Date: February 14, 2024

SMART INTERNZ - APSCHE
AI / ML Training

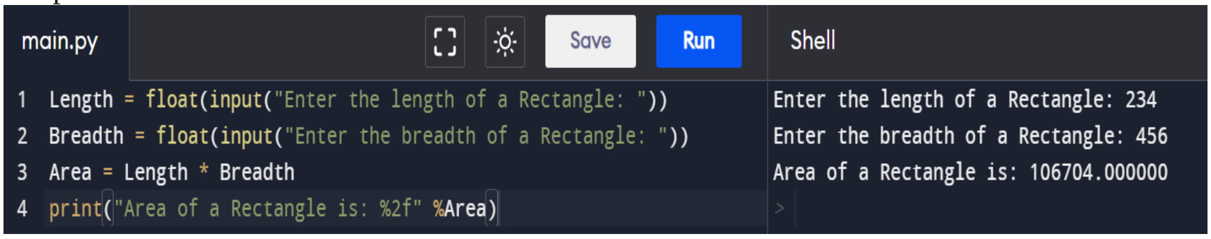
Assessment-1:

1. Write a Python program to calculate the area of a rectangle given its length and width.

Source code:

```
 Length = float(input("Enter the length of a Rectangle: "))
Breadth = float(input("Enter the breadth of a Rectangle: "))
Area = Length * Breadth
print("Area of a Rectangle is: %2f" %Area)
```

Output:



```
Length = float(input("Enter the length of a Rectangle: "))
Breadth = float(input("Enter the breadth of a Rectangle: "))
Area = Length * Breadth
print("Area of a Rectangle is: %2f" %Area)
```

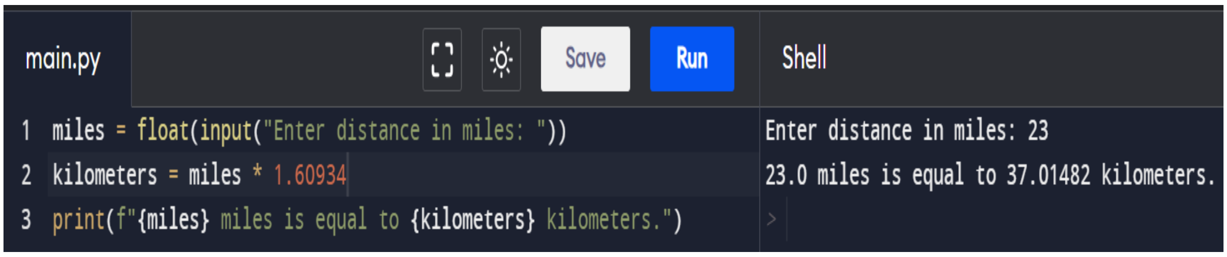
```
Enter the length of a Rectangle: 234
Enter the breadth of a Rectangle: 456
Area of a Rectangle is: 106704.000000
>
```

2. Write a program to convert miles to kilometers.

Source code:

```
miles = float(input("Enter distance in miles: "))
kilometers = miles * 1.60934
print(f"{miles} miles is equal to {kilometers} kilometers.")
```

Output:



```
miles = float(input("Enter distance in miles: "))
kilometers = miles * 1.60934
print(f"{miles} miles is equal to {kilometers} kilometers.")
```

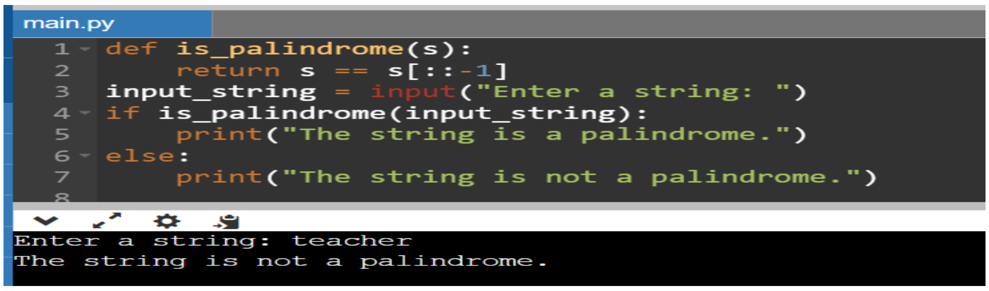
```
Enter distance in miles: 23
23.0 miles is equal to 37.01482 kilometers.
>
```

3. Write a function to check if a given string is a palindrome.

Source code:

```
def is_palindrome(s):
    return s == s[::-1]
input_string = input("Enter a string: ")
if is_palindrome(input_string):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

Output:



```
main.py
def is_palindrome(s):
    return s == s[::-1]
input_string = input("Enter a string: ")
if is_palindrome(input_string):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

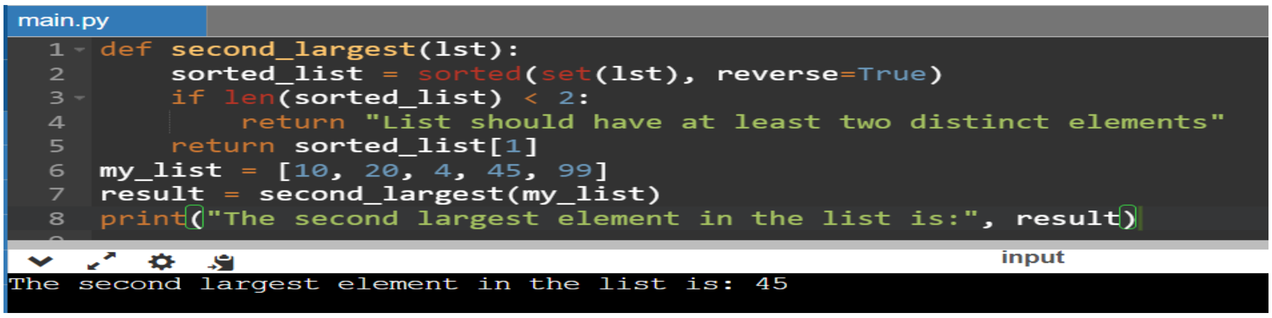
```
Enter a string: teacher
The string is not a palindrome.
```

4. Write a Python program to find the second largest element in a list.

Source code:

```
def second_largest(lst):
    sorted_list = sorted(set(lst), reverse=True)
    if len(sorted_list) < 2:
        return "List should have at least two distinct elements"
    return sorted_list[1]
my_list = [10, 20, 4, 45, 99]
result = second_largest(my_list)
print("The second largest element in the list is:", result)
```

Output:



```
main.py
def second_largest(lst):
    sorted_list = sorted(set(lst), reverse=True)
    if len(sorted_list) < 2:
        return "List should have at least two distinct elements"
    return sorted_list[1]
my_list = [10, 20, 4, 45, 99]
result = second_largest(my_list)
print("The second largest element in the list is:", result)
```

```
The second largest element in the list is: 45
```

5. Explain what indentation means in Python.

   Use of spaces or tabs at the beginning of lines to visually structure code blocks. It denotes the scope of statements within loops, conditionals, functions, etc., ensuring readability and defining the hierarchy of the code. Two types of indentations are

   1. Block Indentations: Used to denote the start and end of code blocks like loops, conditionals, functions, and classes. All statements within the same block have the same level of indentation.
   2. Continuation Indentations: Used for line continuation when a single logical line of code spans multiple physical lines. It maintains readability by aligning continuation lines with the beginning of the expression or statement

6. Write a program to perform set difference operation.

Source code:

```
def set_difference(set1, set2):
    return set1 - set2
```

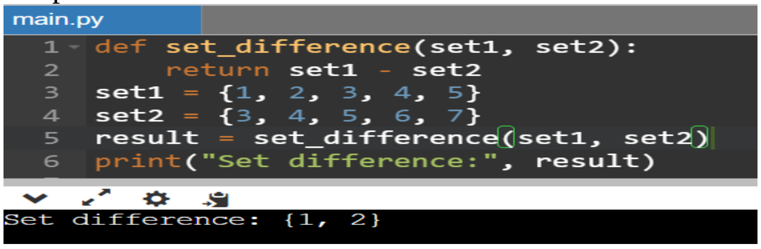
set1 = {1, 2, 3, 4, 5}

set2 = {3, 4, 5, 6, 7}

result = set_difference(set1, set2)

print("Set difference:", result)

Output:



```
main.py
def set_difference(set1, set2):
    return set1 - set2
set1 = {1, 2, 3, 4, 5}
set2 = {3, 4, 5, 6, 7}
result = set_difference(set1, set2)
print("Set difference:", result)
```

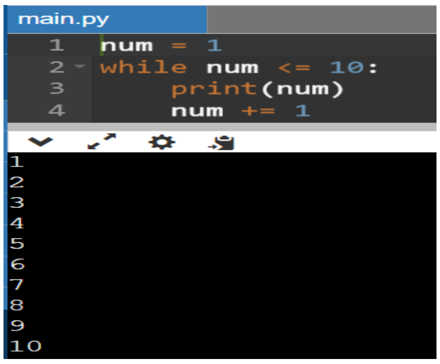
```
Set difference: {1, 2}
```

7. Write a Python program to print numbers from 1 to 10 using a while loop.

Source code:

```
num = 1
while num <= 10:
    print(num)
    num += 1
```

Output:



```
main.py
num = 1
while num <= 10:
    print(num)
    num += 1
```

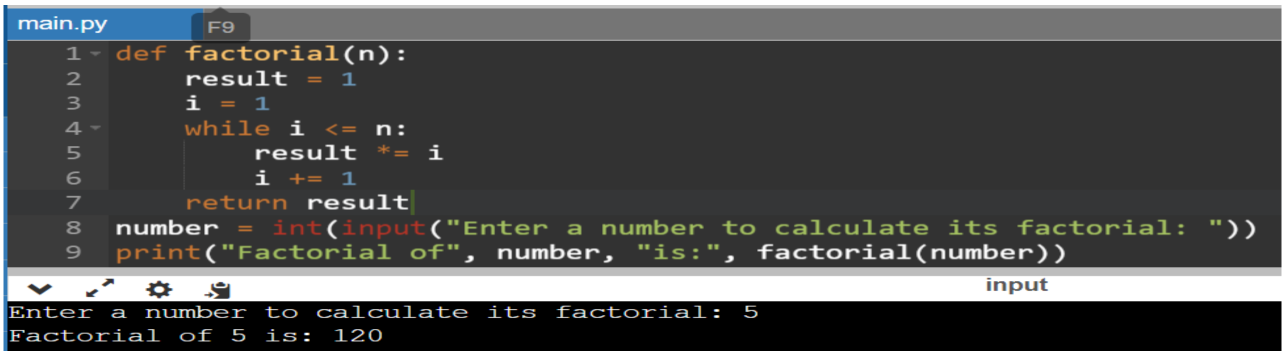
```
1
2
3
4
5
6
7
8
9
10
```

8. Write a program to calculate the factorial of a number using a while loop.

Source code:

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
number = int(input("Enter a number to calculate its factorial: "))
print("Factorial of", number, "is:", factorial(number))
```

Output:



```
main.py    F9
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
number = int(input("Enter a number to calculate its factorial: "))
print("Factorial of", number, "is:", factorial(number))
```

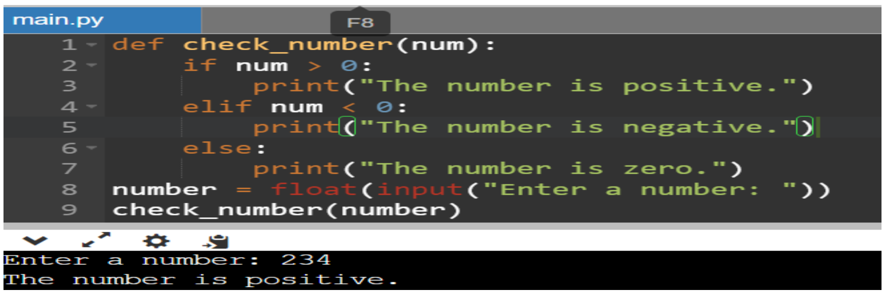
```
Enter a number to calculate its factorial: 5
Factorial of 5 is: 120
```

9. Write a Python program to check if a number is positive, negative, or zero using if-elif-else statements.

Source code:

```
def check_number(num):
 if num > 0:
        print("The number is positive.")
    elif num < 0:
        print("The number is negative.")
    else:
        print("The number is zero.")
number = float(input("Enter a number: "))
check_number(number)
```

Output:



```
main.py    F8
def check_number(num):
    if num > 0:
        print("The number is positive.")
    elif num < 0:
        print("The number is negative.")
    else:
        print("The number is zero.")
number = float(input("Enter a number: "))
check_number(number)
```

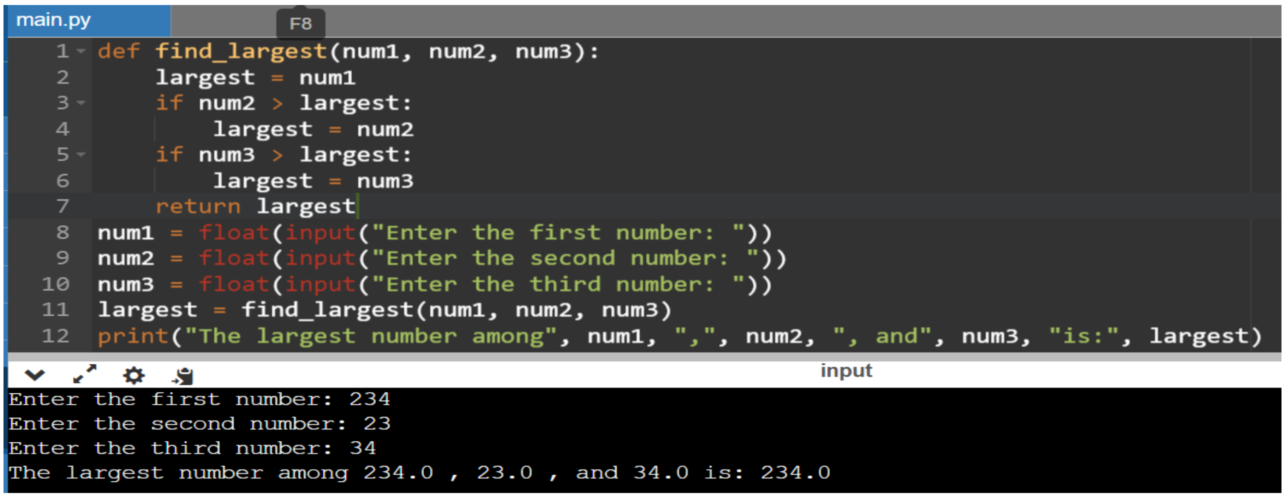
```
Enter a number: 234
The number is positive.
```

10. Write a program to determine the largest among three numbers using conditional statements.

Source code:

```
def find_largest(num1, num2, num3):
    largest = num1
    if num2 > largest:
        largest = num2
    if num3 > largest:
        largest = num3
    return largest
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
largest = find_largest(num1, num2, num3)
print("The largest number among", num1, ",", num2, ", and", num3, "is:", largest)
```

Output:



```python
def find_largest(num1, num2, num3):
    largest = num1
    if num2 > largest:
        largest = num2
    if num3 > largest:
        largest = num3
    return largest
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
largest = find_largest(num1, num2, num3)
print("The largest number among", num1, ",", num2, ", and", num3, "is:", largest)
```

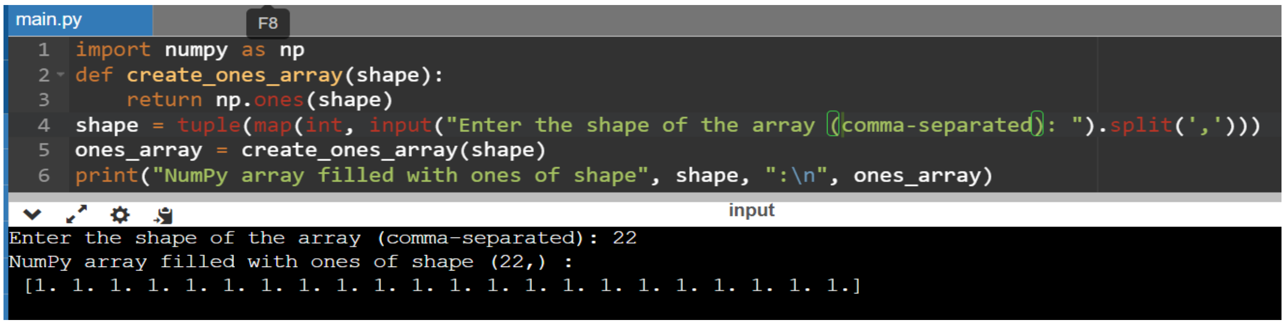
```
Enter the first number: 234
Enter the second number: 23
Enter the third number: 34
The largest number among 234.0 , 23.0 , and 34.0 is: 234.0
```

11.Write a Python program to create a numpy array filled with ones of given shape.

Source code:

```
import numpy as np
def create_ones_array(shape):
    return np.ones(shape)
shape = tuple(map(int, input("Enter the shape of the array (comma-separated): ").split(',')))
ones_array = create_ones_array(shape)
print("NumPy array filled with ones of shape", shape, ":\n", ones_array)
```

Output:



```python
import numpy as np
def create_ones_array(shape):
    return np.ones(shape)
shape = tuple(map(int, input("Enter the shape of the array (comma-separated): ").split(',')))
ones_array = create_ones_array(shape)
print("NumPy array filled with ones of shape", shape, ":\n", ones_array)
```

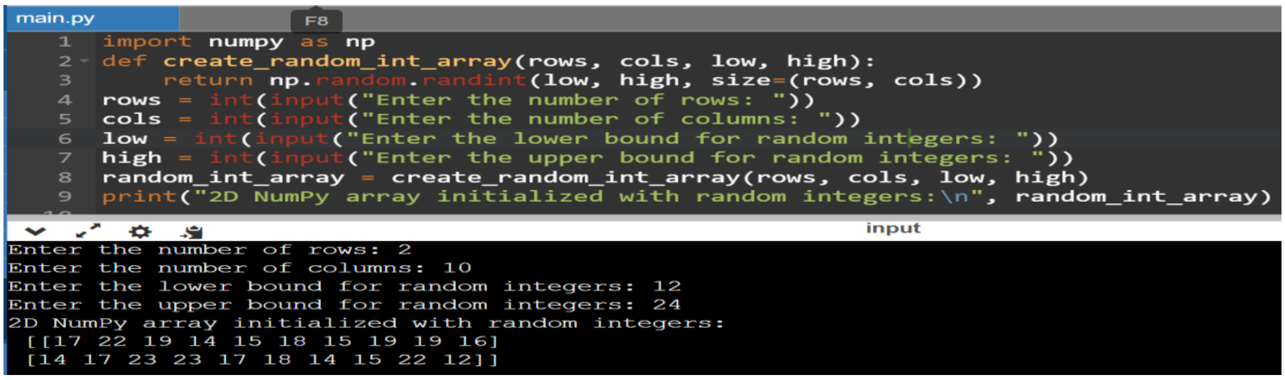
```
Enter the shape of the array (comma-separated): 22
NumPy array filled with ones of shape (22,) :
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

12.Write a program to create a 2D numpy array initialized with random integers.

Source code:

```
import numpy as np
def create_random_int_array(rows, cols, low, high):
    return np.random.randint(low, high, size=(rows, cols))
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))
low = int(input("Enter the lower bound for random integers: "))
high = int(input("Enter the upper bound for random integers: "))
random_int_array = create_random_int_array(rows, cols, low, high)
print("2D NumPy array initialized with random integers:\n", random_int_array)
```

Output:



```
import numpy as np
def create_random_int_array(rows, cols, low, high):
    return np.random.randint(low, high, size=(rows, cols))
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))
low = int(input("Enter the lower bound for random integers: "))
high = int(input("Enter the upper bound for random integers: "))
random_int_array = create_random_int_array(rows, cols, low, high)
print("2D NumPy array initialized with random integers:\n", random_int_array)
```

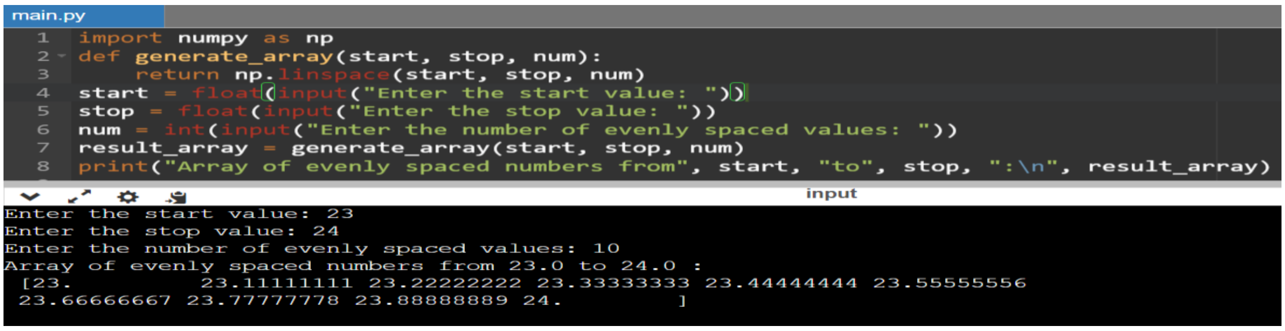
```
Enter the number of rows: 2
Enter the number of columns: 10
Enter the lower bound for random integers: 12
Enter the upper bound for random integers: 24
2D NumPy array initialized with random integers:
 [[17 22 19 14 15 18 15 19 19 16]
 [14 17 23 23 17 18 14 15 22 12]]
```

13.Write a Python program to generate an array of evenly spaced numbers over a specified range using linespace.

Source code:

```
import numpy as np

def generate_array(start, stop, num):

    return np.linspace(start, stop, num)

start = float(input("Enter the start value: "))

stop = float(input("Enter the stop value: "))

num = int(input("Enter the number of evenly spaced values: "))

result_array = generate_array(start, stop, num)

print("Array of evenly spaced numbers from", start, "to", stop, ":\n", result_array)
```

Output:



```
import numpy as np
def generate_array(start, stop, num):
    return np.linspace(start, stop, num)
start = float(input("Enter the start value: "))
stop = float(input("Enter the stop value: "))
num = int(input("Enter the number of evenly spaced values: "))
result_array = generate_array(start, stop, num)
print("Array of evenly spaced numbers from", start, "to", stop, ":\n", result_array)
```

```
Enter the start value: 23
Enter the stop value: 24
Enter the number of evenly spaced values: 10
Array of evenly spaced numbers from 23.0 to 24.0 :
 [23.         23.11111111 23.22222222 23.33333333 23.44444444 23.55555556
 23.66666667 23.77777778 23.88888889 24.        ]
```

14. Write a program to generate an array of 10 equally spaced values between 1 and 100 using linspace.

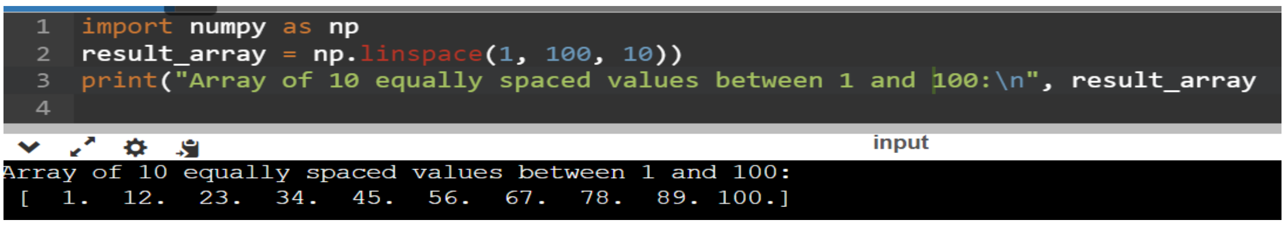
Source code:

import numpy as np

result_array = np.linspace(1, 100, 10))

print("Array of 10 equally spaced values between 1 and 100:\n", result_array

Output:



```
import numpy as np
result_array = np.linspace(1, 100, 10))
print("Array of 10 equally spaced values between 1 and 100:\n", result_array
```

```
Array of 10 equally spaced values between 1 and 100:
 [  1.  12.  23.  34.  45.  56.  67.  78.  89. 100.]
```

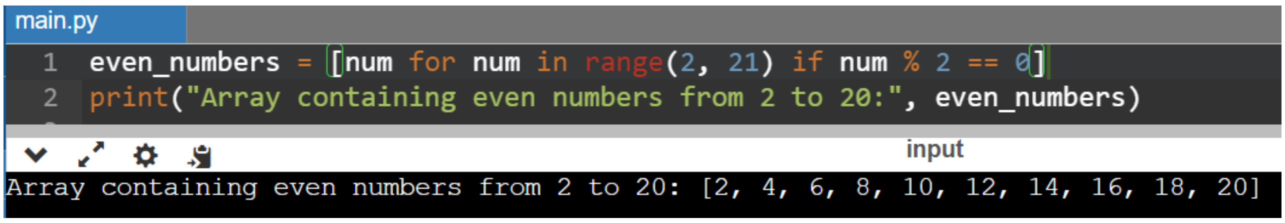
15.Write a Python program to create an array containing even numbers from 2 to 20 using a range.

Source code:

even_numbers = [num for num in range(2, 21) if num % 2 == 0]

print("Array containing even numbers from 2 to 20:", even_numbers)

Output:



main.py

```
even_numbers = [num for num in range(2, 21) if num % 2 == 0]
print("Array containing even numbers from 2 to 20:", even_numbers)
```

```
Array containing even numbers from 2 to 20: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

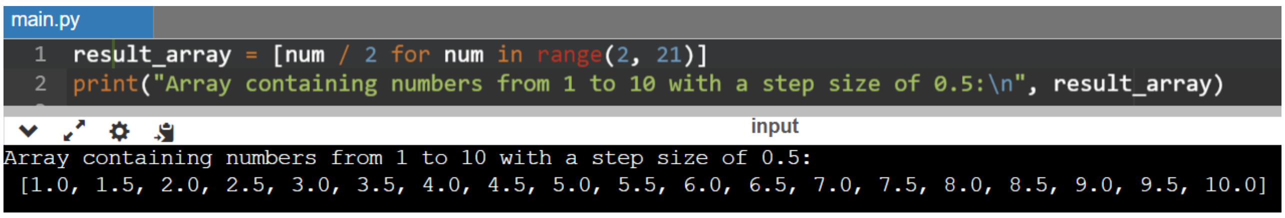
16.Write a program to create an array containing numbers from 1 to 10 with a step size of 0.5 using a range.

Source code:

result_array = [num / 2 for num in range(2, 21)]

print("Array containing numbers from 1 to 10 with a step size of 0.5:\n", result_array)

Output:



main.py

```
result_array = [num / 2 for num in range(2, 21)]
print("Array containing numbers from 1 to 10 with a step size of 0.5:\n", result_array)
```

```
Array containing numbers from 1 to 10 with a step size of 0.5:
 [1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5, 10.0]
```