



Alliance School of Advanced Computing

Department of Computer Science and Engineering

Class Assignment-1

Course Code: 5CS1025

Course Title: Artificial Intelligence

Semester: 04

Class : AIML

2024-25

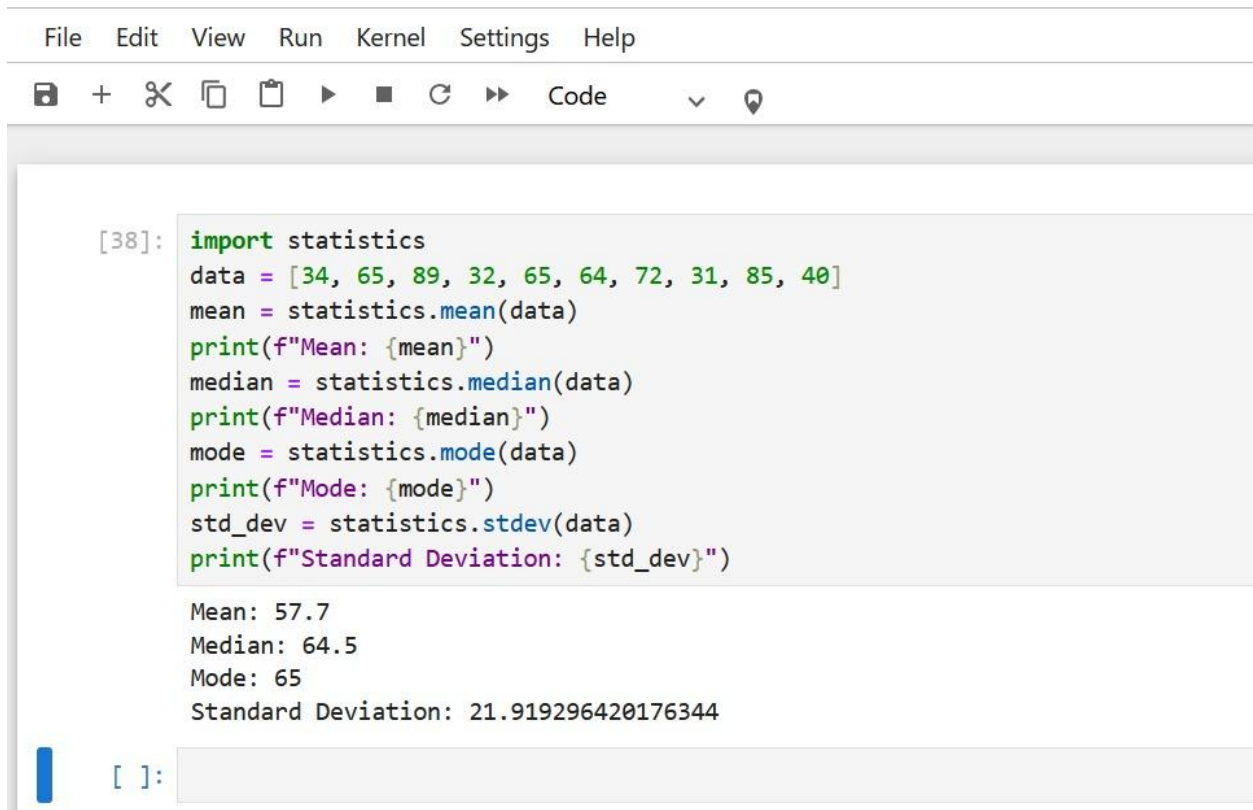
Name : KAMMARA VAMSHI VARDHAN

Reg.NO :2023BCSE07AED241

1. Imagine you are tasked with designing a humanoid robot to assist in a home or office environment. The robot must be capable of interacting with people by **talking** and **listening**, **walking** to different locations, **seeing** and recognizing objects, and **learning** from its surroundings to adapt its behavior. **What technologies, tools, and frameworks would you need to build such a robot?** Give as flow chart

Frameworks	Tools	Technologies
Speech	Talking & Listening	ASR, TTS Dialog flow
Vision	Seeing & Recognizing Objects	Cameras depth sensors AI-powered object detection.
Mobility	Walking & Navigation	Motors, actuators, IMU sensors, SLAM-based navigation.
AI & Learning	Adapting to Environment	Machine learning Tensorflow Pytorch
Control System	Integration & Safety	Middleware like ROS

2. Calculate and interpret mean, median, mode, variance and standard deviation for a given dataset.
Data=[15,21,29,21,15,24,32,21,15,30



The screenshot shows a Jupyter Notebook interface. At the top is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, copying, pasting, running, and other functions. The main area contains a code cell with the following Python code:

```
[38]: import statistics
data = [34, 65, 89, 32, 65, 64, 72, 31, 85, 40]
mean = statistics.mean(data)
print(f"Mean: {mean}")
median = statistics.median(data)
print(f"Median: {median}")
mode = statistics.mode(data)
print(f"Mode: {mode}")
std_dev = statistics.stdev(data)
print(f"Standard Deviation: {std_dev}")
```

Below the code cell, the output is displayed:

```
Mean: 57.7
Median: 64.5
Mode: 65
Standard Deviation: 21.919296420176344
```

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

3. You are analyzing a dataset that captures the daily performance and activity of a humanoid robot in a simulated environment. The dataset link [robot_dataset\(robot_dataset\)_1.csv](#) includes the following attributes

Interaction_Count: Number of conversations the robot had daily.
Steps_Walked: Total steps taken each day.
Objects_Recognized: Number of objects successfully identified by the robot.
Learning_Sessions: Number of learning tasks completed.
Energy_Consumption (kWh): Daily energy usage of robots.

Perform Basic Statistical Operations:

- 1) What is the **average (mean)** number of conversations the robot has daily?

- 2) Find the **total steps walked** by the robot over a given period.
- 3) Determine the **maximum and minimum energy consumption** in the dataset.
- 4) Calculate the **correlation** between the number of steps walked and energy consumption.
- 5) Analyze the **distribution** of objects recognized daily (e.g., histogram or box plot).
- 6) What is the **variance** in the number of learning sessions completed?

```
import pandas as pd
a=pd.read_csv("robot_dataset.csv")
a["Interaction_Count"].mean()
```

5.51

```
import pandas as pd
a=pd.read_csv("robot_dataset.csv")
a["Steps_Walked"].sum()
```

14379

```
import pandas as pd
a=pd.read_csv("robot_dataset.csv")
a["Energy_Consumption (kWh)"].max()
```

3.0

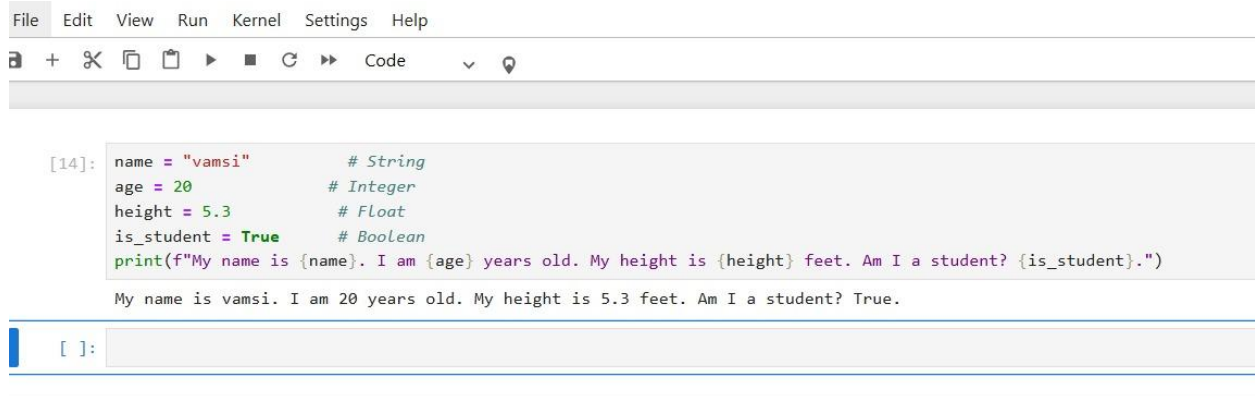
```
import pandas as pd
a=pd.read_csv("robot_dataset.csv")
a["Energy_Consumption (kWh)"].min()
```

1.0

```
import pandas as pd
a=pd.read_csv("robot_dataset.csv")
a["Learning_Sessions"].var()
```

391.9422845691385

4. Write a Python program that declares variables of different data types (e.g., string, integer, float, and boolean). Output the variables in a sentence format using print() and f-strings.



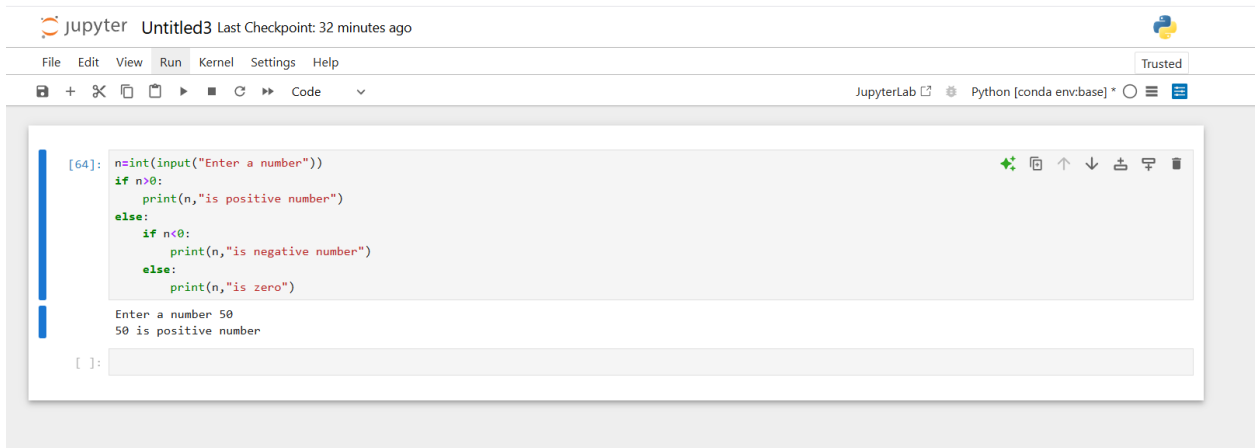
```
File Edit View Run Kernel Settings Help
+ ✂ 📄 📌 ▶ ■ ↺ ▶▶ Code ⌵ 🔍

[14]: name = "vamsi"      # String
      age = 20           # Integer
      height = 5.3       # Float
      is_student = True  # Boolean
      print(f"My name is {name}. I am {age} years old. My height is {height} feet. Am I a student? {is_student}.")

My name is vamsi. I am 20 years old. My height is 5.3 feet. Am I a student? True.

[ ]:
```

5. Write a Python program that takes an integer input and checks whether the number is positive, negative, or zero using conditional statements (if-else).



```
jupyter Untitled3 Last Checkpoint: 32 minutes ago
File Edit View Run Kernel Settings Help Trusted
+ ✂ 📄 📌 ▶ ■ ↺ ▶▶ Code ⌵ Python [conda env:base] * 🔍











[64]: n=int(input("Enter a number"))
      if n>0:
          print(n,"is positive number")
      else:
          if n<0:
              print(n,"is negative number")
          else:
              print(n,"is zero")

Enter a number 50
50 is positive number

[ ]:
```



6. Write a Python program that takes a number as input and prints the multiplication table for that number (from 1 to 10).

File Edit View Run Kernel Settings Help

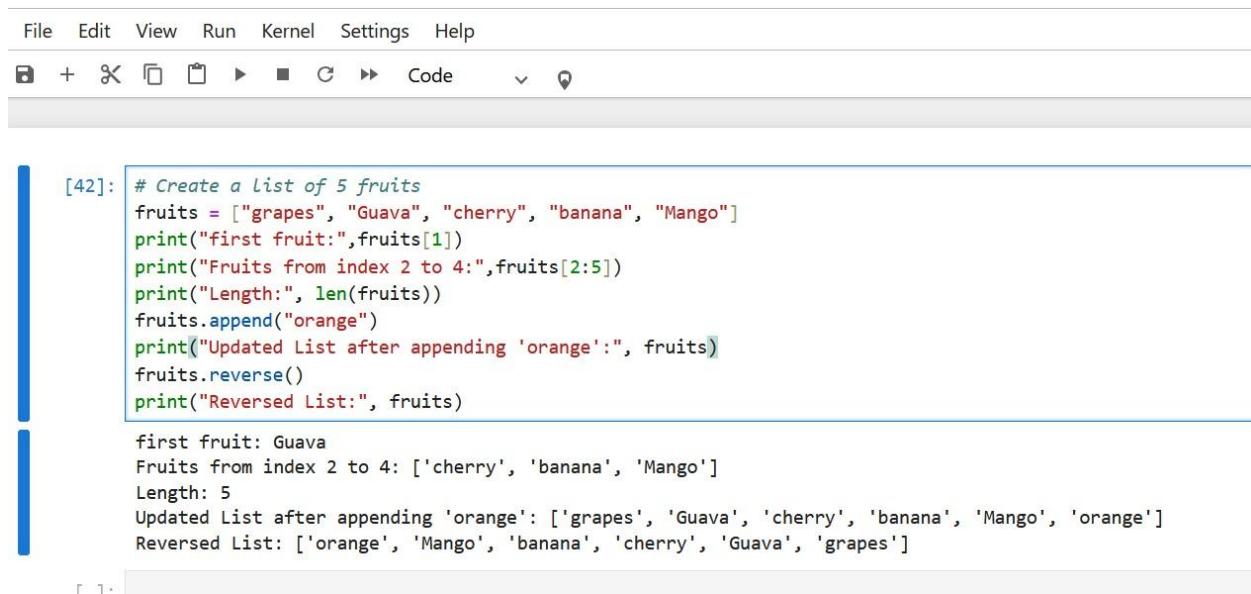
        Code  

```
[12]: num = 15
      for i in range(1, 11):
          print(num, 'x', i, '=', num*i)

15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
```

  1 •

7. Create a Python list that contains the names of 5 different fruits. Perform the given operations on the list.



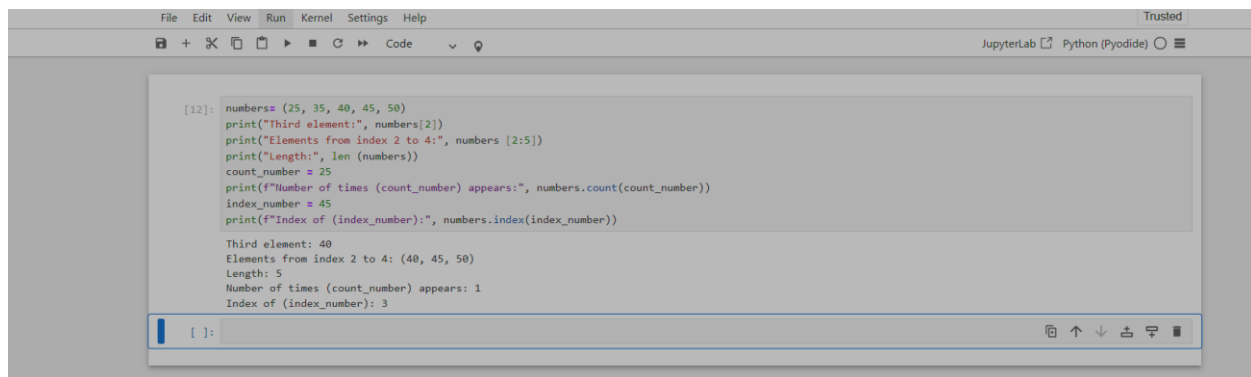
The image shows a JupyterLab interface with a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations, execution, and search. The code cell [42] contains the following Python code:

```
# Create a List of 5 fruits
fruits = ["grapes", "Guava", "cherry", "banana", "Mango"]
print("first fruit:", fruits[1])
print("Fruits from index 2 to 4:", fruits[2:5])
print("Length:", len(fruits))
fruits.append("orange")
print("Updated List after appending 'orange':", fruits)
fruits.reverse()
print("Reversed List:", fruits)
```

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

```
first fruit: Guava
Fruits from index 2 to 4: ['cherry', 'banana', 'Mango']
Length: 5
Updated List after appending 'orange': ['grapes', 'Guava', 'cherry', 'banana', 'Mango', 'orange']
Reversed List: ['orange', 'Mango', 'banana', 'cherry', 'Guava', 'grapes']
```

8. Write a Python program that creates a tuple containing 5 numbers. Perform the given operations on the tuple.



The image shows a JupyterLab interface with a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations, execution, and search. The code cell [12] contains the following Python code:

```
numbers = (25, 35, 40, 45, 50)
print("Third element:", numbers[2])
print("Elements from index 2 to 4:", numbers[2:5])
print("Length:", len(numbers))
count_number = 25
print("Number of times (count_number) appears:", numbers.count(count_number))
index_number = 45
print("Index of (index_number):", numbers.index(index_number))
```

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

```
Third element: 40
Elements from index 2 to 4: (40, 45, 50)
Length: 5
Number of times (count_number) appears: 1
Index of (index_number): 3
```

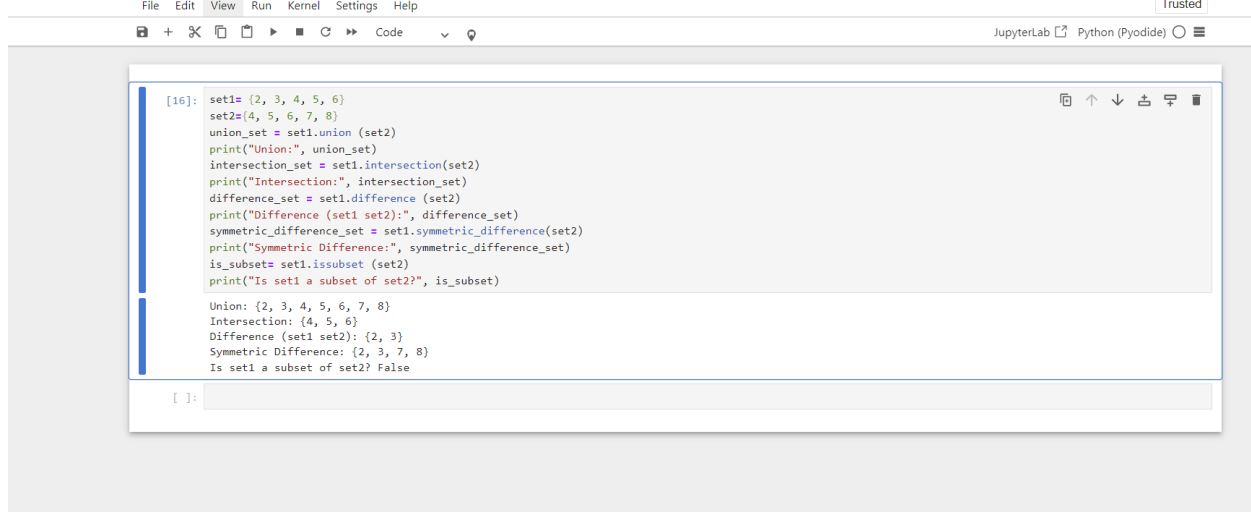
9. Create a dictionary that stores the names of 3 students as keys and their marks in mathematics as values. Perform the given operations.

```
File Edit View Run Kernel Settings Help
+ ✂ 📄 📄 ▶ ■ ↺ ▶ Code ▾ 🔍

•[12]: # Creating the dictionary
students_marks = {
    "vamshi": 85,
    "chethan": 78,
    "Cherry": 92
}
# 1. Add a new student and their marks
students_marks["balaji"] = 88
# 2. Update the marks of an existing student
students_marks["chetan"] = 82
# 3. Delete a student from the dictionary
del students_marks["vamshi"]
# 4. Retrieve and print the marks of a specific student
cherry_marks = students_marks.get("Cherry")
print(f"Cherry Marks: {cherry_marks}")
# 5. Print the final dictionary
print("Final Students Marks Dictionary:", students_marks)

Cherry Marks: 92
Final Students Marks Dictionary: {'chethan': 78, 'Cherry': 92, 'balaji': 88, 'chetan': 82}
```


10. Create two sets of integers. Perform the given set operations.

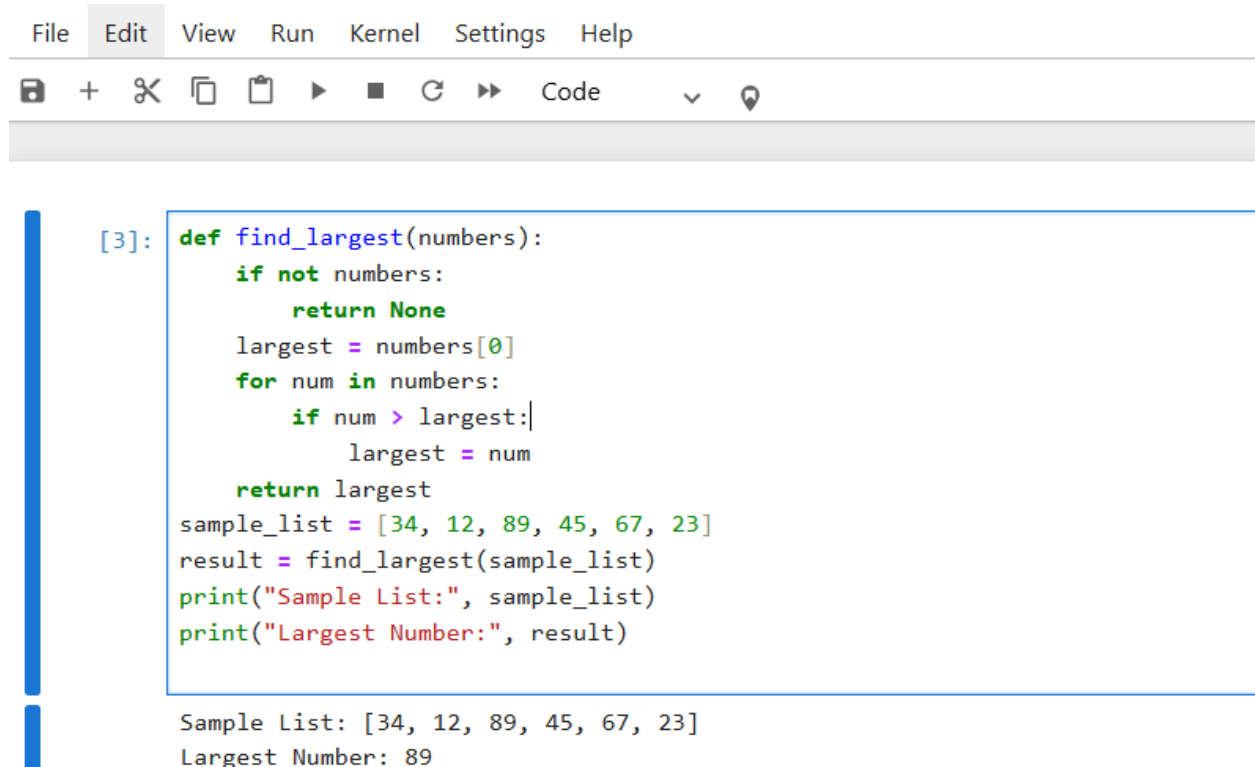


The image shows a JupyterLab interface with a code editor and a console. The code defines two sets, set1 and set2, and performs various set operations including union, intersection, difference, symmetric difference, and subset checking. The output displays the results of these operations.

```
[16]: set1={2, 3, 4, 5, 6}
      set2={4, 5, 6, 7, 8}
      union_set = set1.union(set2)
      print("Union:", union_set)
      intersection_set = set1.intersection(set2)
      print("Intersection:", intersection_set)
      difference_set = set1.difference(set2)
      print("Difference (set1 set2):", difference_set)
      symmetric_difference_set = set1.symmetric_difference(set2)
      print("Symmetric Difference:", symmetric_difference_set)
      is_subset = set1.issubset(set2)
      print("Is set1 a subset of set2?", is_subset)

      Union: {2, 3, 4, 5, 6, 7, 8}
      Intersection: {4, 5, 6}
      Difference (set1 set2): {2, 3}
      Symmetric Difference: {2, 3, 7, 8}
      Is set1 a subset of set2? False
```

11. Write a Python function called find_largest() that takes a list of numbers as input and returns the largest number from the list. Test the function with a sample list.

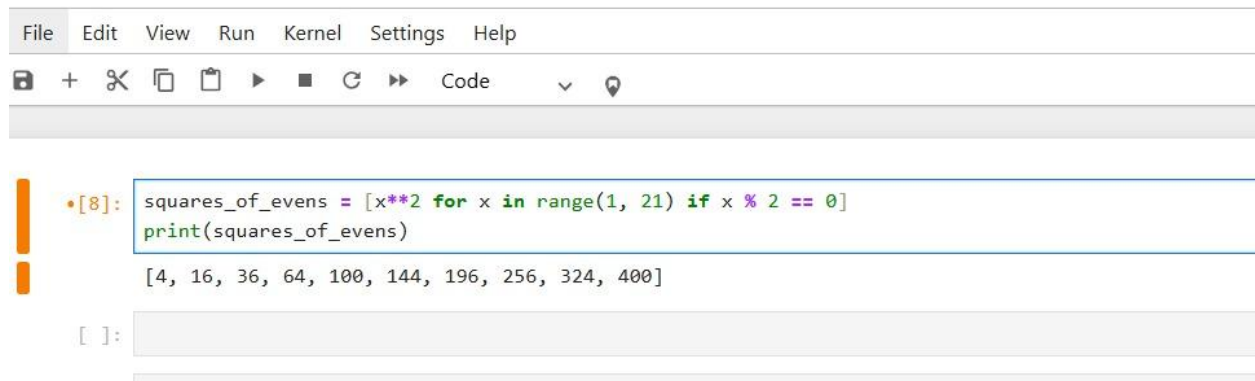


The image shows a JupyterLab interface with a code editor and a console. The code defines a function find_largest() that takes a list of numbers as input and returns the largest number. It also tests the function with a sample list and prints the results.

```
[3]: def find_largest(numbers):
      if not numbers:
          return None
      largest = numbers[0]
      for num in numbers:
          if num > largest:
              largest = num
      return largest
      sample_list = [34, 12, 89, 45, 67, 23]
      result = find_largest(sample_list)
      print("Sample List:", sample_list)
      print("Largest Number:", result)

      Sample List: [34, 12, 89, 45, 67, 23]
      Largest Number: 89
```

12. Use list comprehension to create a list of squares of all even numbers between 1 and 20.



The image shows a Jupyter Notebook interface. The top menu bar includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, copying, pasting, running, and other functions. The main area displays a code cell with the following Python code:

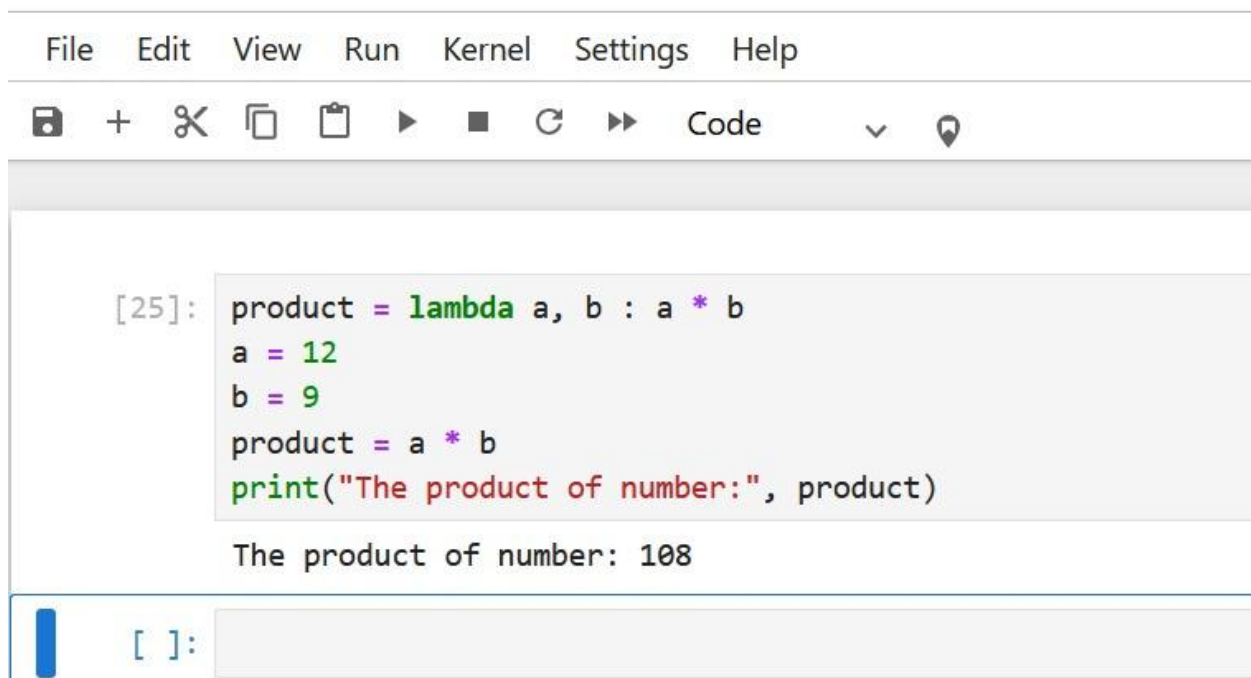
```
[8]: squares_of_evens = [x**2 for x in range(1, 21) if x % 2 == 0]
     print(squares_of_evens)
```

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

```
[4, 16, 36, 64, 100, 144, 196, 256, 324, 400]
```

Below the output, there is an empty code cell with the prompt `[]:`.

13. Write a Python script that uses a lambda function to calculate the product of two numbers provided by the user.



The image shows a Jupyter Notebook interface. The top menu bar includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, copying, pasting, running, and other functions. The main area displays a code cell with the following Python code:

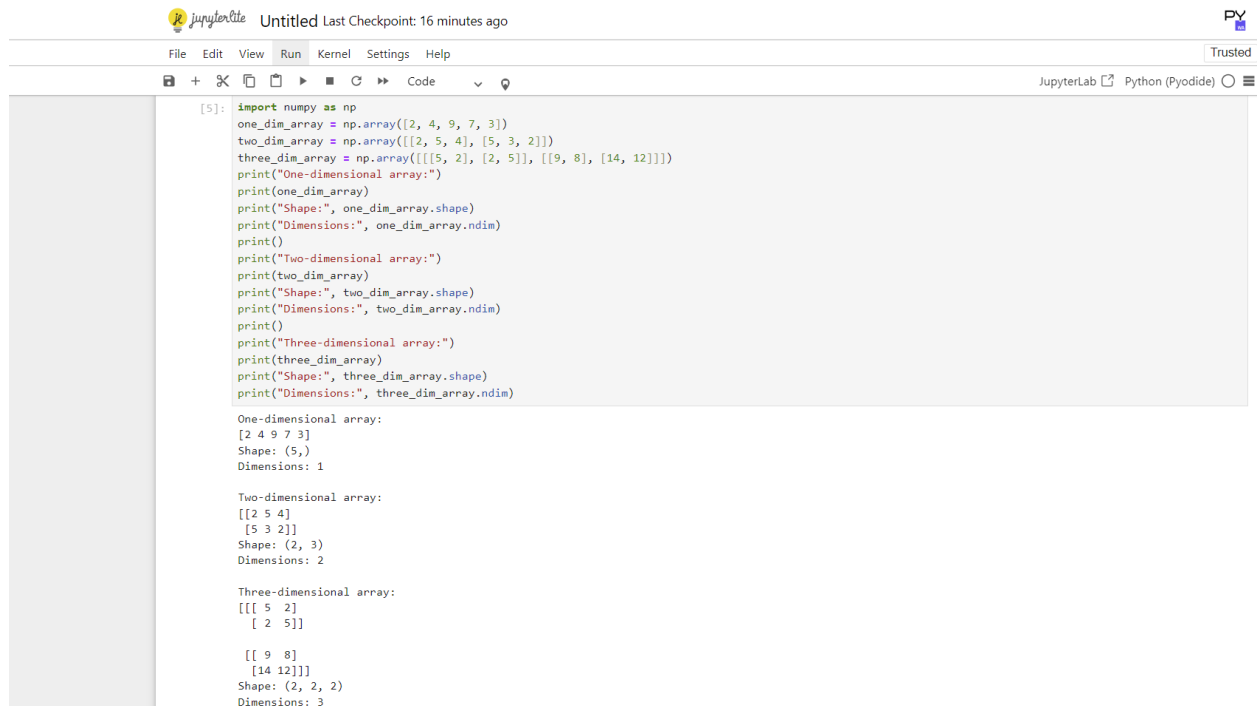
```
[25]: product = lambda a, b : a * b
      a = 12
      b = 9
      product = a * b
      print("The product of number:", product)
```

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

```
The product of number: 108
```

Below the output, there is an empty code cell with the prompt `[]:`.

14. Write a Python program to create a one-dimensional, two-dimensional, and three-dimensional NumPy array. Print the shape and dimensions of each array.



```
[5]: import numpy as np
one_dim_array = np.array([2, 4, 9, 7, 3])
two_dim_array = np.array([[2, 5, 4], [5, 3, 2]])
three_dim_array = np.array([[[5, 2], [2, 5]], [[9, 8], [14, 12]]])
print("One-dimensional array:")
print(one_dim_array)
print("Shape:", one_dim_array.shape)
print("Dimensions:", one_dim_array.ndim)
print()
print("Two-dimensional array:")
print(two_dim_array)
print("Shape:", two_dim_array.shape)
print("Dimensions:", two_dim_array.ndim)
print()
print("Three-dimensional array:")
print(three_dim_array)
print("Shape:", three_dim_array.shape)
print("Dimensions:", three_dim_array.ndim)
```

One-dimensional array:
[2 4 9 7 3]
Shape: (5,)
Dimensions: 1

Two-dimensional array:
[[2 5 4]
 [5 3 2]]
Shape: (2, 3)
Dimensions: 2

Three-dimensional array:
[[[5 2]
 [2 5]]

 [[9 8]
 [14 12]]]
Shape: (2, 2, 2)
Dimensions: 3

15. Write a Python program to create a 5x5 NumPy array of random integers and Perform array indexing as given.



```
[13]: import numpy as np
random_array = np.random.randint(0, 100, size=(5, 5))
print("Original 5x5 array:")
print(random_array)
print()
element = random_array[2, 3]
print("Element at row 2, column 3:", element)
third_row = random_array[2,:]
print("3rd row:", third_row)
fourth_column = random_array[:, 3]
print("4th column:", fourth_column)
subarray = random_array[1:4, 2:5]
print("Subarray (rows 1-3, columns 2-4):")
print(subarray)
```

Original 5x5 array:
[[43 12 78 8 3]
 [74 18 15 59 26]
 [10 58 34 19 7]
 [41 64 62 93 63]
 [95 34 88 56 71]]

Element at row 2, column 3: 19
3rd row: [10 58 34 19 7]
4th column: [8 59 19 93 56]
Subarray (rows 1-3, columns 2-4):
[[15 59 26]
 [34 19 7]
 [62 93 63]]

16. create a NumPy array of shape (4, 4) containing numbers from 1 to 16. Use slicing to extract for the given conditions

```
File Edit View Run Kernel Settings Help

[5]: import numpy as np # Importing NumPy
arr = np.arange(1, 17).reshape(4, 4)
print("Original 4x4 Array:\n", arr)
first_two_rows = arr[:2, :]
print("\nFirst two rows:\n", first_two_rows)
last_two_columns = arr[:, -2:]
print("\nLast two columns:\n", last_two_columns)
center_submatrix = arr[1:3, 1:3]
print("\n2x2 Center Sub-Matrix:\n", center_submatrix)
selected_rows = arr[[1, 3], :]
```

17. Write a Python program that creates a 2D array of shape (6, 2) using np.arange() and then reshapes it into a 3D array of shape (2, 3, 2). Flatten the reshaped array and print the result.

```
File Edit View Run Kernel Settings Help Trusted

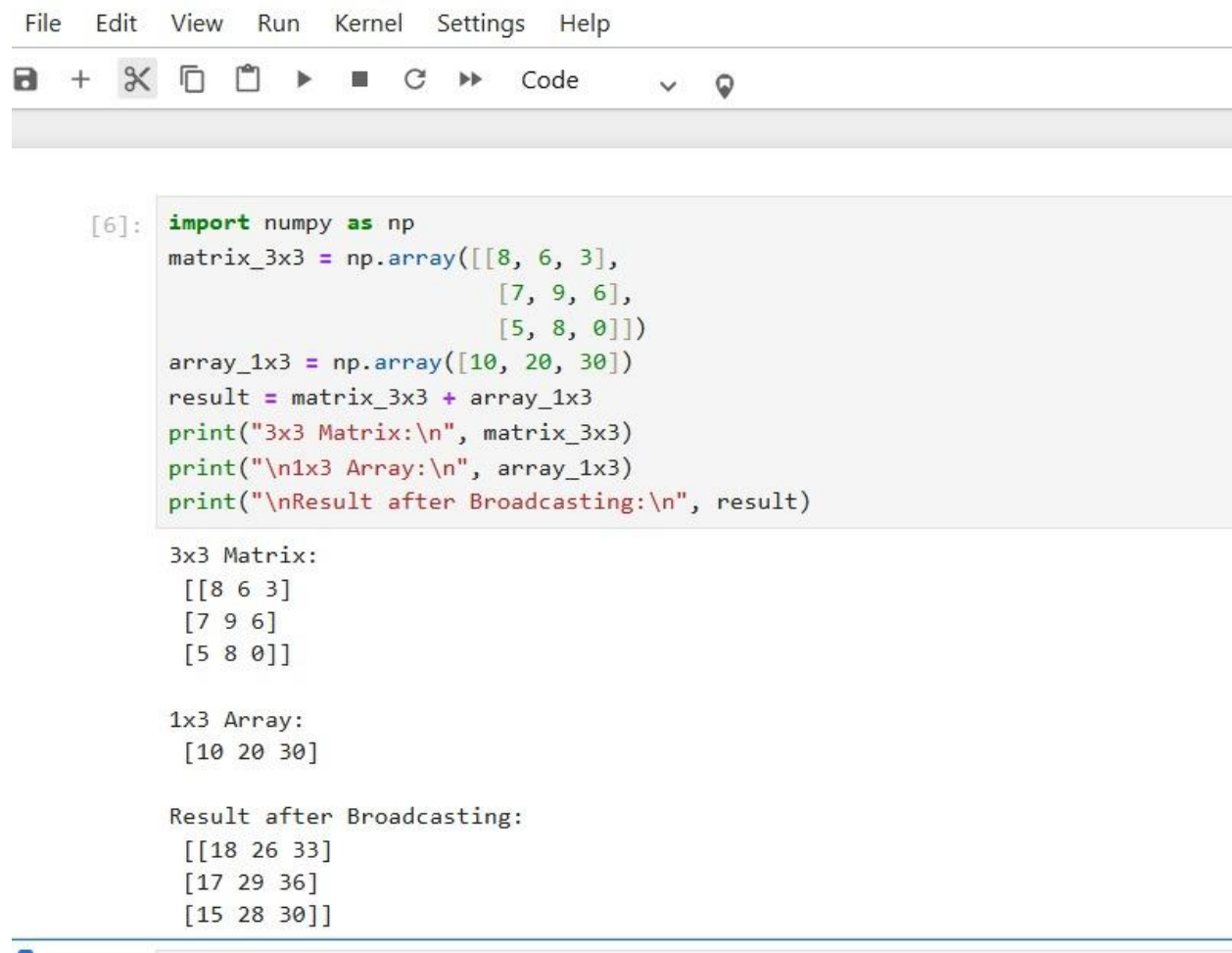
[12]: import numpy as np
array_2d = np.arange(12).reshape(6, 2)
print("Original 2D array (shape: (6, 2)):")
print(array_2d)
print()
array_3d = array_2d.reshape(2, 3, 2)
print("Reshaped 3D array (shape: (2, 3, 2)):")
print(array_3d)
print()
flattened_array = array_3d.flatten()
print("Flattened array:")
print(flattened_array)

Original 2D array (shape: (6, 2)):
[[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]

Reshaped 3D array (shape: (2, 3, 2)):
[[[ 0  1]
   [ 2  3]
   [ 4  5]]
 [[ 6  7]
   [ 8  9]
   [10 11]]]

Flattened array:
[ 0  1  2  3  4  5  6  7  8  9 10 11]
```

18. Write a Python program to demonstrate broadcasting. Create an array of shape (3, 3) and add a one-dimensional array of shape (1, 3) to it using broadcasting.



```
File Edit View Run Kernel Settings Help
[6]: import numpy as np
      matrix_3x3 = np.array([[8, 6, 3],
                             [7, 9, 6],
                             [5, 8, 0]])
      array_1x3 = np.array([10, 20, 30])
      result = matrix_3x3 + array_1x3
      print("3x3 Matrix:\n", matrix_3x3)
      print("\n1x3 Array:\n", array_1x3)
      print("\nResult after Broadcasting:\n", result)

3x3 Matrix:
[[8 6 3]
 [7 9 6]
 [5 8 0]]

1x3 Array:
[10 20 30]

Result after Broadcasting:
[[18 26 33]
 [17 29 36]
 [15 28 30]]
```

19. Create two NumPy arrays of the same shape, A and B. Perform the following arithmetic operations:

Element-wise addition.

Element-wise subtraction.

Element-wise multiplication.

Element-wise division.

```
File Edit View Run Kernel Settings Help
+ ✂ 📄 ▶ ■ ↺ ▶▶ Code ⌵ 🔍

[30]: import numpy as np
A = np.array ([ [3, 4, 5] ,[6, 7, 8] ])
B= np.array([[40, 50, 60], [70, 80, 90]])
print("Array A:")
print(A)
print()
print("Array B:")
print(B)
print()
addition = A + B
print("Element-wise addition (A + B): ")
print(addition)
print()
subtraction = A - B
print("Element-wise subtraction (A - B): ")
print(subtraction)
print()
multiplication = A * B
print("Element-wise multiplication (A * B) :")
print(multiplication)
print()
division = A / B
print("Element-wise division (A / B): ")
print(division)
```

20. Create a Pandas DataFrame with the given Name and marks of 3 courses:

Add a new column named 'Total' that represents the sum of all the courses. Add 'Grade' based on the values of the 'Total'. Print the updated DataFrame with the new 'Total' and 'Grade' column.

File Edit View Run Kernel Settings Help

📁 + ✂ 📄 ▶ ■ ↺ ▶▶ Code ▼ 🔔

```
[3]: import pandas as pd # Importing Pandas
data = {
    "Name": ["vamshi", "chetan", "Cherry"],
    "Math": [85, 78, 92],
    "Science": [88, 74, 90],
    "English": [82, 80, 85]
}
df = pd.DataFrame(data)
df["Total"] = df["Math"] + df["Science"] + df["English"]
def assign_grade(total):
    if total >= 250:
        return "A"
    elif total >= 220:
        return "B"
    elif total >= 180:
        return "C"
    else:
        return "D"
df["Grade"] = df["Total"].apply(assign_grade)
print(df)
```

	Name	Math	Science	English	Total	Grade
0	vamshi	85	88	82	255	A
1	chetan	78	74	80	232	B
2	Cherry	92	90	85	267	A