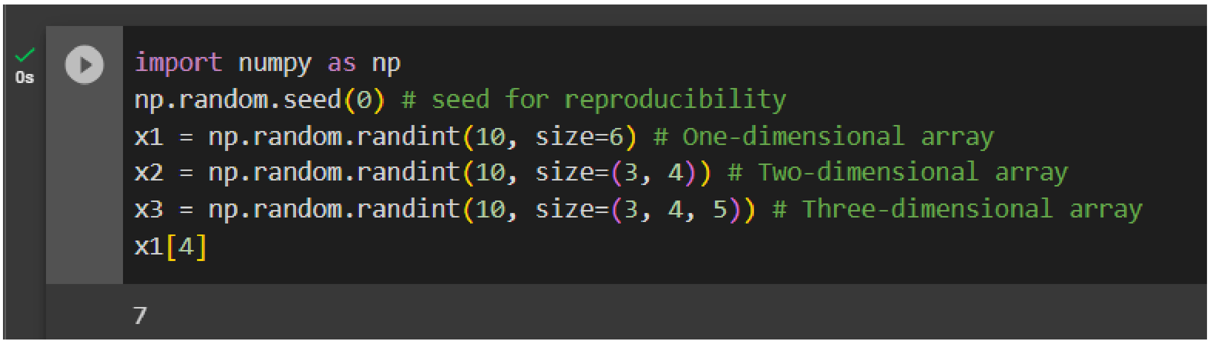
```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x1
```

```
array([5, 0, 3, 3, 7, 9])
```



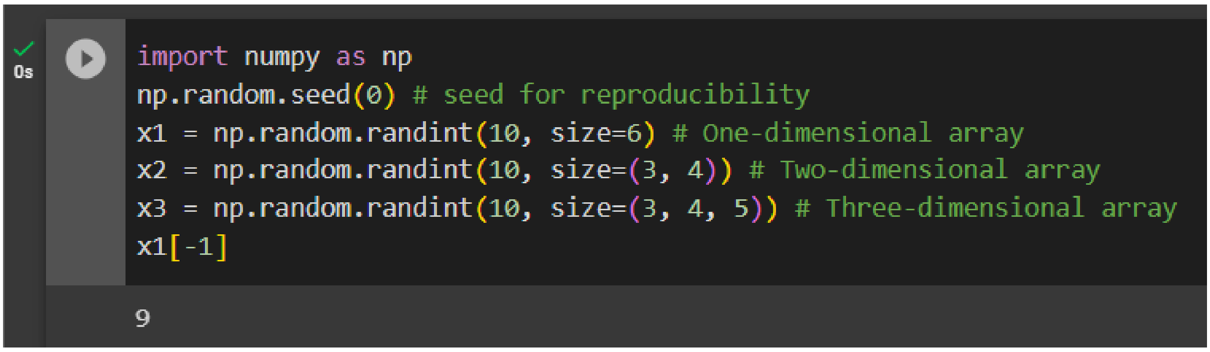
```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x1[0]
```

```
5
```



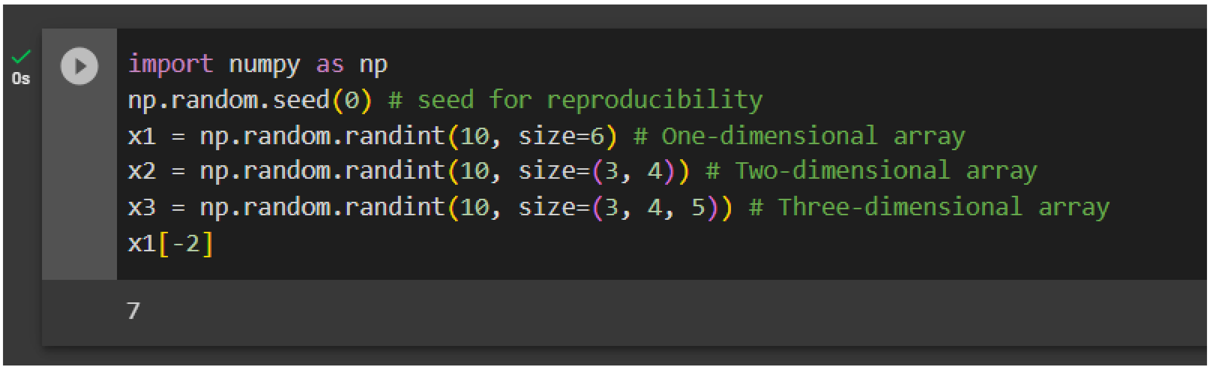
```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x1[4]
```

```
7
```



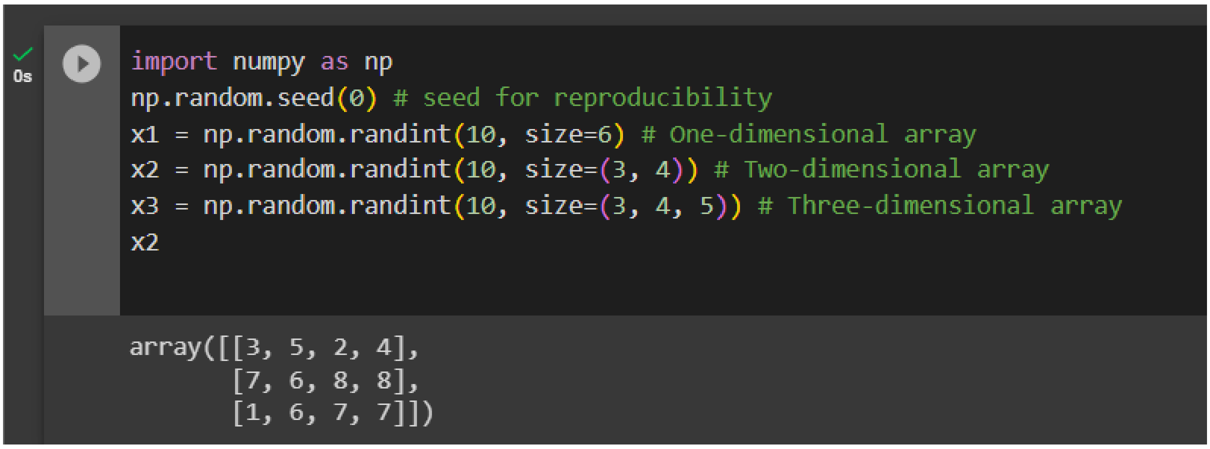
```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x1[-1]
```

```
9
```



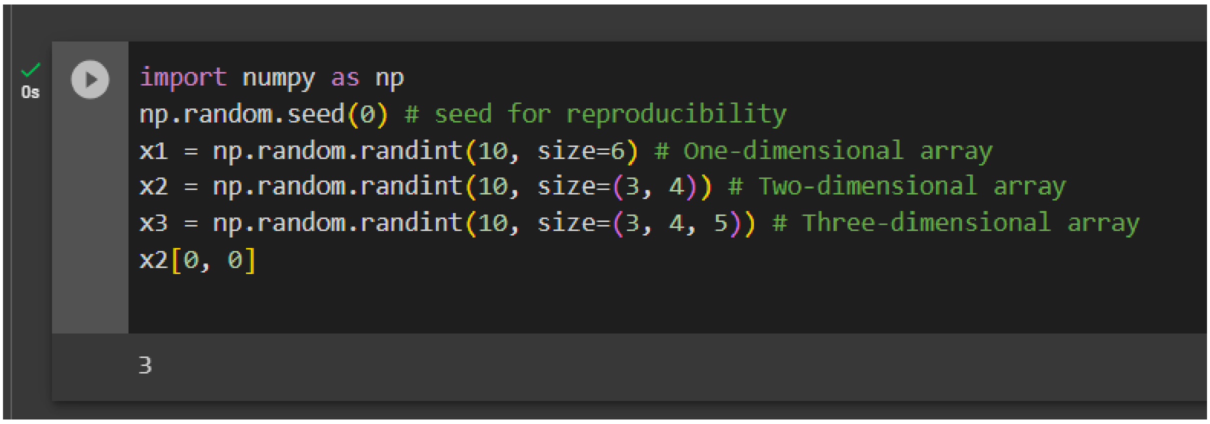
```python
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x1[-2]
```

```
7
```



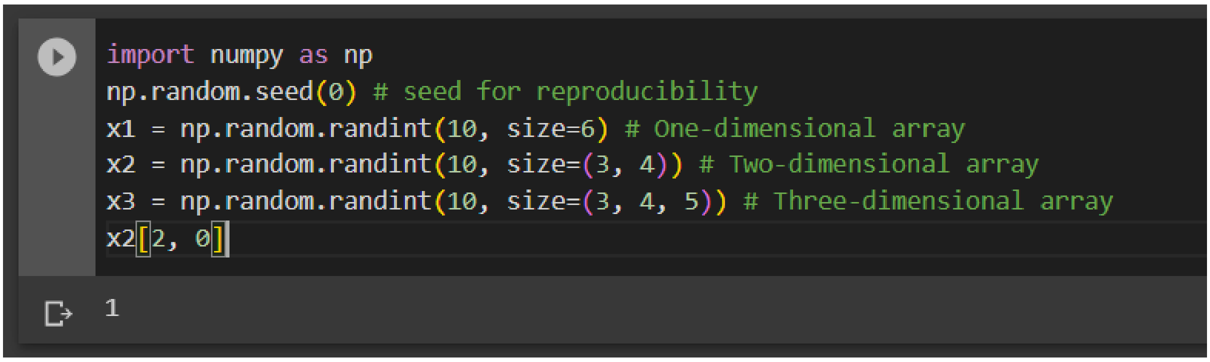
```python
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x2
```

```
array([[3, 5, 2, 4],
       [7, 6, 8, 8],
       [1, 6, 7, 7]])
```



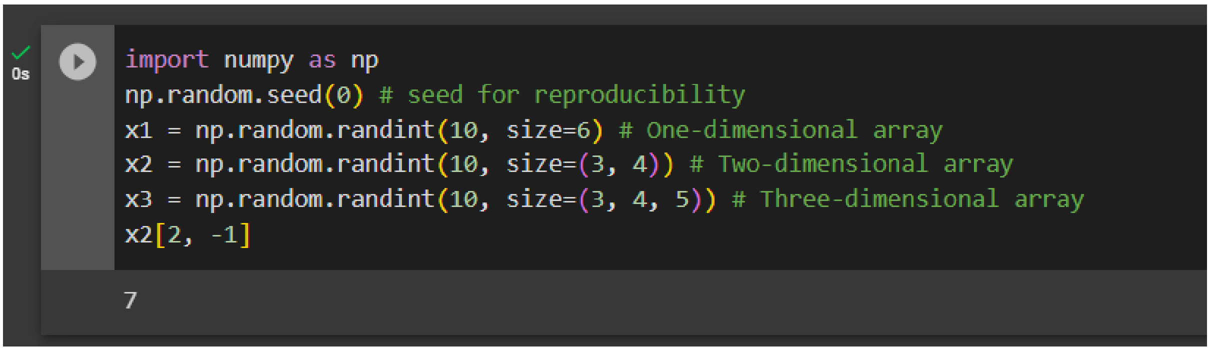
```python
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x2[0, 0]
```

```
3
```



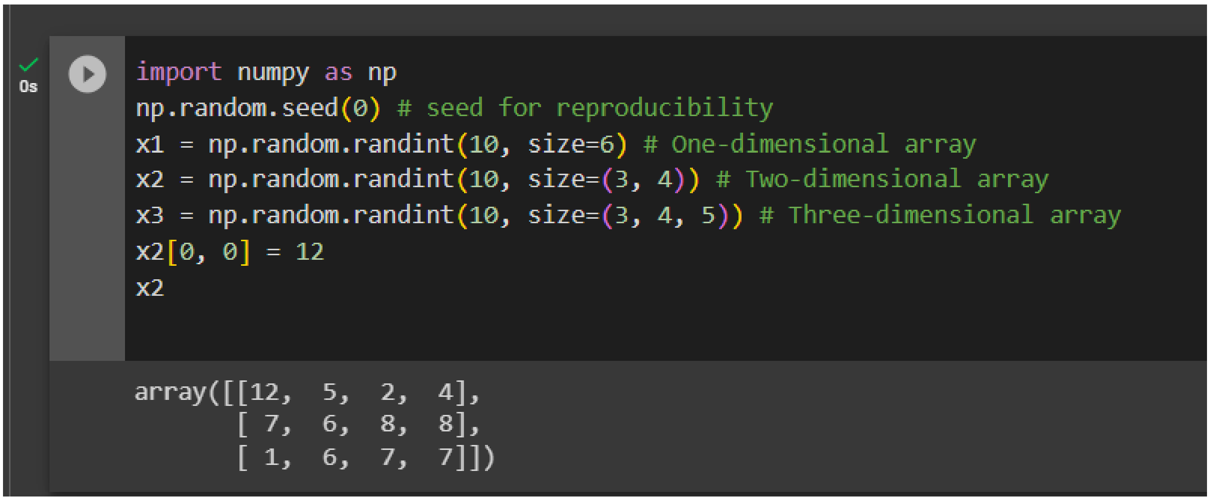
```python
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x2[2, 0]
```

```
1
```



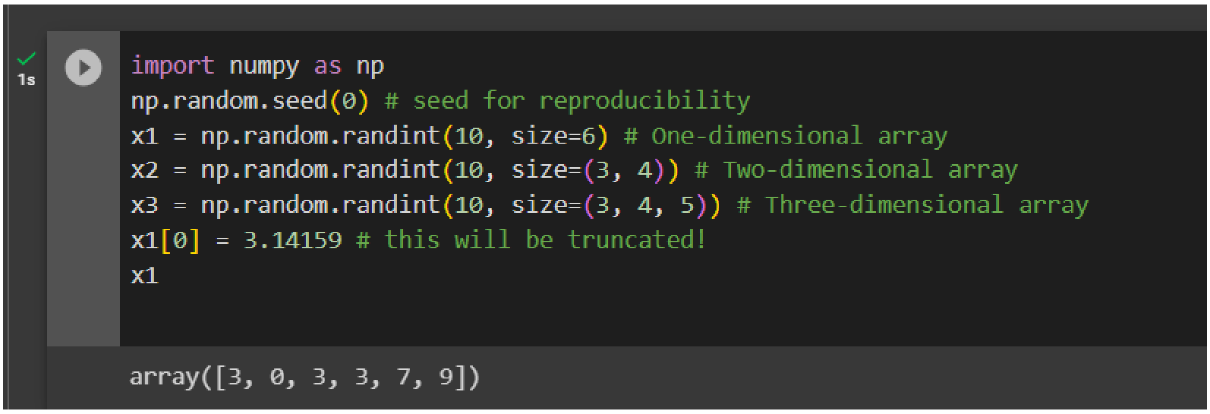
```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x2[2, -1]
```

```
7
```



```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x2[0, 0] = 12
x2
```

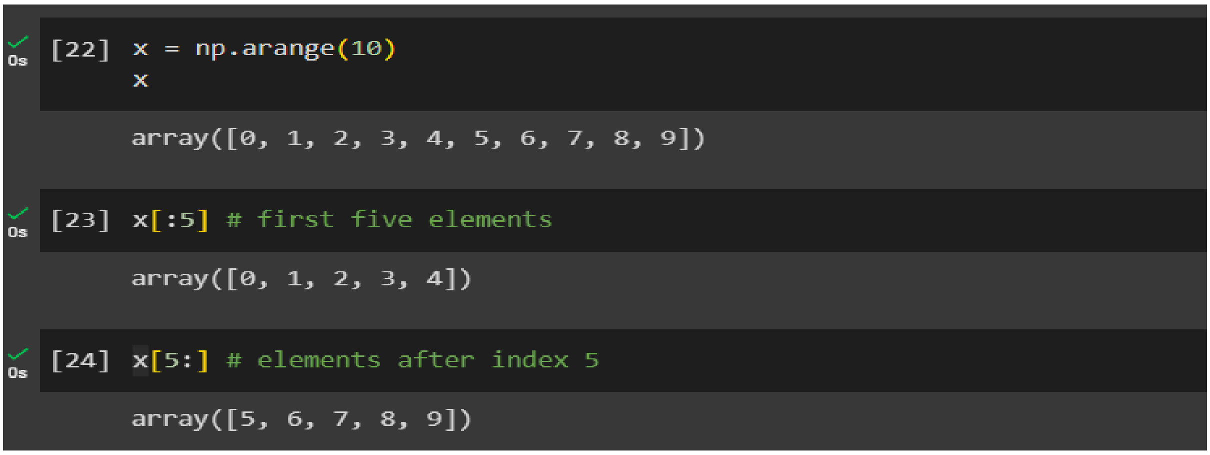
```
array([[12,  5,  2,  4],
       [ 7,  6,  8,  8],
       [ 1,  6,  7,  7]])
```



```
import numpy as np
np.random.seed(0) # seed for reproducibility
x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
x1[0] = 3.14159 # this will be truncated!
x1
```

```
array([3, 0, 3, 3, 7, 9])
```

## Array Slicing -



```
[22] x = np.arange(10)
     x
```

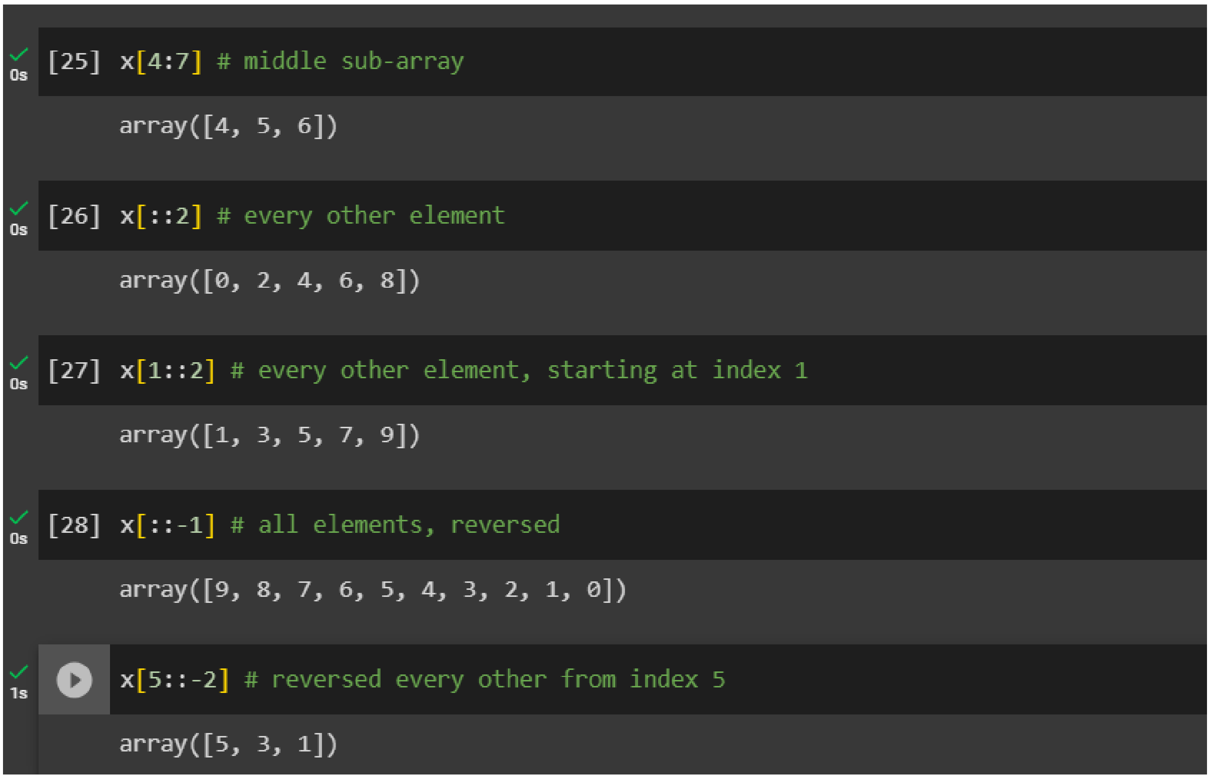
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
[23] x[:5] # first five elements
```

```
array([0, 1, 2, 3, 4])
```

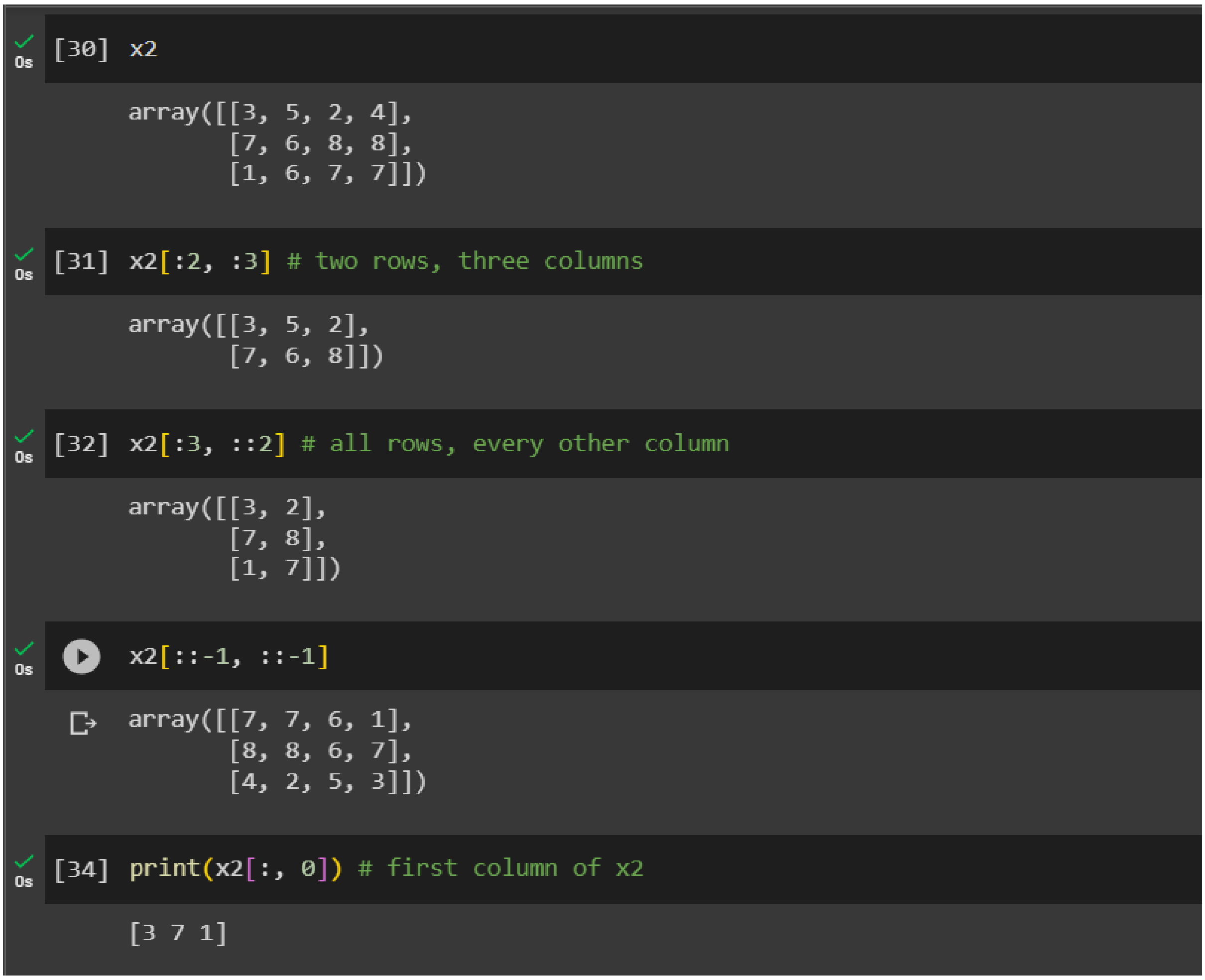
```
[24] x[5:] # elements after index 5
```

```
array([5, 6, 7, 8, 9])
```

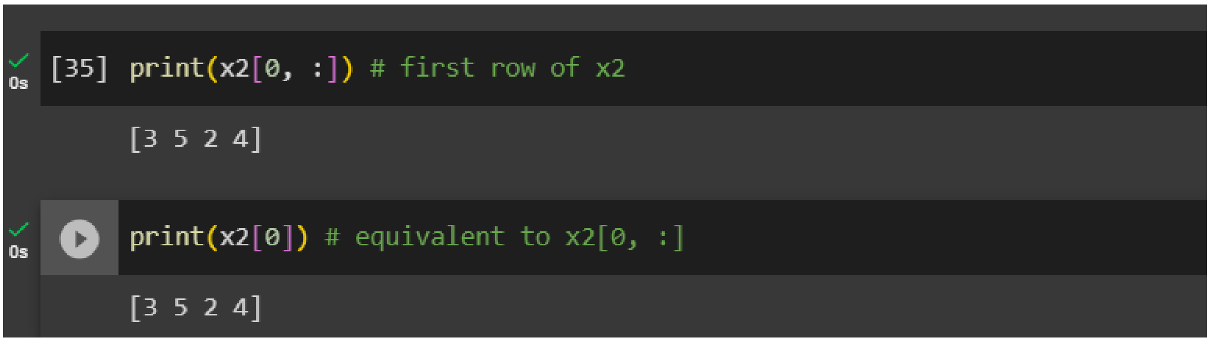


```
[25] x[4:7] # middle sub-array

     array([4, 5, 6])

[26] x[::2] # every other element

     array([0, 2, 4, 6, 8])

[27] x[1::2] # every other element, starting at index 1

     array([1, 3, 5, 7, 9])

[28] x[::-1] # all elements, reversed

     array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])

     x[5::-2] # reversed every other from index 5

     array([5, 3, 1])
```

Multi Dimensional Subarrays-



```
[30] x2

     array([[3, 5, 2, 4],
            [7, 6, 8, 8],
            [1, 6, 7, 7]])

[31] x2[:2, :3] # two rows, three columns

     array([[3, 5, 2],
            [7, 6, 8]])

[32] x2[:3, ::2] # all rows, every other column

     array([[3, 2],
            [7, 8],
            [1, 7]])

     x2[::-1, ::-1]

     array([[7, 7, 6, 1],
            [8, 8, 6, 7],
            [4, 2, 5, 3]])

[34] print(x2[:, 0]) # first column of x2

     [3 7 1]
```



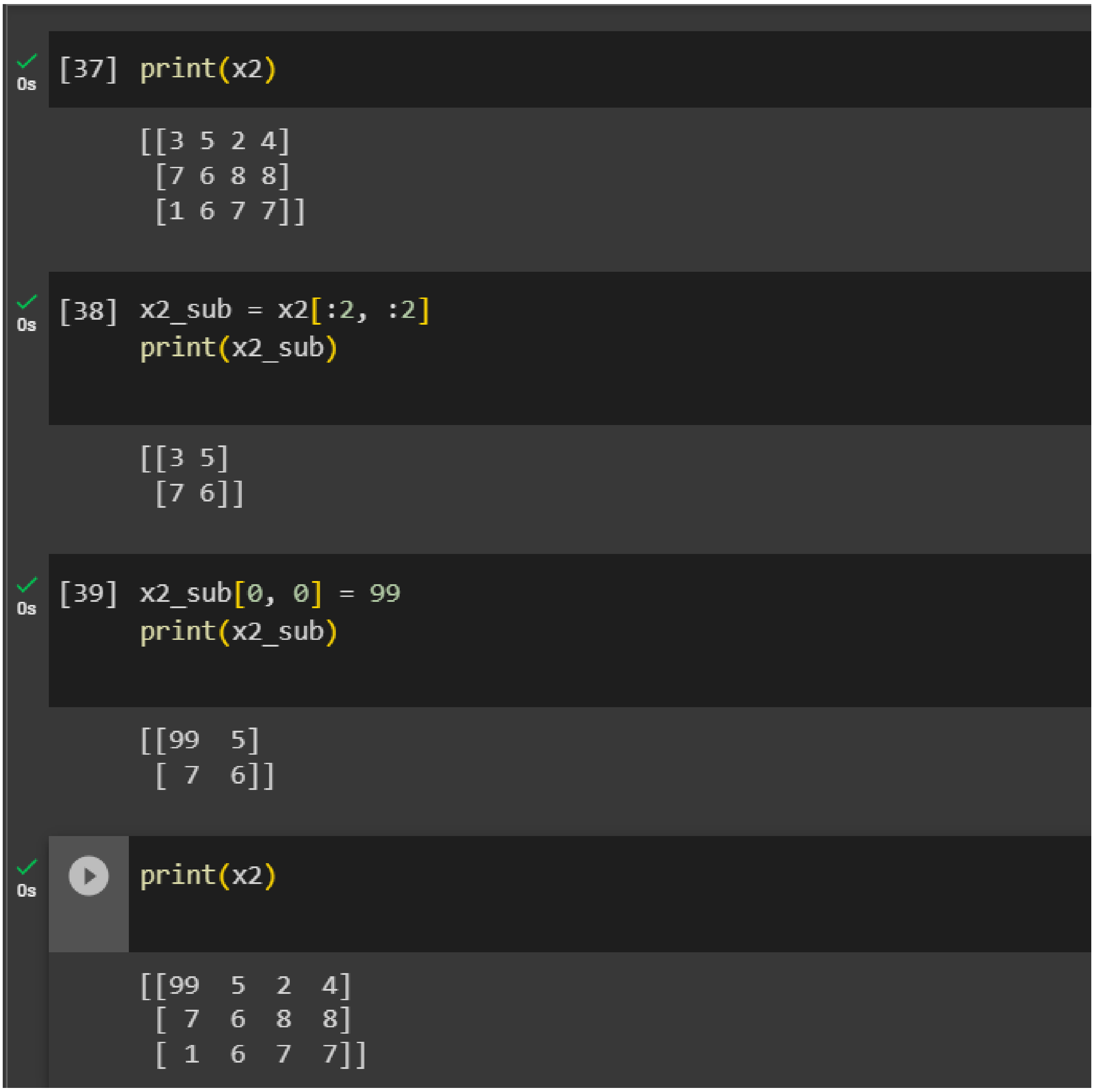
```
print(x2[0, :]) # first row of x2
```

```
[3 5 2 4]
```

```
print(x2[0]) # equivalent to x2[0, :]
```

```
[3 5 2 4]
```

Sub arrays as no copy views-



```
print(x2)
```

```
[[3 5 2 4]
 [7 6 8 8]
 [1 6 7 7]]
```

```
x2_sub = x2[:2, :2]
print(x2_sub)
```

```
[[3 5]
 [7 6]]
```

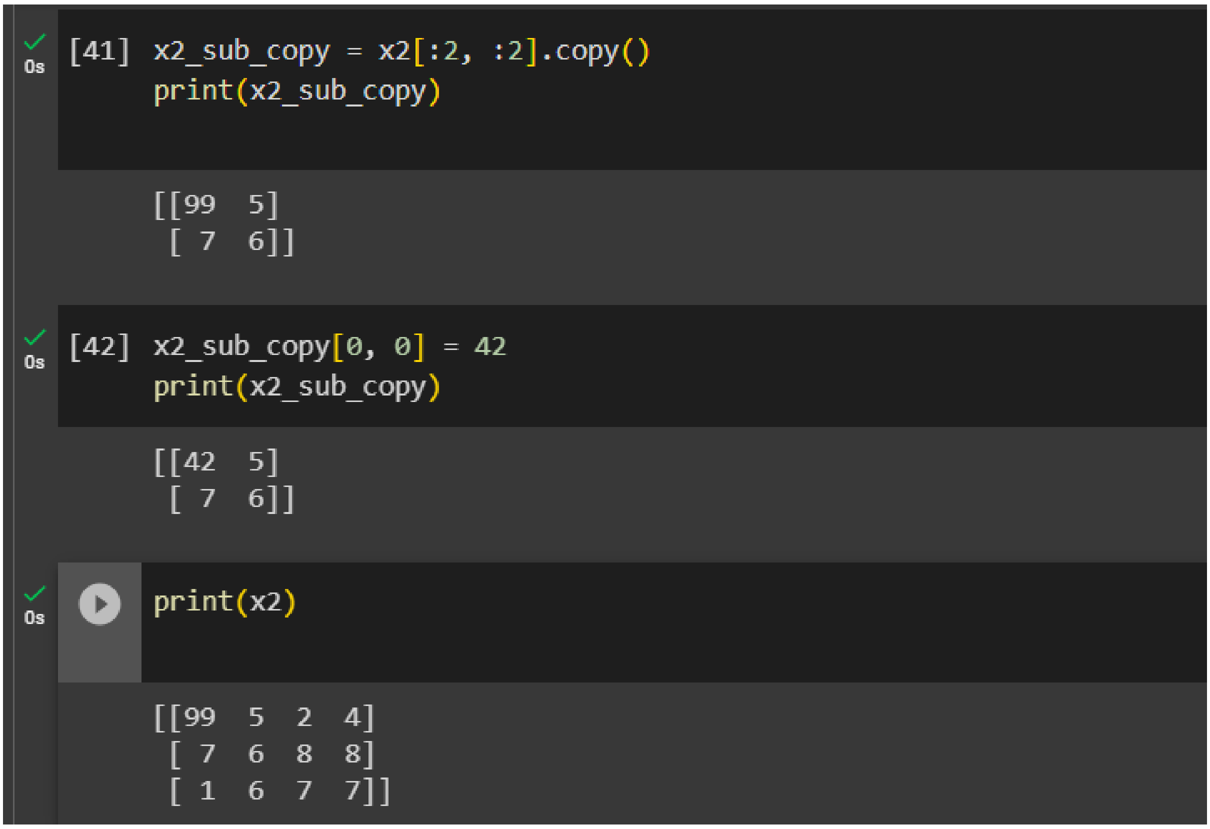
```
x2_sub[0, 0] = 99
print(x2_sub)
```

```
[[99  5]
 [ 7  6]]
```

```
print(x2)
```

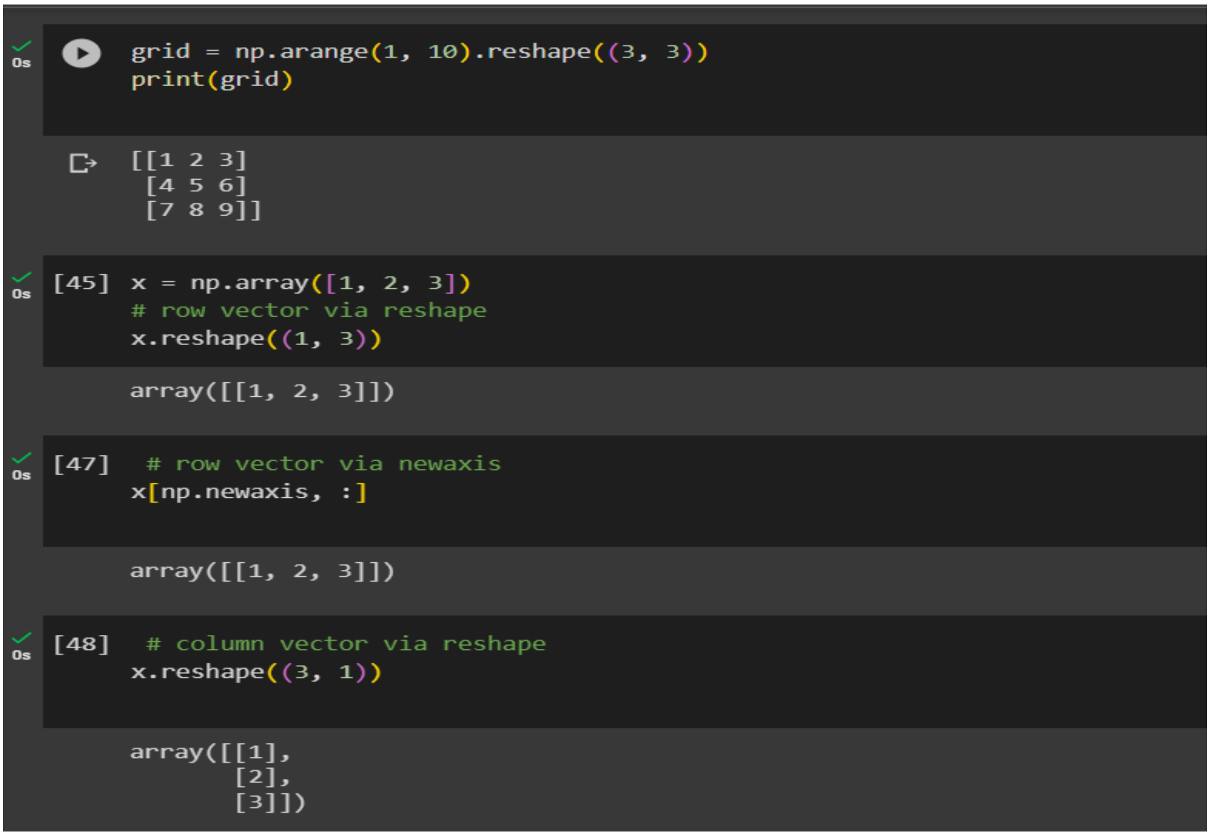
```
[[99  5  2  4]
 [ 7  6  8  8]
 [ 1  6  7  7]]
```

Creating copies of arrays-

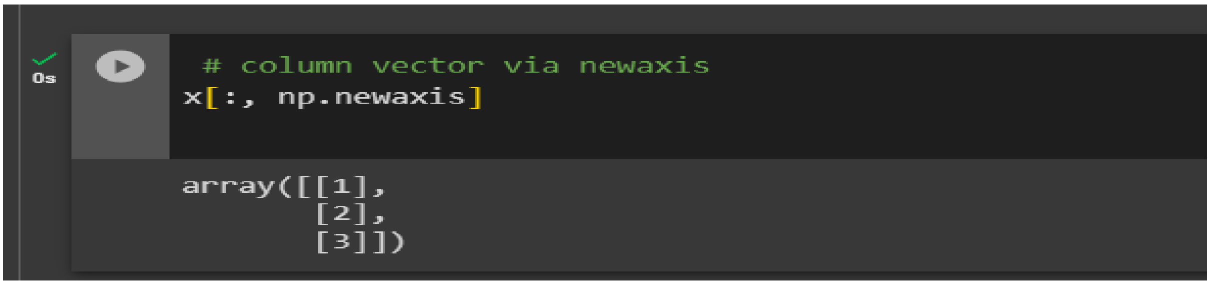


```
[41] x2_sub_copy = x2[:2, :2].copy()
     print(x2_sub_copy)

     [[99  5]
      [ 7  6]]

[42] x2_sub_copy[0, 0] = 42
     print(x2_sub_copy)

     [[42  5]
      [ 7  6]]

     print(x2)

     [[99  5  2  4]
      [ 7  6  8  8]
      [ 1  6  7  7]]
```

Reshaping of arrays-



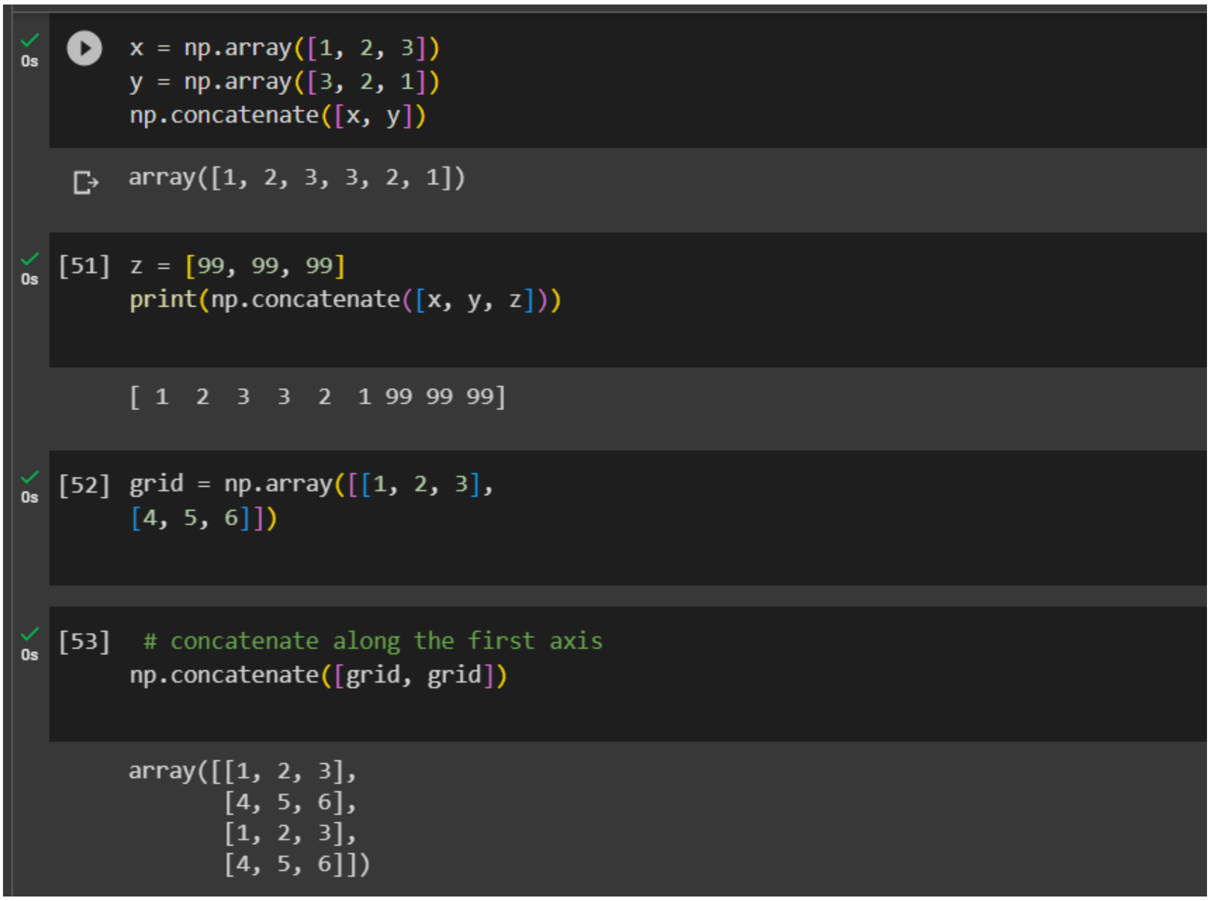
```
     grid = np.arange(1, 10).reshape((3, 3))
     print(grid)

     [[1 2 3]
      [4 5 6]
      [7 8 9]]

[45] x = np.array([1, 2, 3])
     # row vector via reshape
     x.reshape((1, 3))

     array([[1, 2, 3]])

[47]  # row vector via newaxis
     x[np.newaxis, :]

     array([[1, 2, 3]])

[48]  # column vector via reshape
     x.reshape((3, 1))

     array([[1],
            [2],
            [3]])
```



```
# column vector via newaxis
x[:, np.newaxis]
```

```
array([[1],
       [2],
       [3]])
```

Concatenation of arrays-



```
x = np.array([1, 2, 3])
y = np.array([3, 2, 1])
np.concatenate([x, y])
```

```
array([1, 2, 3, 3, 2, 1])
```

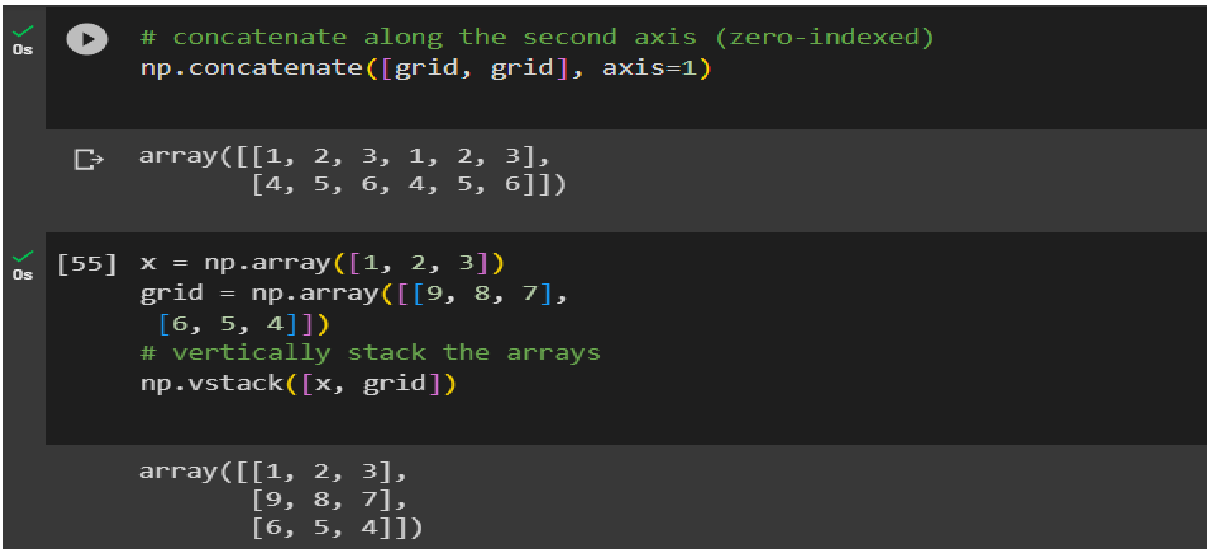
```
[51] z = [99, 99, 99]
     print(np.concatenate([x, y, z]))
```

```
[ 1  2  3  3  2  1 99 99 99]
```

```
[52] grid = np.array([[1, 2, 3],
     [4, 5, 6]])
```

```
[53]  # concatenate along the first axis
     np.concatenate([grid, grid])
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [1, 2, 3],
       [4, 5, 6]])
```

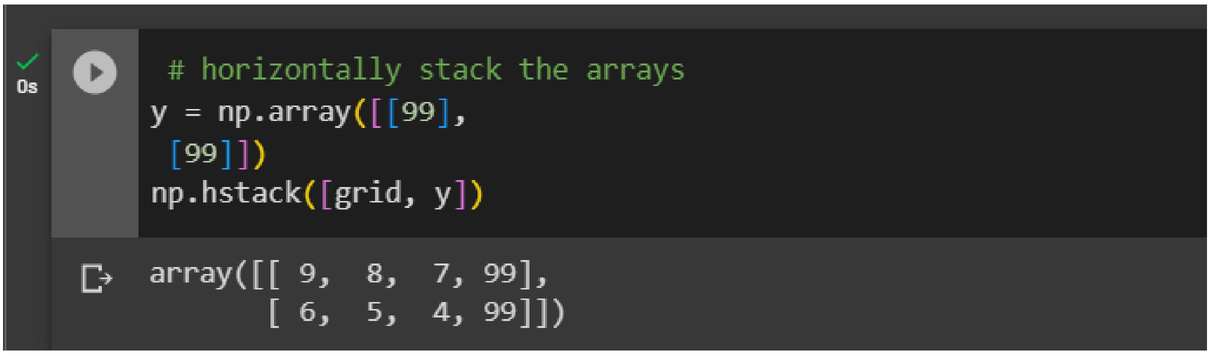


```
# concatenate along the second axis (zero-indexed)
np.concatenate([grid, grid], axis=1)
```

```
array([[1, 2, 3, 1, 2, 3],
       [4, 5, 6, 4, 5, 6]])
```

```
[55] x = np.array([1, 2, 3])
     grid = np.array([[9, 8, 7],
      [6, 5, 4]])
     # vertically stack the arrays
     np.vstack([x, grid])
```

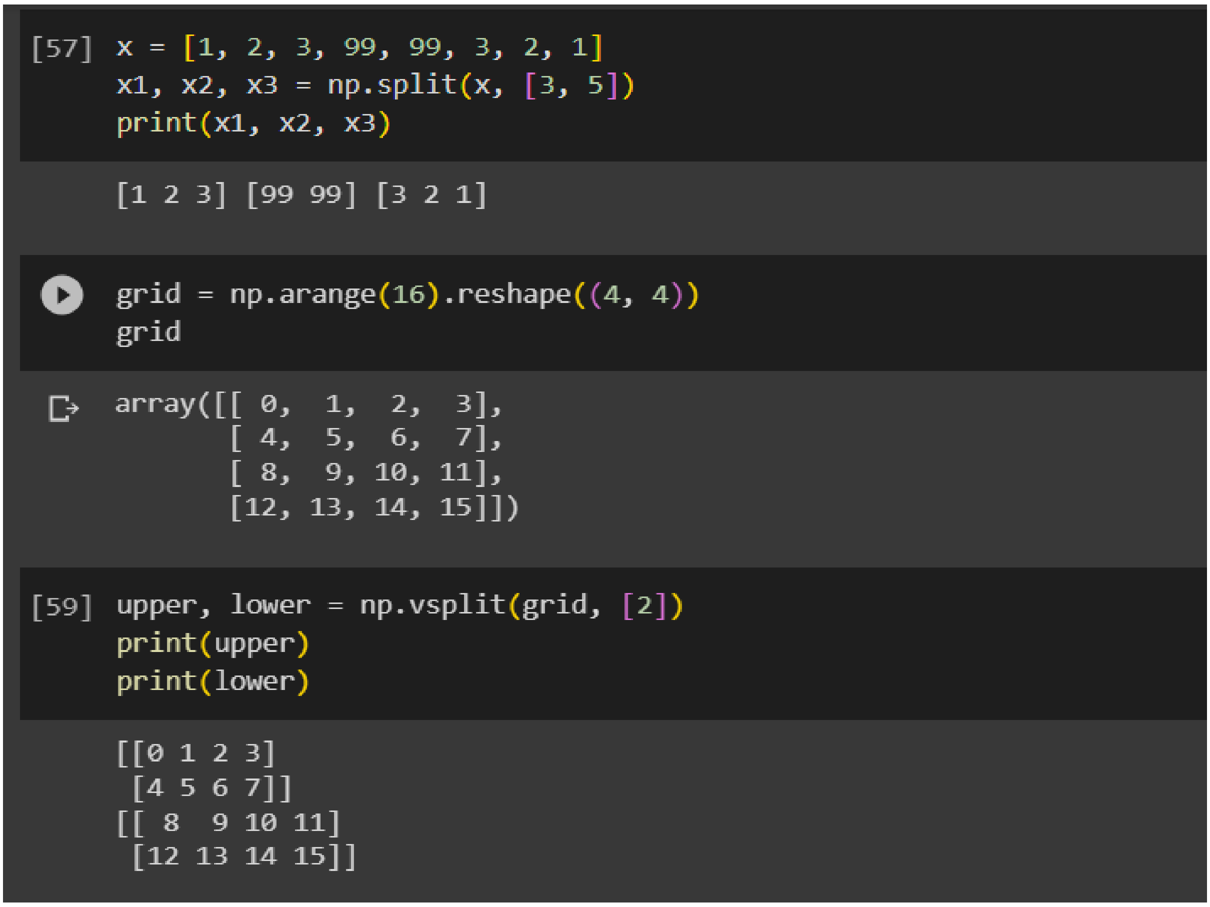
```
array([[1, 2, 3],
       [9, 8, 7],
       [6, 5, 4]])
```



```
# horizontally stack the arrays
y = np.array([[99],
 [99]])
np.hstack([grid, y])
```

```
array([[ 9,  8,  7, 99],
       [ 6,  5,  4, 99]])
```

Splitting of arrays-



```
x = [1, 2, 3, 99, 99, 3, 2, 1]
x1, x2, x3 = np.split(x, [3, 5])
print(x1, x2, x3)
```

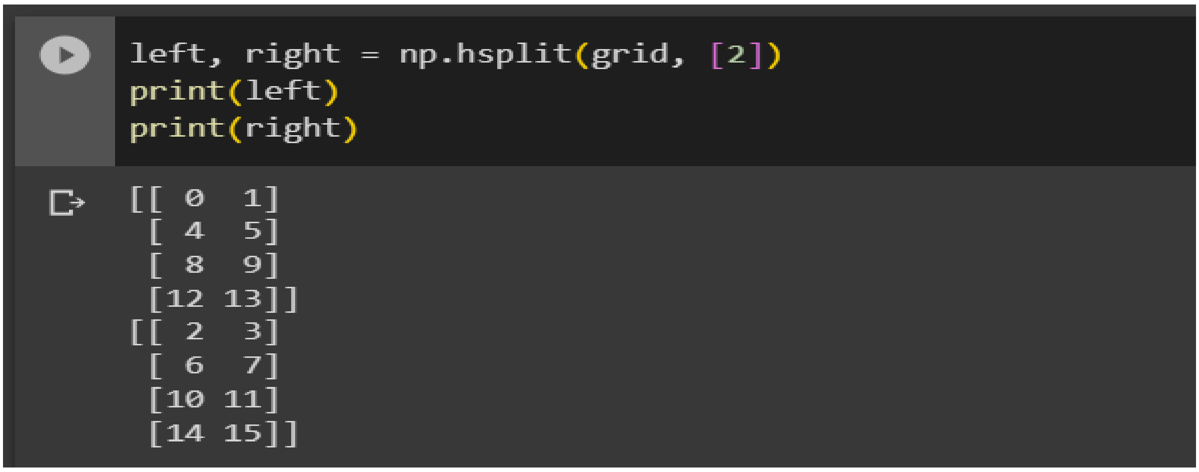
```
[1 2 3] [99 99] [3 2 1]
```

```
grid = np.arange(16).reshape((4, 4))
grid
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

```
upper, lower = np.vsplit(grid, [2])
print(upper)
print(lower)
```

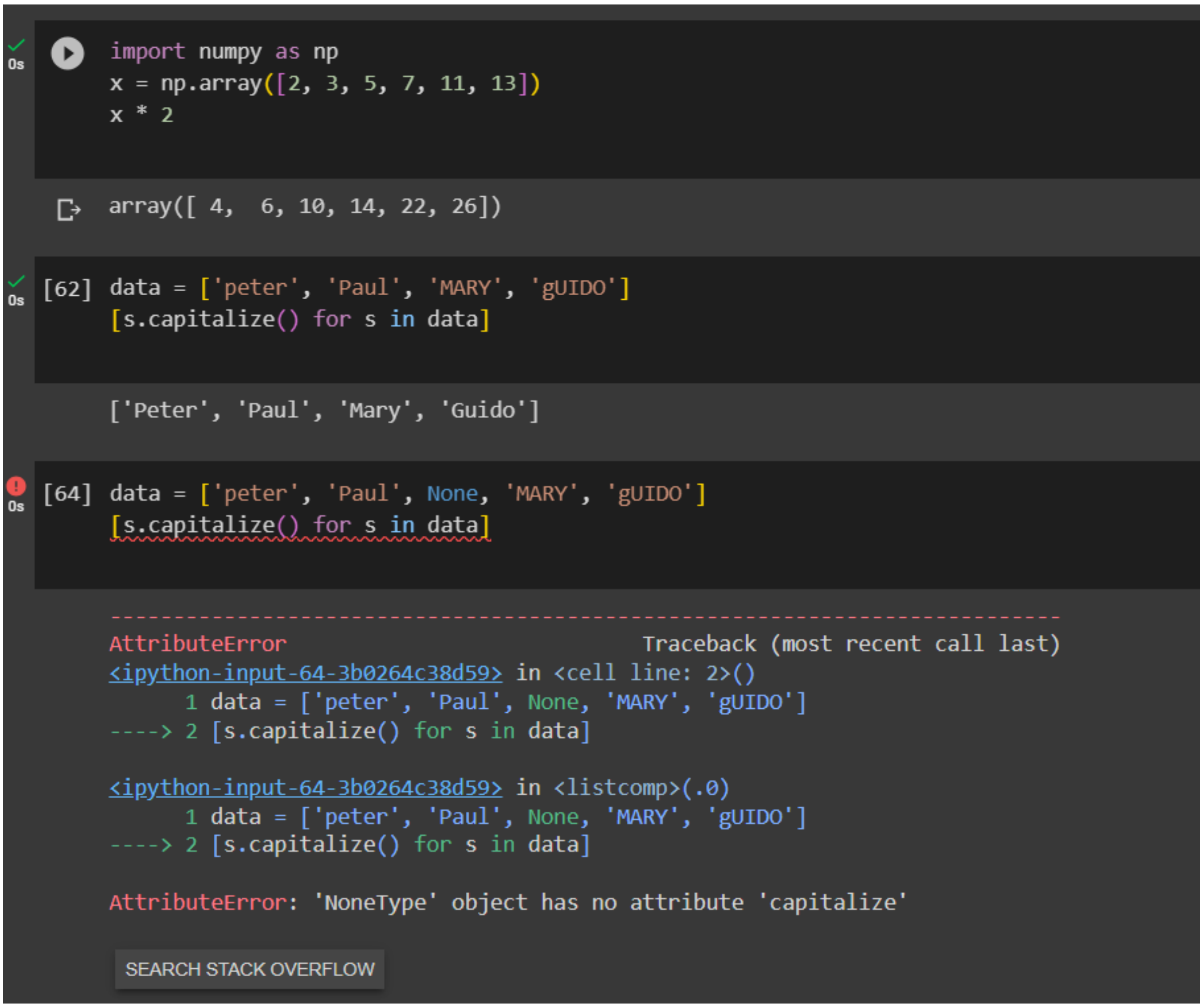
```
[[0 1 2 3]
 [4 5 6 7]]
[[ 8  9 10 11]
 [12 13 14 15]]
```



```
left, right = np.hsplit(grid, [2])
print(left)
print(right)
```

```
[[ 0  1]
 [ 4  5]
 [ 8  9]
 [12 13]]
[[ 2  3]
 [ 6  7]
 [10 11]
 [14 15]]
```

# Introducing Pandas String Operations-



```
import numpy as np
x = np.array([2, 3, 5, 7, 11, 13])
x * 2
```

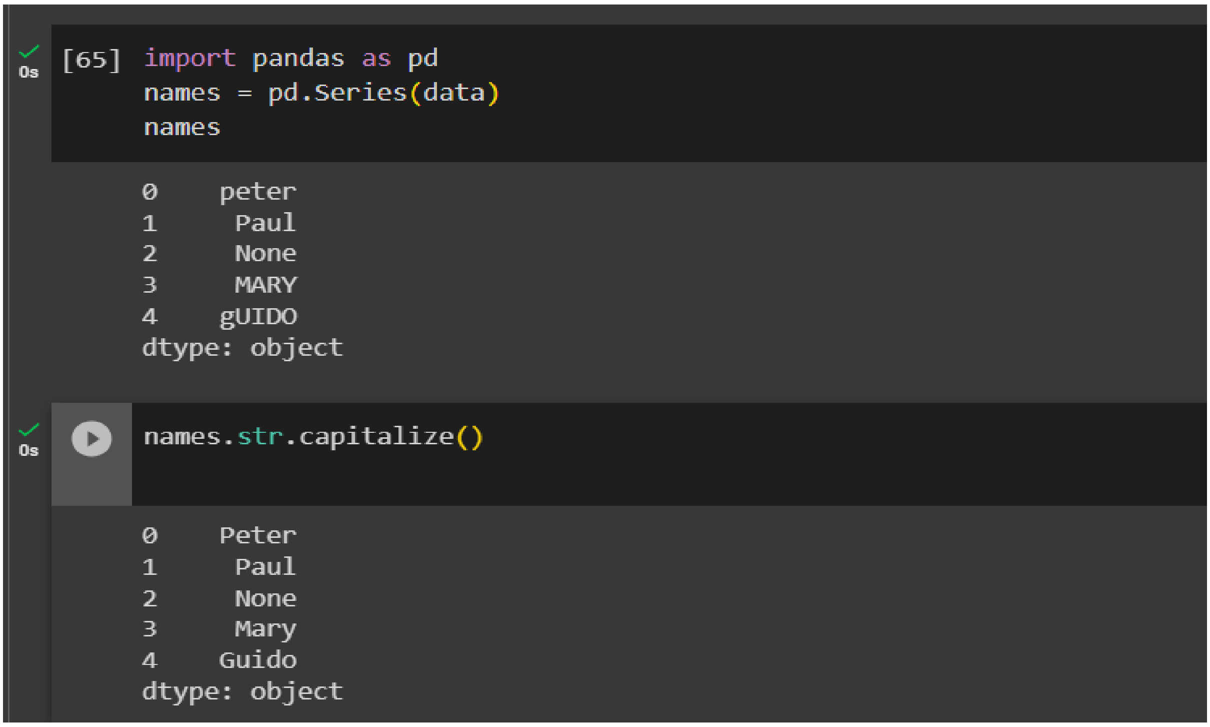
```
array([ 4,  6, 10, 14, 22, 26])
```

```
[62] data = ['peter', 'Paul', 'MARY', 'gUIDO']
     [s.capitalize() for s in data]
```

```
['Peter', 'Paul', 'Mary', 'Guido']
```

```
[64] data = ['peter', 'Paul', None, 'MARY', 'gUIDO']
     [s.capitalize() for s in data]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-64-3b0264c38d59> in <cell line: 2>()
      1 data = ['peter', 'Paul', None, 'MARY', 'gUIDO']
----> 2 [s.capitalize() for s in data]

<ipython-input-64-3b0264c38d59> in <listcomp>(.0)
      1 data = ['peter', 'Paul', None, 'MARY', 'gUIDO']
----> 2 [s.capitalize() for s in data]

AttributeError: 'NoneType' object has no attribute 'capitalize'
```



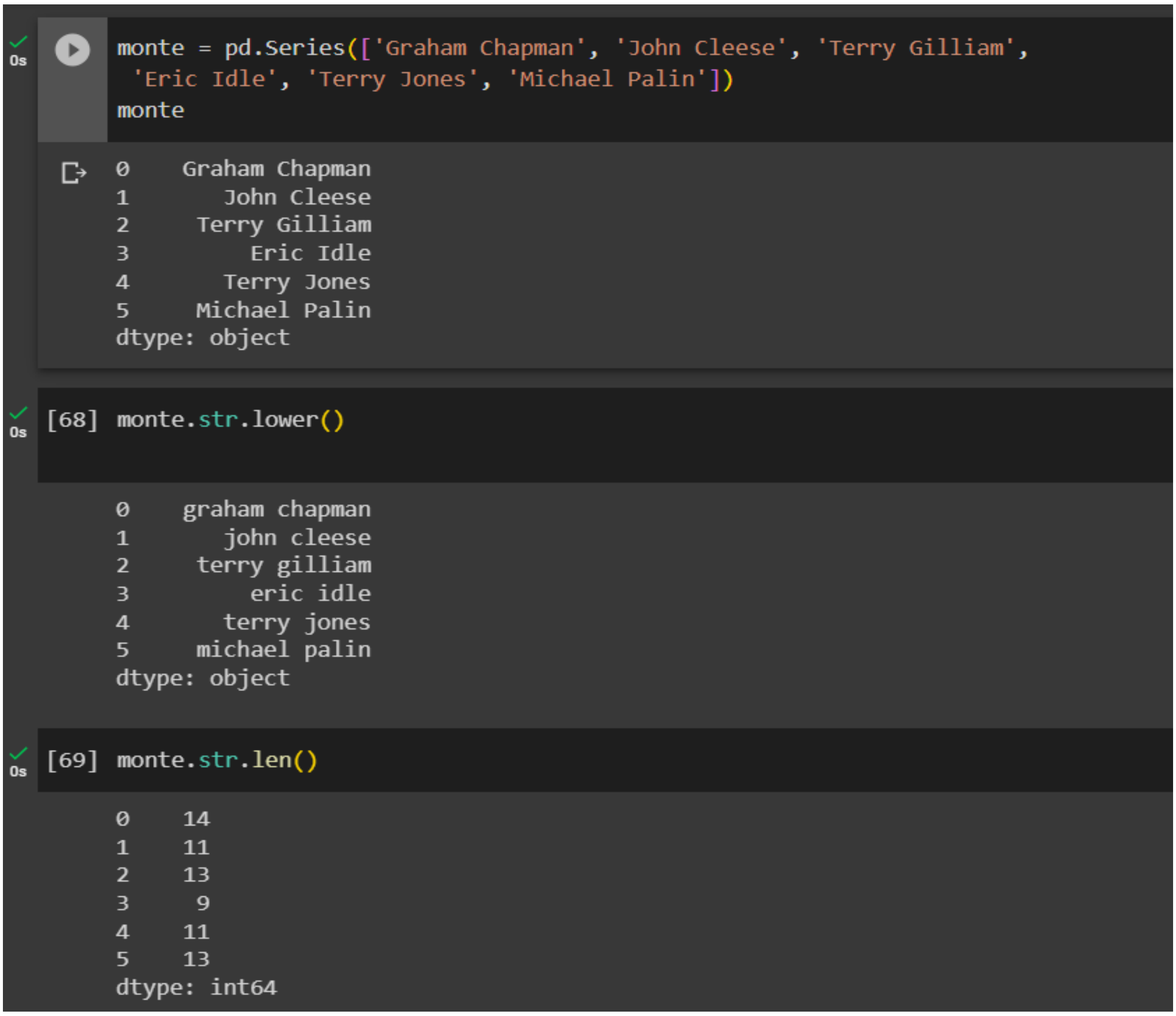
```
[65] import pandas as pd
     names = pd.Series(data)
     names
```

```
0    peter
1     Paul
2     None
3     MARY
4    gUIDO
dtype: object
```

```
names.str.capitalize()
```

```
0    Peter
1     Paul
2     None
3     Mary
4    Guido
dtype: object
```

# Tables of Pandas String Methods-



```
monte = pd.Series(['Graham Chapman', 'John Cleese', 'Terry Gilliam',
 'Eric Idle', 'Terry Jones', 'Michael Palin'])
monte
```

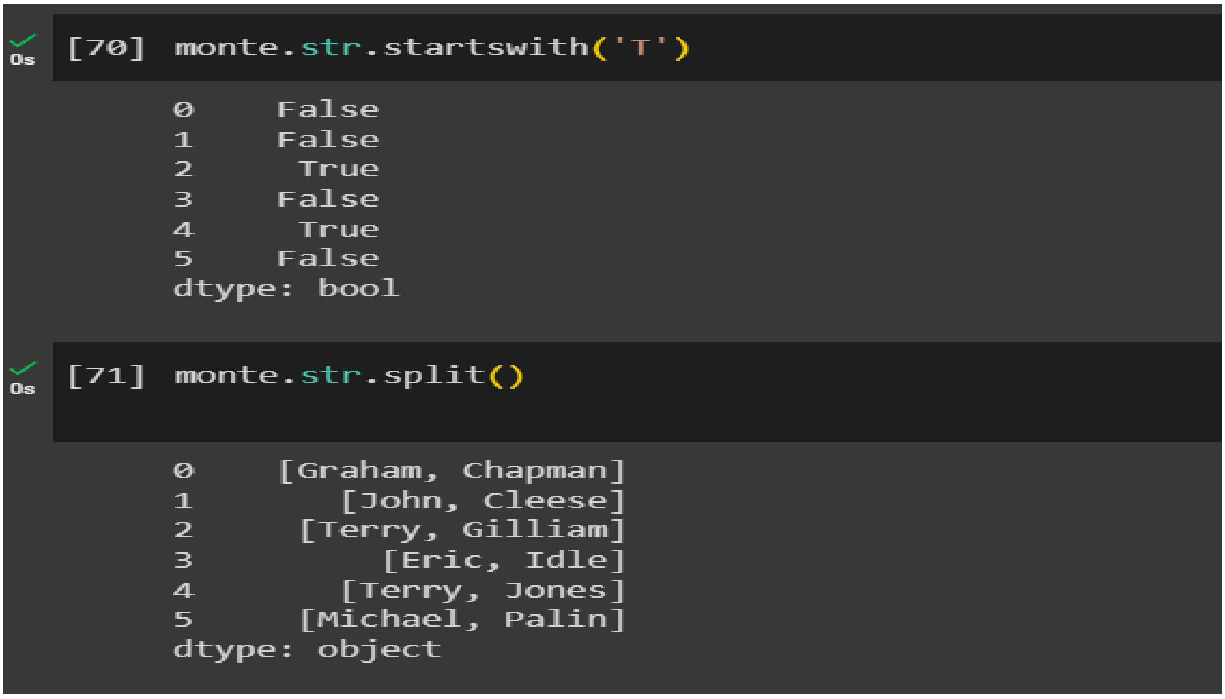
```
0    Graham Chapman
1       John Cleese
2     Terry Gilliam
3         Eric Idle
4       Terry Jones
5     Michael Palin
dtype: object
```

```
[68] monte.str.lower()
```

```
0    graham chapman
1       john cleese
2     terry gilliam
3         eric idle
4       terry jones
5     michael palin
dtype: object
```

```
[69] monte.str.len()
```

```
0    14
1    11
2    13
3     9
4    11
5    13
dtype: int64
```



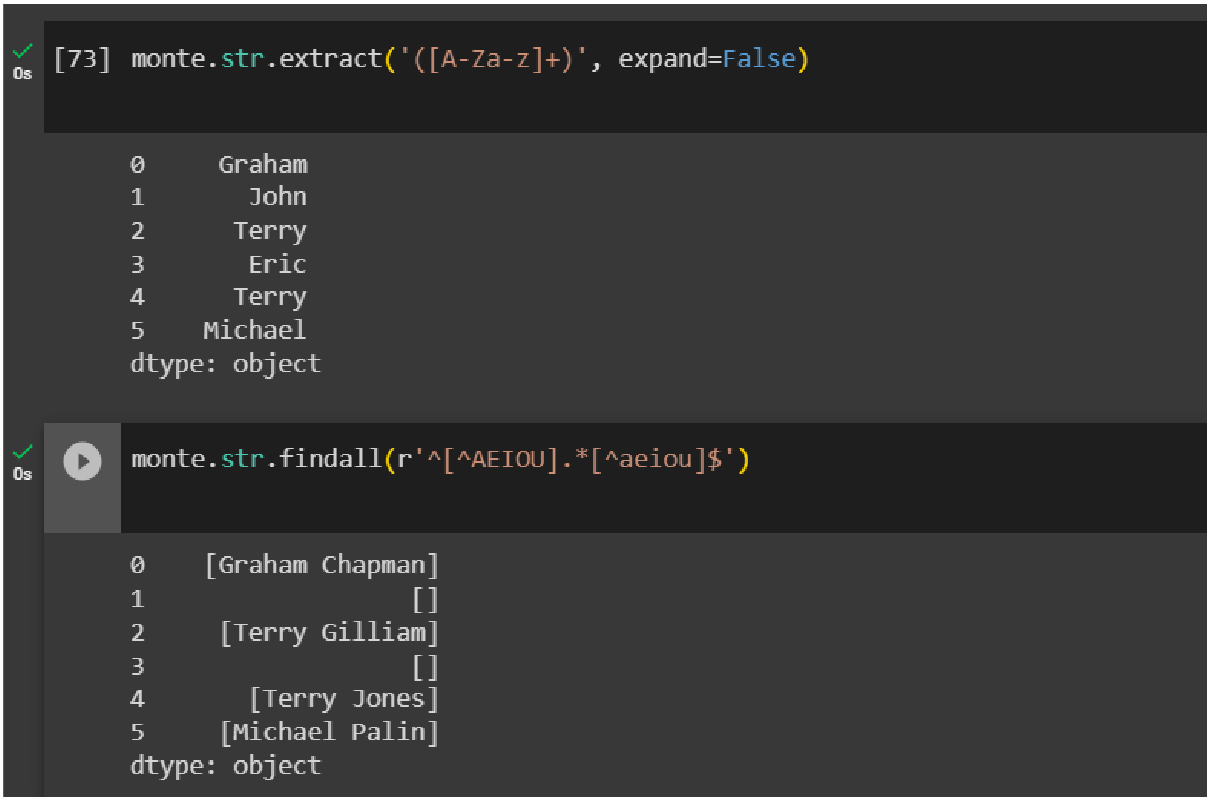
```
[70] monte.str.startswith('T')
```

```
0    False
1    False
2     True
3    False
4     True
5    False
dtype: bool
```

```
[71] monte.str.split()
```

```
0    [Graham, Chapman]
1       [John, Cleese]
2     [Terry, Gilliam]
3         [Eric, Idle]
4       [Terry, Jones]
5     [Michael, Palin]
dtype: object
```

## Methods using Regular Expressions-



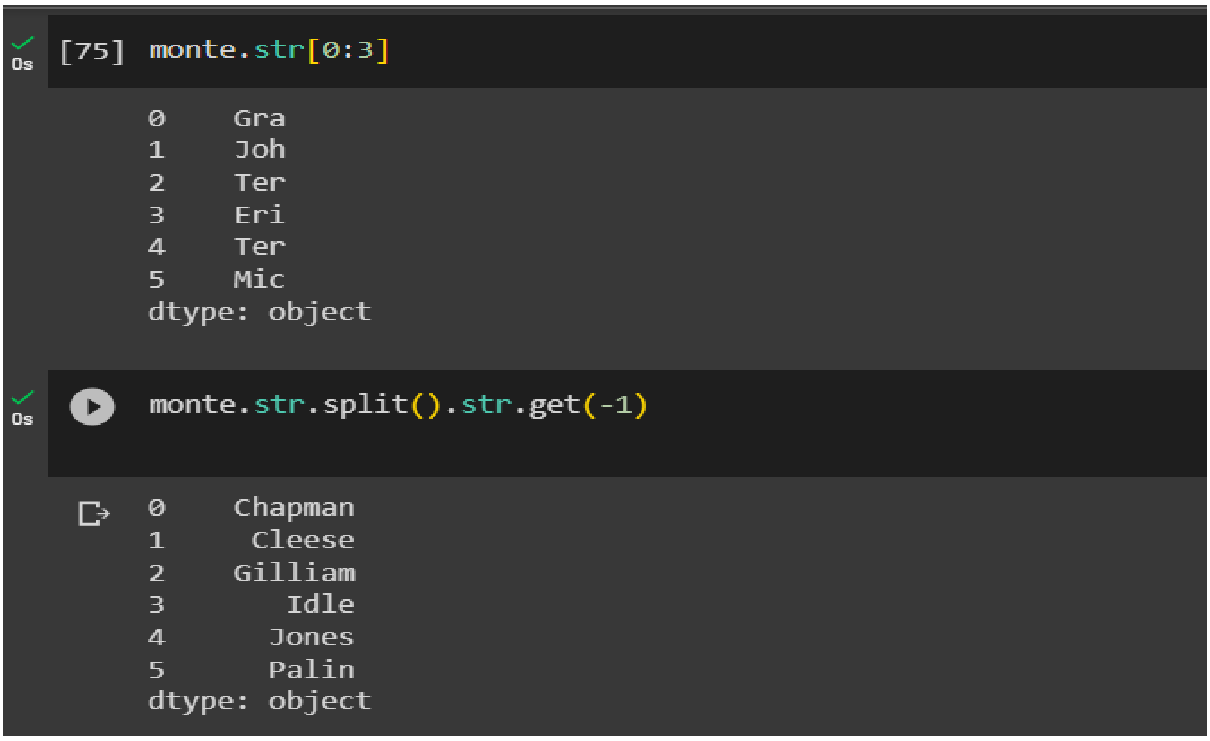
```
[73] monte.str.extract('([A-Za-z]+)', expand=False)
```

```
0     Graham
1       John
2      Terry
3       Eric
4      Terry
5    Michael
dtype: object
```

```
monte.str.findall(r'^[^AEIOU].*[^aeiou]$')
```

```
0    [Graham Chapman]
1                  []
2     [Terry Gilliam]
3                  []
4       [Terry Jones]
5     [Michael Palin]
dtype: object
```

## Miscellaneous Methods-

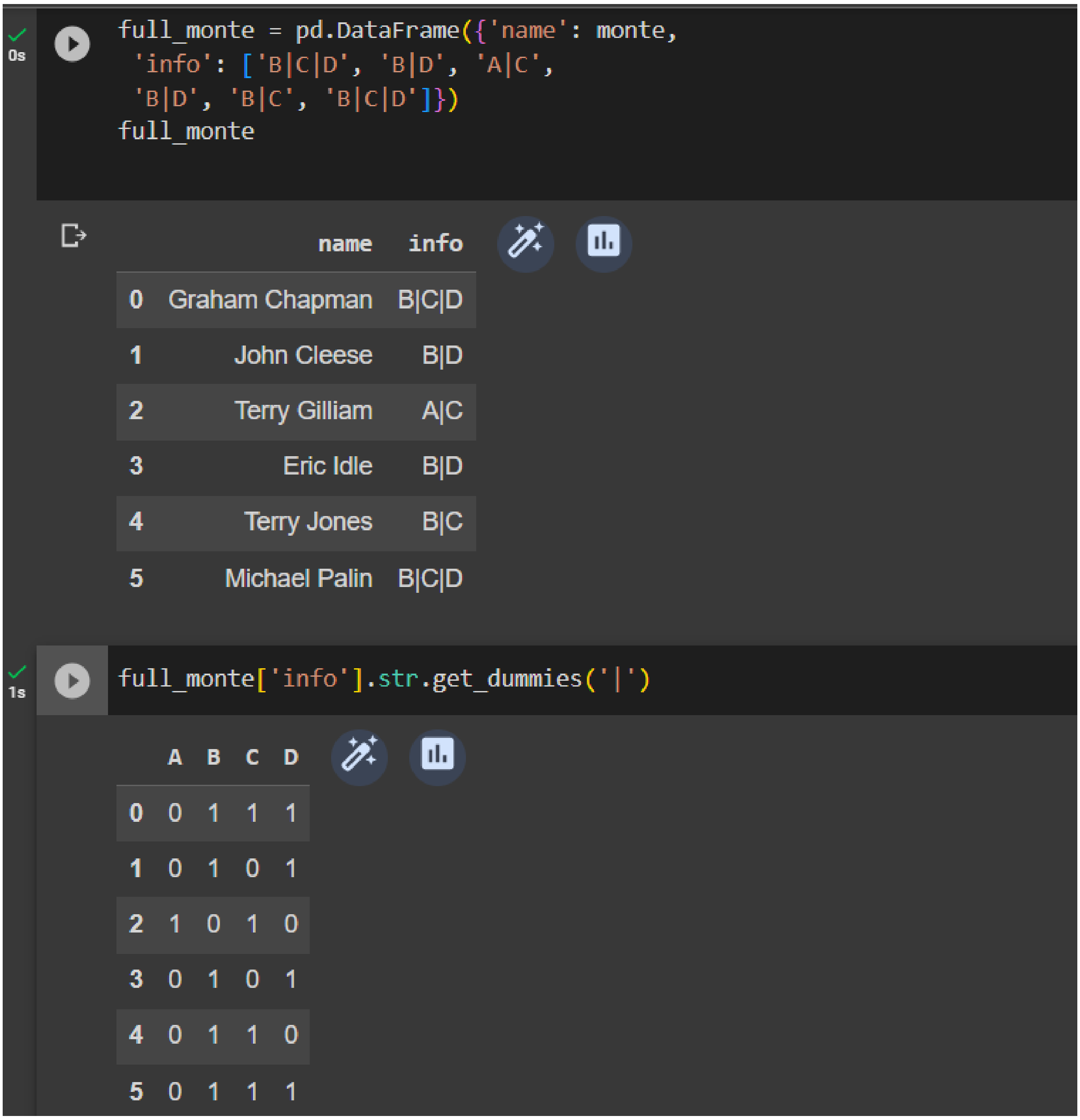


```
[75] monte.str[0:3]
```

```
0    Gra
1    Joh
2    Ter
3    Eri
4    Ter
5    Mic
dtype: object
```

```
monte.str.split().str.get(-1)
```

```
0    Chapman
1     Cleese
2    Gilliam
3       Idle
4      Jones
5      Palin
dtype: object
```



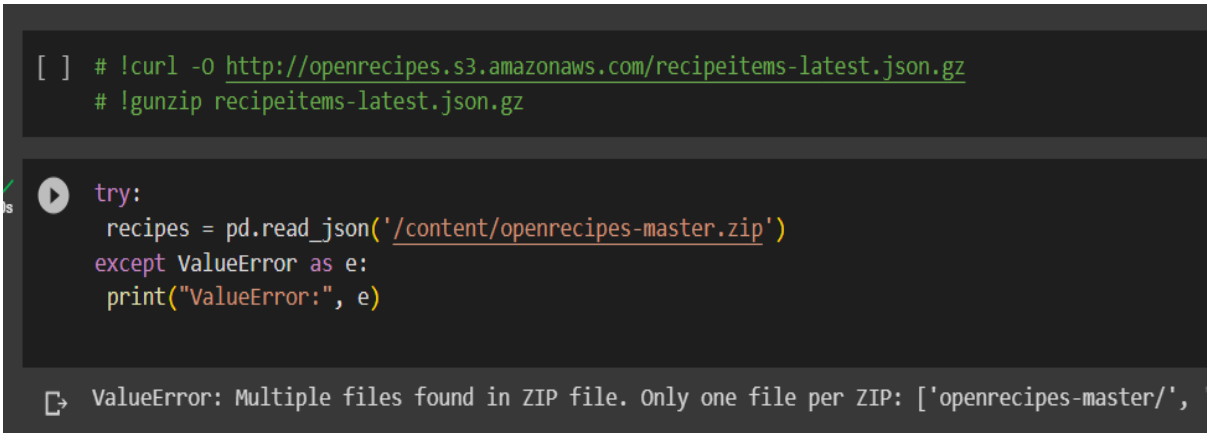
```
full_monte = pd.DataFrame({'name': monte,
 'info': ['B|C|D', 'B|D', 'A|C',
 'B|D', 'B|C', 'B|C|D']})
full_monte
```

|   | name | info |
|---|---|---|
| 0 | Graham Chapman | B\|C\|D |
| 1 | John Cleese | B\|D |
| 2 | Terry Gilliam | A\|C |
| 3 | Eric Idle | B\|D |
| 4 | Terry Jones | B\|C |
| 5 | Michael Palin | B\|C\|D |

```
full_monte['info'].str.get_dummies('|')
```

|   | A | B | C | D |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 2 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 1 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 1 | 1 |

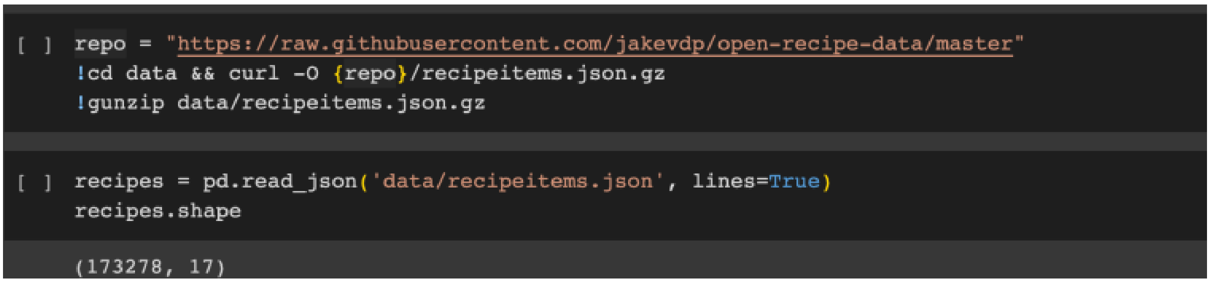
## Recipe Database-



```
[ ] # !curl -O http://openrecipes.s3.amazonaws.com/recipeitems-latest.json.gz
    # !gunzip recipeitems-latest.json.gz
```

```
try:
  recipes = pd.read_json('/content/openrecipes-master.zip')
except ValueError as e:
  print("ValueError:", e)
```

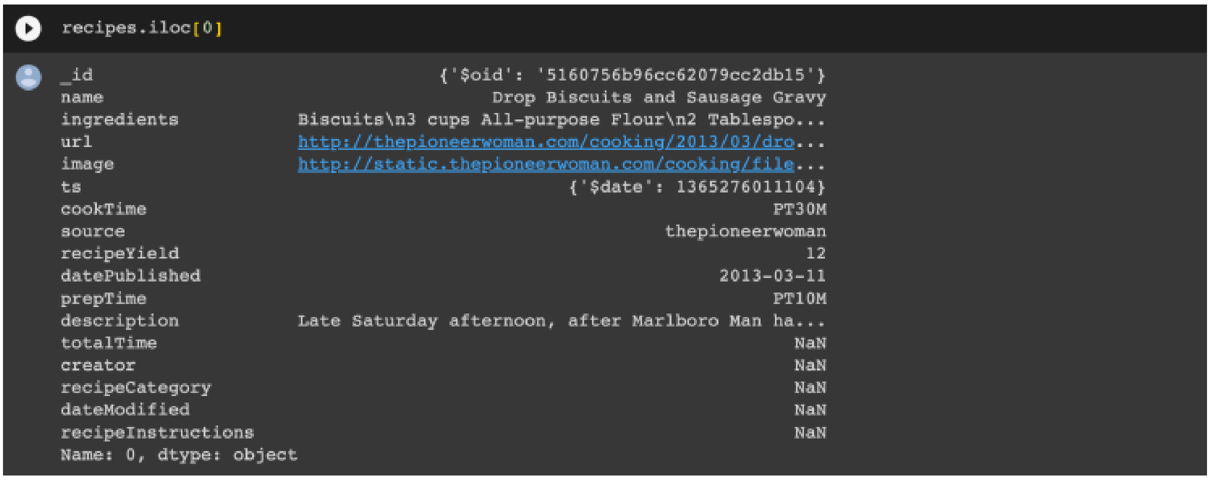
```
ValueError: Multiple files found in ZIP file. Only one file per ZIP: ['openrecipes-master/',
```



```
repo = "https://raw.githubusercontent.com/jakevdp/open-recipe-data/master"
!cd data && curl -O {repo}/recipeitems.json.gz
!gunzip data/recipeitems.json.gz
```

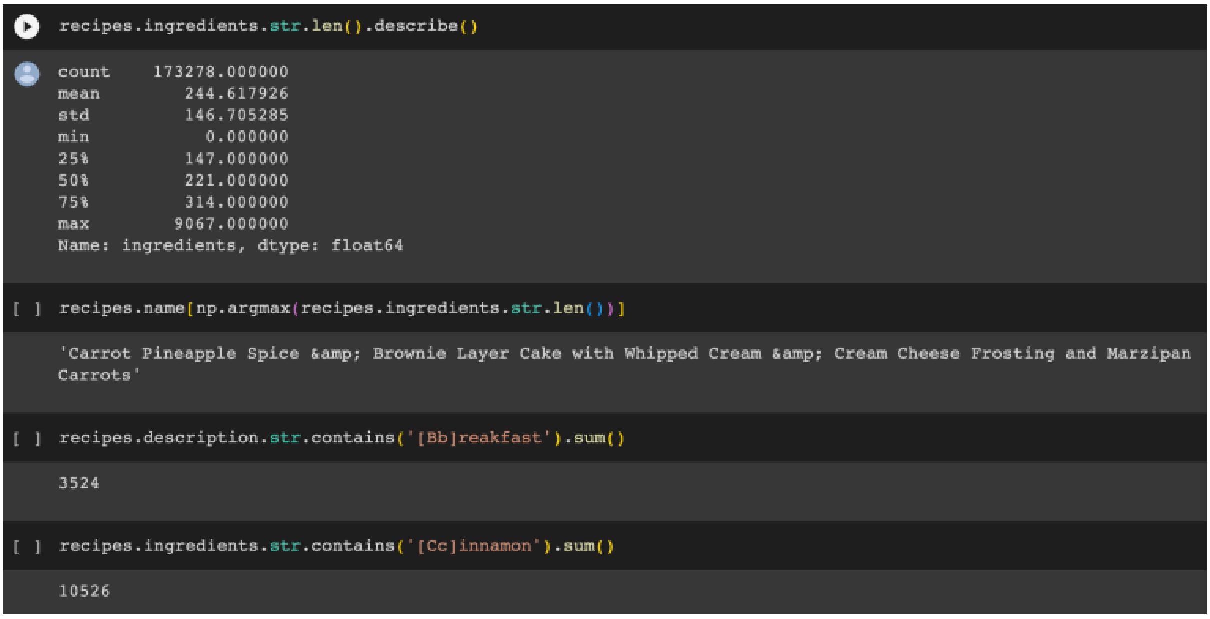
```
recipes = pd.read_json('data/recipeitems.json', lines=True)
recipes.shape
```

```
(173278, 17)
```



```
recipes.iloc[0]
```

```
_id                                {'$oid': '5160756b96cc62079cc2db15'}
name                                    Drop Biscuits and Sausage Gravy
ingredients           Biscuits\n3 cups All-purpose Flour\n2 Tablespo...
url                   http://thepioneerwoman.com/cooking/2013/03/dro...
image                 http://static.thepioneerwoman.com/cooking/file...
ts                                     {'$date': 1365276011104}
cookTime                                                          PT30M
source                                                  thepioneerwoman
recipeYield                                                          12
datePublished                                                2013-03-11
prepTime                                                          PT10M
description           Late Saturday afternoon, after Marlboro Man ha...
totalTime                                                           NaN
creator                                                             NaN
recipeCategory                                                      NaN
dateModified                                                        NaN
recipeInstructions                                                  NaN
Name: 0, dtype: object
```



```
recipes.ingredients.str.len().describe()
```

```
count    173278.000000
mean        244.617926
std         146.705285
min           0.000000
25%         147.000000
50%         221.000000
75%         314.000000
max        9067.000000
Name: ingredients, dtype: float64
```

```
recipes.name[np.argmax(recipes.ingredients.str.len())]
```

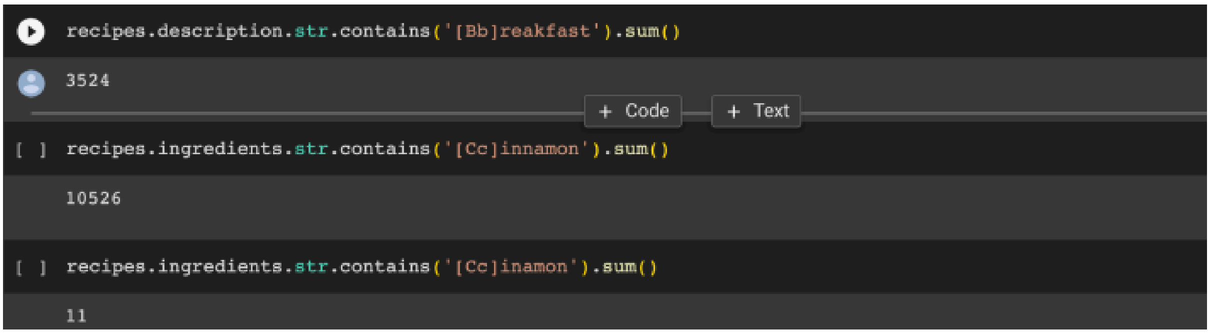
```
'Carrot Pineapple Spice &amp; Brownie Layer Cake with Whipped Cream &amp; Cream Cheese Frosting and Marzipan
Carrots'
```

```
recipes.description.str.contains('[Bb]reakfast').sum()
```

```
3524
```

```
recipes.ingredients.str.contains('[Cc]innamon').sum()
```

```
10526
```



```
recipes.description.str.contains('[Bb]reakfast').sum()
```

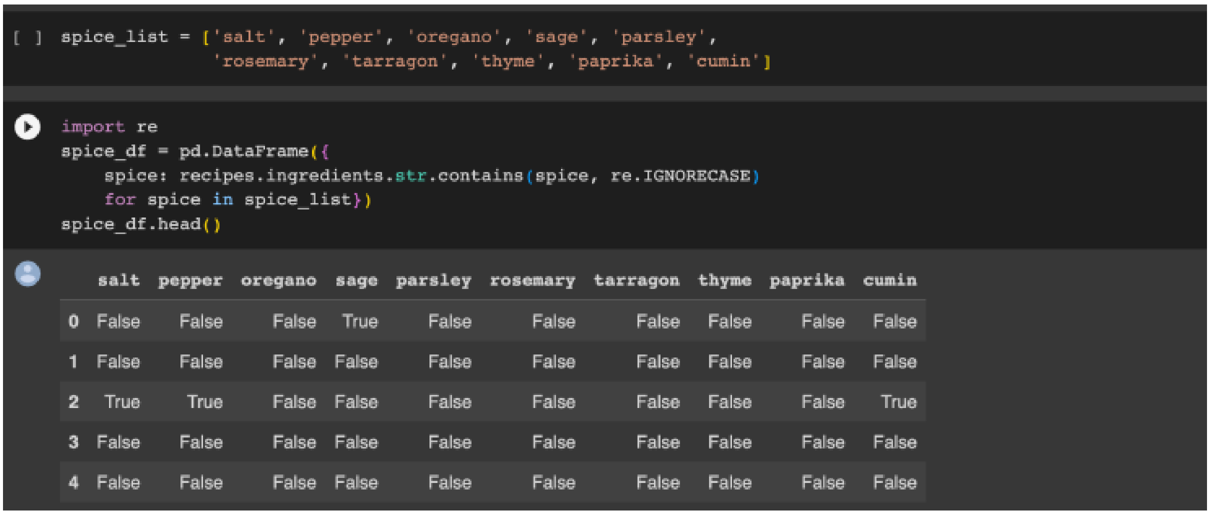
```
3524
```

```
recipes.ingredients.str.contains('[Cc]innamon').sum()
```

```
10526
```

```
recipes.ingredients.str.contains('[Cc]inamon').sum()
```

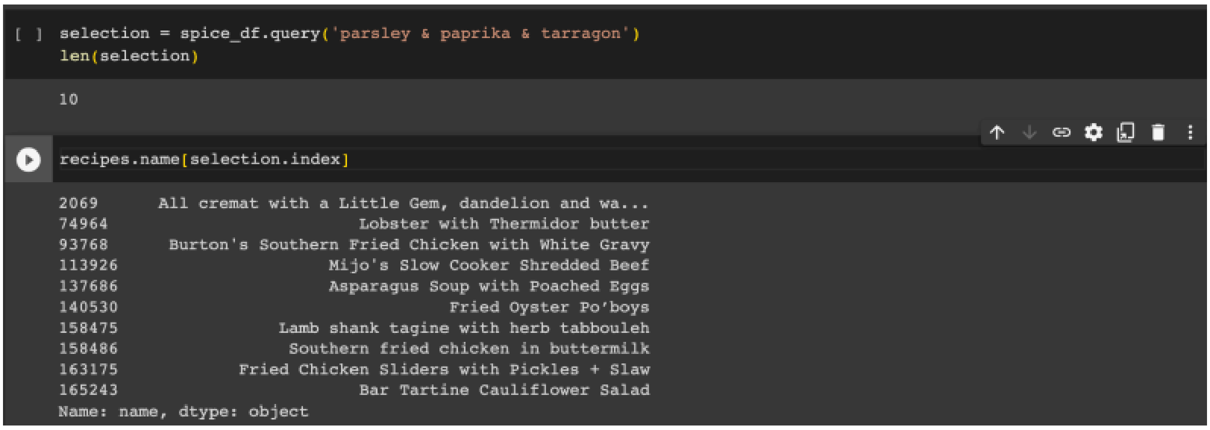
```
11
```



```
[ ] spice_list = ['salt', 'pepper', 'oregano', 'sage', 'parsley',
                  'rosemary', 'tarragon', 'thyme', 'paprika', 'cumin']
```

```
import re
spice_df = pd.DataFrame({
    spice: recipes.ingredients.str.contains(spice, re.IGNORECASE)
    for spice in spice_list})
spice_df.head()
```

|   | salt | pepper | oregano | sage | parsley | rosemary | tarragon | thyme | paprika | cumin |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | False | False | False | True | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False | False | False | False | False |
| 2 | True | True | False | False | False | False | False | False | False | True |
| 3 | False | False | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False | False | False |



```
[ ] selection = spice_df.query('parsley & paprika & tarragon')
    len(selection)

    10
```

```
recipes.name[selection.index]

2069      All cremat with a Little Gem, dandelion and wa...
74964                        Lobster with Thermidor butter
93768         Burton's Southern Fried Chicken with White Gravy
113926                     Mijo's Slow Cooker Shredded Beef
137686                     Asparagus Soup with Poached Eggs
140530                                 Fried Oyster Po'boys
158475                      Lamb shank tagine with herb tabbouleh
158486                       Southern fried chicken in buttermilk
163175                   Fried Chicken Sliders with Pickles + Slaw
165243                          Bar Tartine Cauliflower Salad
Name: name, dtype: object
```

-------------------------------------X-----------------------------------------

Thank you!