

## ASSESSMENT-1 FOR AIML PROGRAM

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

Sol) `def calculate_rectangle_area(length, width):`

`area = length * width`

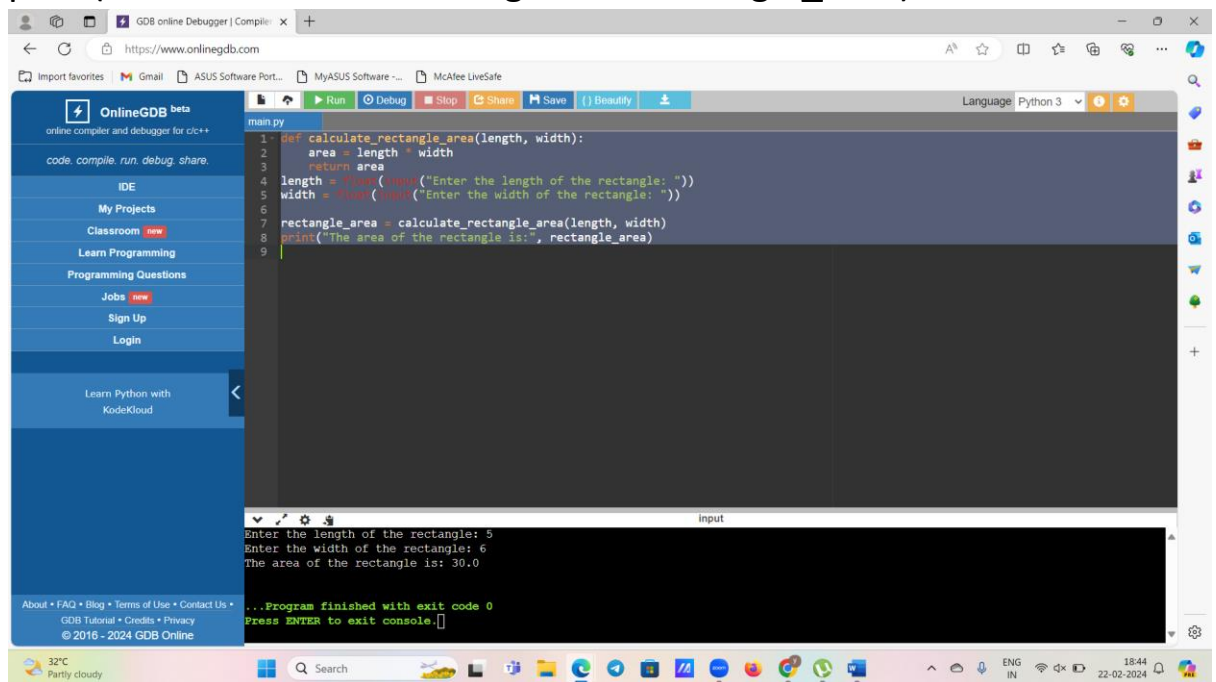
`return area`

`length = float(input("Enter the length of the rectangle: "))`

`width = float(input("Enter the width of the rectangle: "))`

`rectangle_area = calculate_rectangle_area(length, width)`

`print("The area of the rectangle is:", rectangle_area)`



The screenshot displays the OnlineGDB web application. The left sidebar contains navigation links such as 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Jobs', 'Sign Up', and 'Login'. The main editor area shows a Python script with the following code:

```
1 def calculate_rectangle_area(length, width):
2     area = length * width
3     return area
4 length = float(input("Enter the length of the rectangle: "))
5 width = float(input("Enter the width of the rectangle: "))
6
7 rectangle_area = calculate_rectangle_area(length, width)
8 print("The area of the rectangle is:", rectangle_area)
9
```

Below the editor, the 'Input' section shows the user's input: 'Enter the length of the rectangle: 5' and 'Enter the width of the rectangle: 6'. The output section displays the result: 'The area of the rectangle is: 30.0'. At the bottom, a status message indicates: '...Program finished with exit code 0 Press ENTER to exit console.'

2.) Write a program to convert miles to kilometers

Sol) `def miles_to_kilometers(miles):`

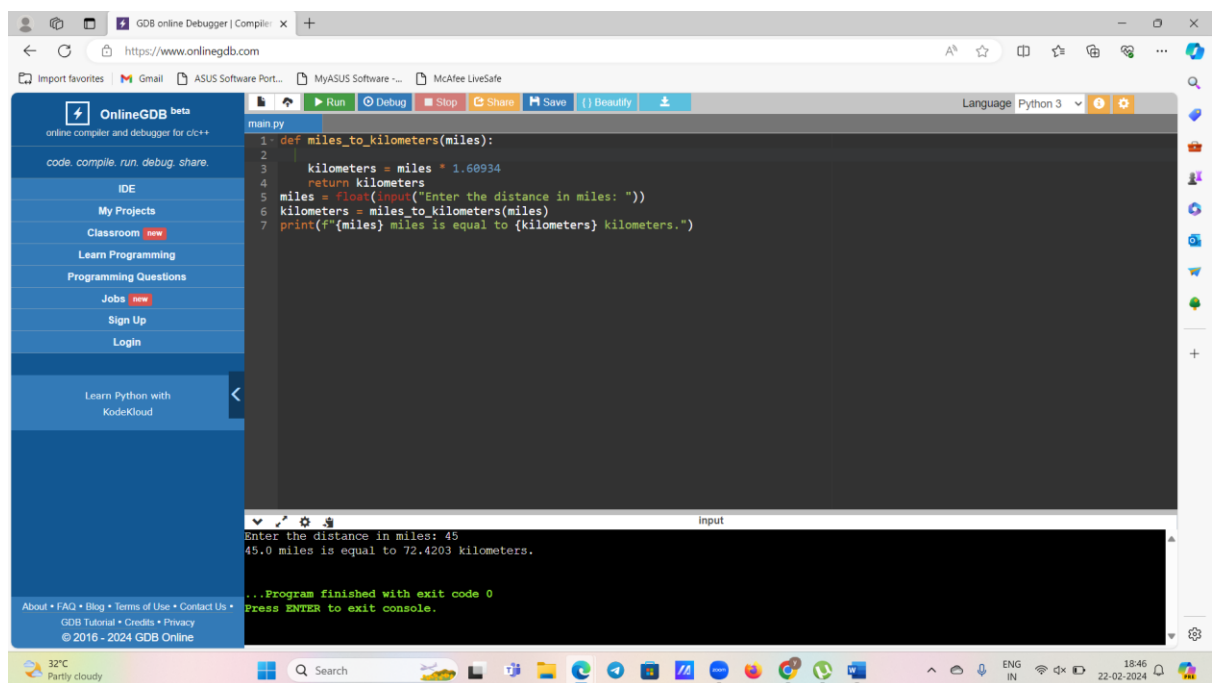
`kilometers = miles * 1.60934`

`return kilometers`

`miles = float(input("Enter the distance in miles: "))`

`kilometers = miles_to_kilometers(miles)`

`print(f"{miles} miles is equal to {kilometers} kilometers.")`



The screenshot shows the OnlineGDB web IDE interface. The left sidebar contains navigation links: OnlineGDB beta, code, compile, run, debug, share, IDE, My Projects, Classroom, Learn Programming, Programming Questions, Jobs, Sign Up, and Login. The main editor area displays a Python script named 'main.py' with the following code:

```
1 def miles_to_kilometers(miles):
2     kilometers = miles * 1.60934
3     return kilometers
4
5 miles = float(input("Enter the distance in miles: "))
6 kilometers = miles_to_kilometers(miles)
7 print(f"{miles} miles is equal to {kilometers} kilometers.")
```

The 'Run' button is highlighted in green. Below the editor, the 'Input' section shows the user input '45' and the program output '45.0 miles is equal to 72.4203 kilometers.'. The status bar at the bottom indicates 'Program finished with exit code 0' and 'Press ENTER to exit console.'.

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

Sol) `def is_palindrome(s):`

`s = s.replace(" ", "").lower()`

`return s == s[::-1]`

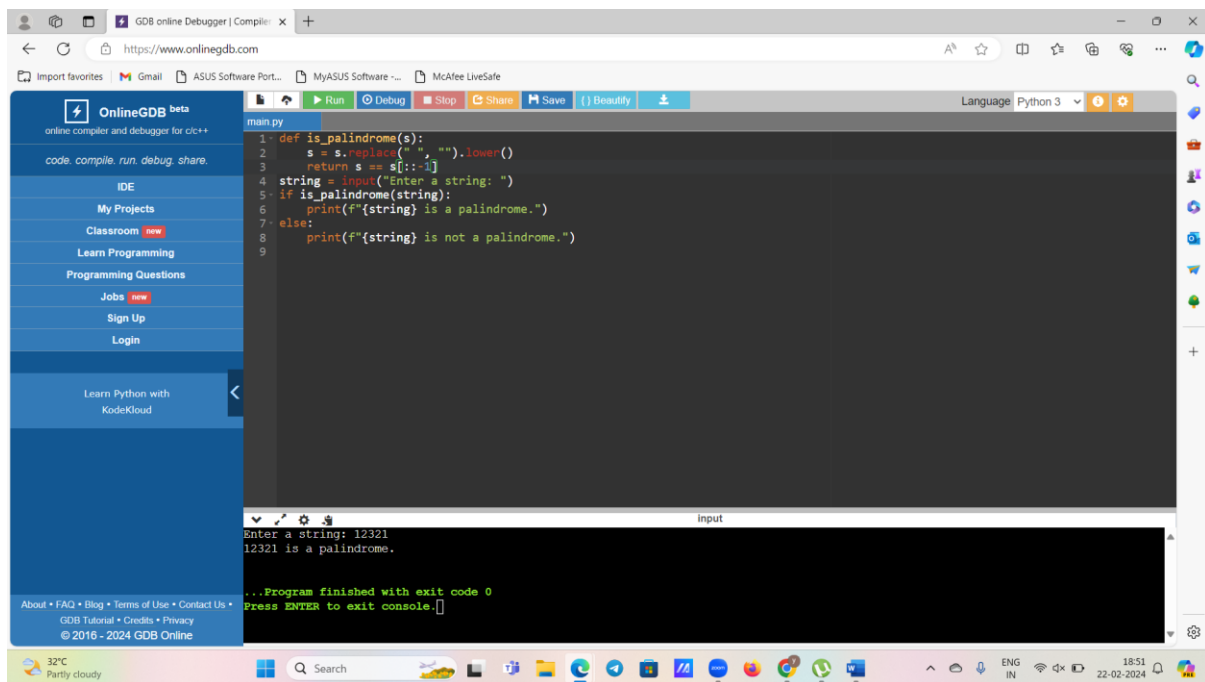
`string = input("Enter a string: ")`

`if is_palindrome(string):`

`print(f"{string} is a palindrome.")`

`else:`

`print(f"{string} is not a palindrome.")`



The screenshot shows the OnlineGDB web interface. The browser address bar displays `https://www.onlinegdb.com`. The left sidebar contains navigation links: IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Jobs (marked 'new'), Sign Up, and Login. The main editor area shows a Python script named `main.py` with the following code:

```
1- def is_palindrome(s):
2-     s = s.replace(" ", "").lower()
3-     return s == s[::-1]
4- string = input("Enter a string: ")
5- if is_palindrome(string):
6-     print(f"{string} is a palindrome.")
7- else:
8-     print(f"{string} is not a palindrome.")
9-
```

The output console at the bottom shows the execution results:

```
Input
Enter a string: 12321
12321 is a palindrome.

...Program finished with exit code 0
Press ENTER to exit console.
```

The Windows taskbar at the bottom indicates a temperature of 32°C, 'Partly cloudy' weather, and the date 22-02-2024.

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

```
def find_second_largest(nums):  
    if len(nums) < 2:  
        return None  
  
    largest = second_largest = float('-inf')  
  
    for num in nums:  
        if num > largest:  
            second_largest = largest  
            largest = num  
  
        elif num > second_largest and num != largest:  
            second_largest = num  
  
    return second_largest if second_largest != float('-inf') else None  
  
numbers = [int(x) for x in input("Enter the list of numbers separated  
by space: ").split()]  
  
second_largest = find_second_largest(numbers)  
  
if second_largest is not None:  
    print("The second largest element in the list is:", second_largest)  
else:  
    print("The list doesn't have a second largest element.")
```

```
1 def find_second_largest(nums):
2     if len(nums) < 2:
3         return None
4     largest = second_largest = float('-inf')
5     for num in nums:
6         if num > largest:
7             second_largest = largest
8             largest = num
9         elif num > second_largest and num != largest:
10            second_largest = num
11    return second_largest if second_largest != float('-inf') else None
12 numbers = [int(x) for x in input("Enter the list of numbers separated by space: ").split()]
13 second_largest = find_second_largest(numbers)
14 if second_largest is not None:
15     print("The second largest element in the list is:", second_largest)
16 else:
17     print("The list doesn't have a second largest element.")
18
```

Input: Enter the list of numbers separated by space: 1 2 3 4 5 6 7

Output: The second largest element in the list is: 6

...Program finished with exit code 0  
Press ENTER to exit console.

## 5.) Explain what indentation means in Python.

Indentation is a very important concept of Python because without properly indenting the Python code, you will end up seeing Indentation Error and the code will not get compiled.

Python indentation refers to adding white space before a statement to a particular block of code. In another word, all the statements with the same space to the right, belong to the same code block.

Python indentation is a way of telling a Python interpreter that the group of statements belongs to a particular block of code. A block is a combination of all these statements. Block can be regarded as the grouping of statements for a specific purpose. Most programming languages like C, C++, and Java use braces { } to define a block of code. Python uses indentation to highlight the blocks of code. Whitespace is used for indentation in Python. All statements with the same distance to the right belong to the same block of code. If a block has to be more deeply nested, it is simply indented further to the right.

For ex :- Statement (line 1), if condition (line 2), and statement (last line) belongs to the same block which means that after statement 1, if condition will be executed. and suppose the if condition becomes False then the Python will jump to the last statement for execution.

6.) Write a program to perform set difference operation.

Sol)def set\_difference(set1, set2):

    return set1 - set2

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

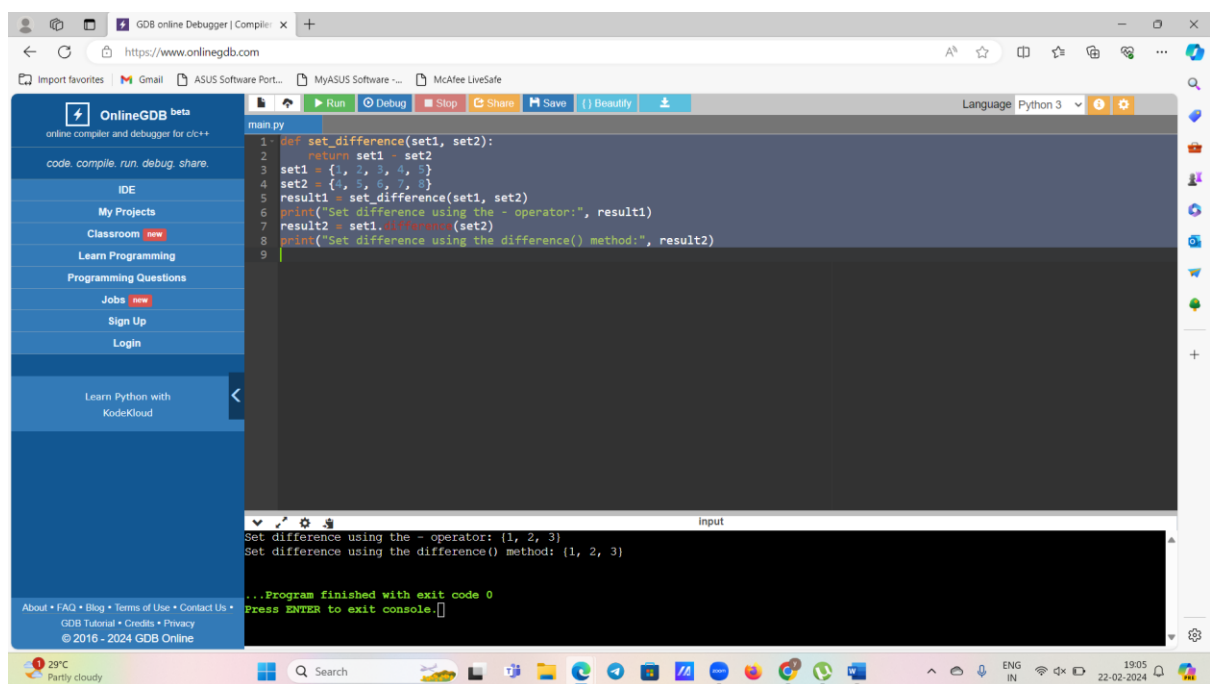
set2 = {4, 5, 6, 7, 8}

result1 = set\_difference(set1, set2)

print("Set difference using the - operator:", result1)

result2 = set1.difference(set2)

print("Set difference using the difference() method:", result2)



```
1 def set_difference(set1, set2):
2     return set1 - set2
3 set1 = {1, 2, 3, 4, 5}
4 set2 = {4, 5, 6, 7, 8}
5 result1 = set_difference(set1, set2)
6 print("Set difference using the - operator:", result1)
7 result2 = set1.difference(set2)
8 print("Set difference using the difference() method:", result2)
9
```

Set difference using the - operator: {1, 2, 3}

Set difference using the difference() method: {1, 2, 3}

...Program finished with exit code 0  
Press ENTER to exit console.

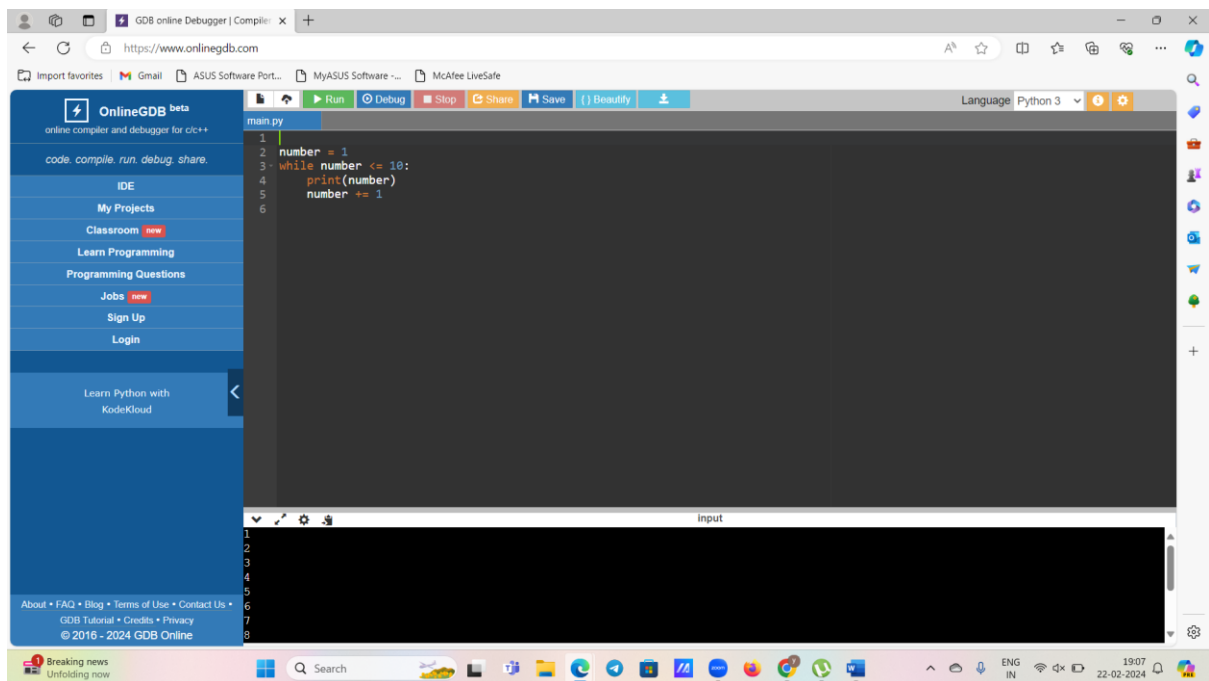
7.) Write a Python program to print numbers from 1 to 10 using a while loop.

Sol) number = 1

while number <= 10:

    print(number)

    number += 1



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

Sol) def factorial(n):

if n < 0:

return None

elif n == 0:

return 1

else:

result = 1

while n > 0:

result \*= n

n -= 1

return result

number = int(input("Enter a number to calculate its factorial: "))

fact = factorial(number)

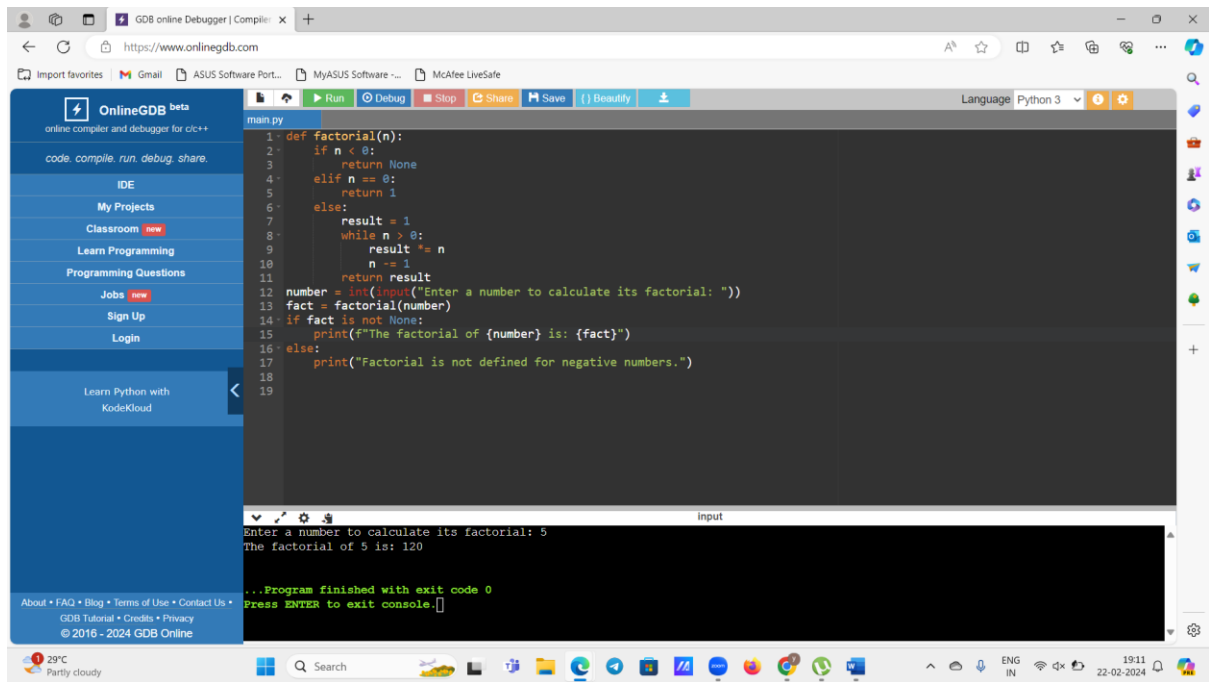
if fact is not None:

print(f"The factorial of {number} is: {fact}")

else:

print("Factorial is not defined for negative numbers.")





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

Sol) def check\_number(num):

if num > 0:

return "Positive"

elif num < 0:

return "Negative"

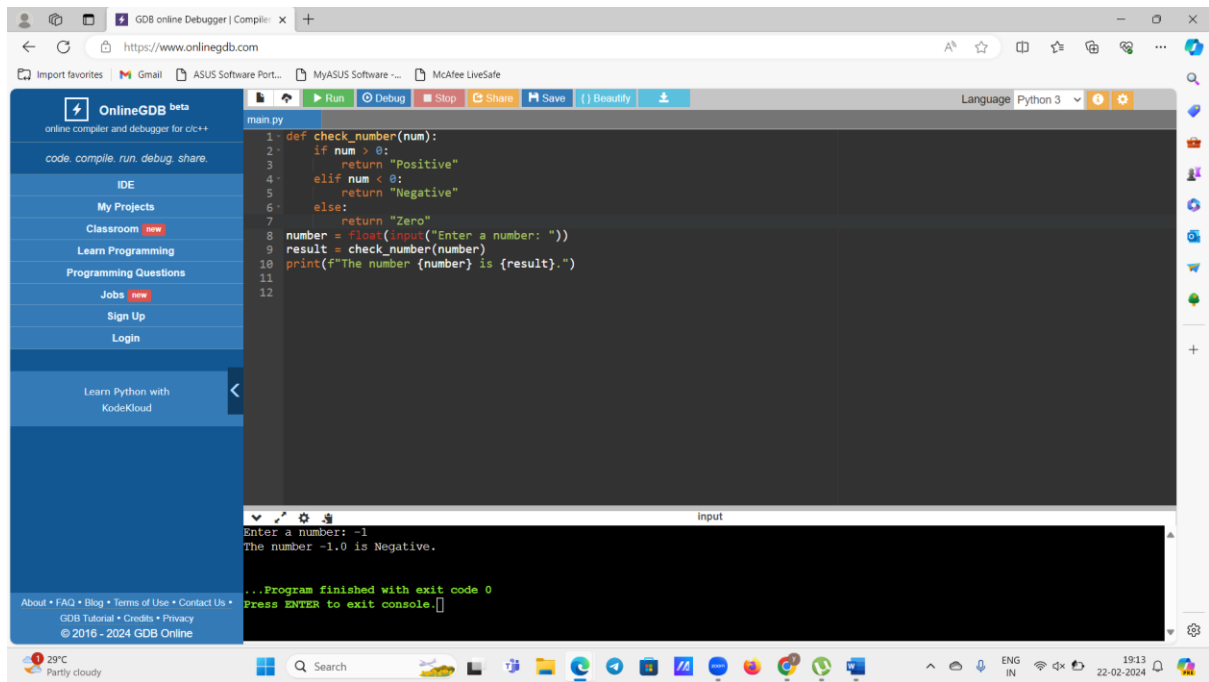
else:

return "Zero"

number = float(input("Enter a number: "))

result = check\_number(number)

print(f"The number {number} is {result}.")



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

Sol)def find\_largest(num1, num2, num3):

if num1 >= num2 and num1 >= num3:

return num1

elif num2 >= num1 and num2 >= num3:

return num2

else:

return num3

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(f"The largest number among {num1}, {num2}, and {num3} is: {largest}")

The screenshot shows the OnlineGDB IDE interface. The left sidebar contains navigation links: OnlineGDB beta, code.compile.run.debug.share, IDE, My Projects, Classroom, Learn Programming, Programming Questions, Jobs, Sign Up, and Login. The main editor area displays a Python script named 'main.py' that defines a function 'find\_largest' to find the maximum of three numbers. The script prompts the user for three numbers and prints the result. The console output shows the program execution with inputs 5, 2, and 1, resulting in the largest number being 5.0.

```
1 def find_largest(num1, num2, num3):
2     if num1 >= num2 and num1 >= num3:
3         return num1
4     elif num2 >= num1 and num2 >= num3:
5         return num2
6     else:
7         return num3
8 num1 = float(input("Enter the first number: "))
9 num2 = float(input("Enter the second number: "))
10 num3 = float(input("Enter the third number: "))
11 largest = find_largest(num1, num2, num3)
12 print(f"The largest number among {num1}, {num2}, and {num3} is: {largest}")
13
```

Enter the first number: 5  
Enter the second number: 2  
Enter the third number: 1  
The largest number among 5.0, 2.0, and 1.0 is: 5.0  
...Program finished with exit code 0  
Press ENTER to exit console.

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

```
import numpy as np
```

```
def ones_array(shape):
```

```
    return np.ones(shape)
```

```
shape = tuple(map(int, input("Enter the shape of the array (comma-separated): ").split(',')))
```

```
array_ones = ones_array(shape)
```

```
print("Array filled with ones of shape", shape, ":\n", array_ones)
```

The screenshot shows the OnlineGDB IDE interface. The left sidebar is the same as in the first image. The main editor area displays a Python script named 'main.py' that imports numpy, defines a function 'ones\_array' to create a numpy array of ones, and prompts the user for the shape of the array. The script prints the resulting array. The console output shows the program execution with input shape (3, 4), resulting in a 3x4 array of ones.

```
1 import numpy as np
2 def ones_array(shape):
3     return np.ones(shape)
4 shape = tuple(map(int, input("Enter the shape of the array (comma-separated): ").split(',')))
5 array_ones = ones_array(shape)
6 print("Array filled with ones of shape", shape, ":\n", array_ones)
7
```

Array filled with ones of shape (3, 4) :  
[[1. 1. 1. 1.]  
 [1. 1. 1. 1.]  
 [1. 1. 1. 1.]  
...Program finished with exit code 0  
Press ENTER to exit console.

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

```
import numpy as np
```

```
def random_int_array(rows, cols, low=0, high=10):
```

```
    return np.random.randint(low, high, size=(rows, cols))
```

```
rows = 3
```

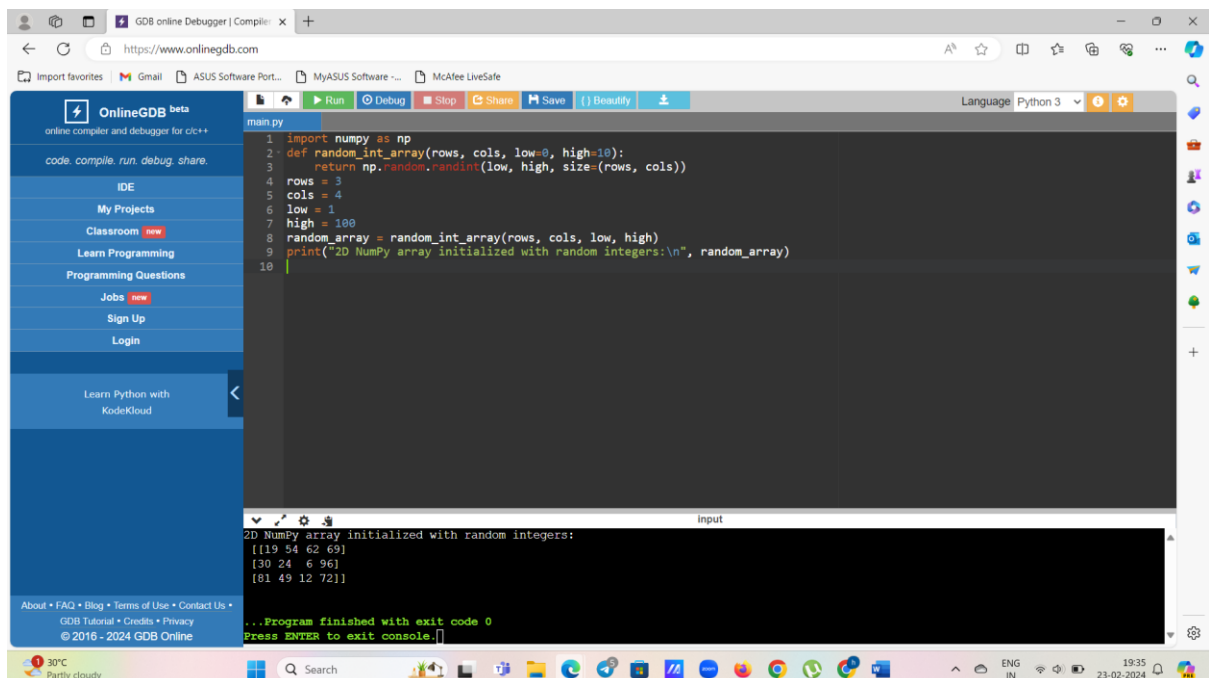
```
cols = 4
```

```
low = 1
```

```
high = 100
```

```
random_array = random_int_array(rows, cols, low, high)
```

```
print("2D NumPy array initialized with random integers:\n",  
      random_array)
```



The screenshot displays the OnlineGDB web application. The left sidebar contains navigation links such as 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Jobs', 'Sign Up', and 'Login'. The main editor area shows a Python script with the following code:

```
1 import numpy as np
2 def random_int_array(rows, cols, low=0, high=10):
3     return np.random.randint(low, high, size=(rows, cols))
4 rows = 3
5 cols = 4
6 low = 1
7 high = 100
8 random_array = random_int_array(rows, cols, low, high)
9 print("2D NumPy array initialized with random integers:\n", random_array)
10
```

The output console at the bottom shows the execution results:

```
2D NumPy array initialized with random integers:
[[19 54 62 69]
 [30 24 6 96]
 [81 49 12 72]]
...Program finished with exit code 0
Press ENTER to exit console.
```

The browser's taskbar at the bottom indicates a temperature of 30°C, a 'Partly cloudy' weather condition, and the date and time as 19:35 on 23-02-2024.

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

```
import numpy as np
```

```
def generate_linspace(start, stop, num=50):
```

```
    return np.linspace(start, stop, num)
```

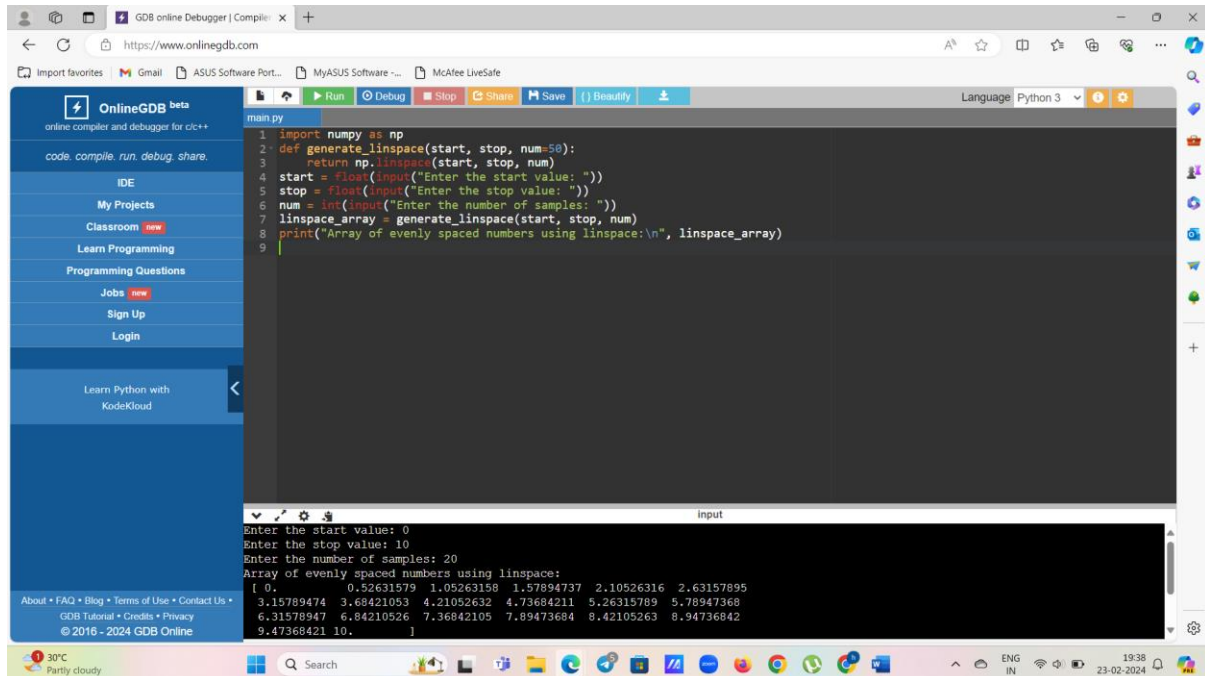
```
start = float(input("Enter the start value: "))
```

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

```
num = int(input("Enter the number of samples: "))
```

```
linspace_array = generate_linspace(start, stop, num)
```

```
print("Array of evenly spaced numbers using linspace:\n",  
linspace_array)
```



The screenshot shows the OnlineGDB web interface. The code is pasted into the editor, and the output is displayed in the console. The output shows the array of evenly spaced numbers generated by the program.

```
1 import numpy as np
2 def generate_linspace(start, stop, num=50):
3     return np.linspace(start, stop, num)
4 start = float(input("Enter the start value: "))
5 stop = float(input("Enter the stop value: "))
6 num = int(input("Enter the number of samples: "))
7 linspace_array = generate_linspace(start, stop, num)
8 print("Array of evenly spaced numbers using linspace:\n", linspace_array)
9
```

Input:

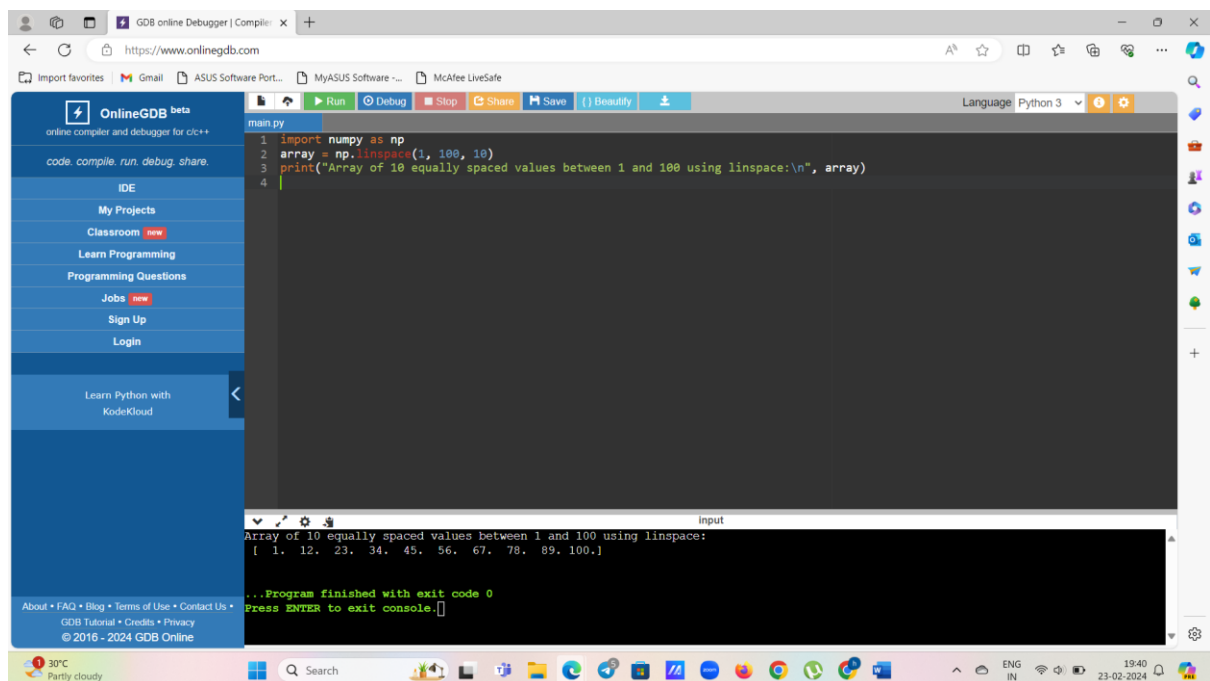
```
Enter the start value: 0
Enter the stop value: 10
Enter the number of samples: 20
Array of evenly spaced numbers using linspace:
[ 0.          0.52631579  1.05263158  1.57894737  2.10526316  2.63157895
 3.15789474  3.68421053  4.21052632  4.73684211  5.26315789  5.78947368
 6.31578947  6.84210526  7.36842105  7.89473684  8.42105263  8.94736842
 9.47368421 10.]
```

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

```
import numpy as np
```

```
array = np.linspace(1, 100, 10)
```

```
print("Array of 10 equally spaced values between 1 and 100  
using linspace:\n", array)
```



The screenshot shows the OnlineGDB web IDE interface. The left sidebar contains navigation links: IDE, My Projects, Classroom, Learn Programming, Programming Questions, Jobs, Sign Up, and Login. The main editor area displays a Python script named 'main.py' with the following code:

```
1 import numpy as np
2 array = np.linspace(1, 100, 10)
3 print("Array of 10 equally spaced values between 1 and 100 using linspace:\n", array)
4
```

The output console at the bottom shows the program's execution results:

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

...Program finished with exit code 0
Press ENTER to exit console.
```

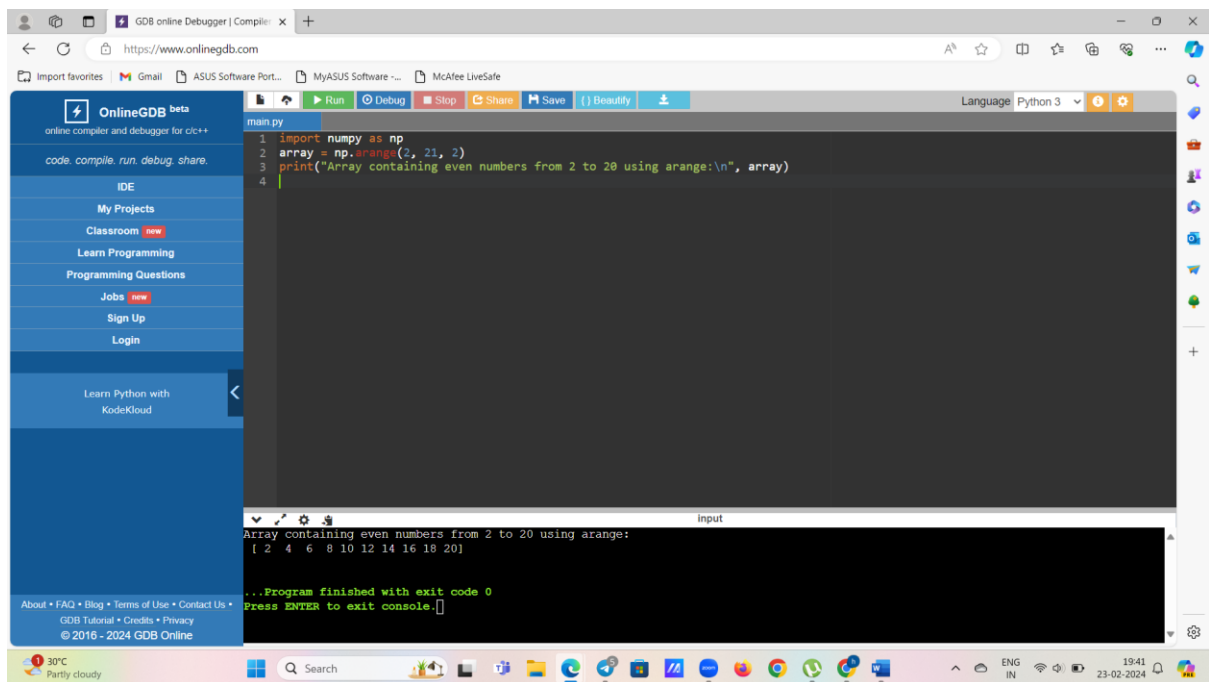
The browser's address bar shows the URL 'https://www.onlinegdb.com'. The bottom status bar indicates the system temperature is 30°C, the weather is 'Partly cloudy', and the time is 19:40 on 23-02-2024.

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

```
import numpy as np
```

```
array = np.arange(2, 21, 2)
```

```
print("Array containing even numbers from 2 to 20 using arange:\n",  
array)
```



The screenshot shows the OnlineGDB web IDE interface. The left sidebar contains navigation links: IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Jobs (new), Sign Up, and Login. Below these is a section for 'Learn Python with KodeKloud' and a footer with links to About, FAQ, Blog, Terms of Use, Contact Us, GDB Tutorial, Credits, Privacy, and Copyright (© 2016 - 2024 GDB Online). The main editor area displays a Python script in 'main.py' with the following code:

```
1 import numpy as np
2 array = np.arange(2, 21, 2)
3 print("Array containing even numbers from 2 to 20 using arange:\n", array)
4
```

The output console at the bottom shows the program's execution results:

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

...Program finished with exit code 0
Press ENTER to exit console.
```

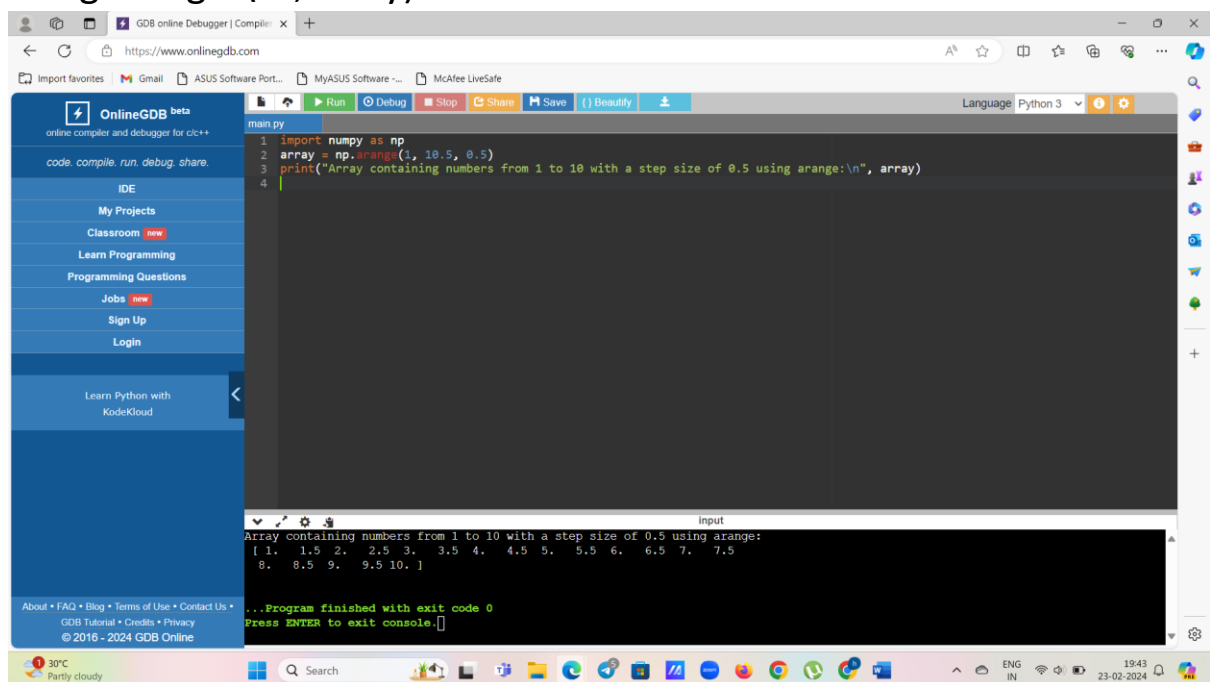
The browser's address bar shows the URL 'https://www.onlinegdb.com'. The Windows taskbar at the bottom indicates a temperature of 30°C, 'Partly cloudy' weather, and the date '23-02-2024'.

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

```
import numpy as np
```

```
array = np.arange(1, 10.5, 0.5)
```

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



The screenshot shows the OnlineGDB web interface. The code editor contains the following Python code:

```
main.py
1 import numpy as np
2 array = np.arange(1, 10.5, 0.5)
3 print("Array containing numbers from 1 to 10 with a step size of 0.5 using arange:\n", array)
4
```

The console output shows the execution result:

```
input
Array containing numbers from 1 to 10 with a step size of 0.5 using arange:
[ 1.  1.5  2.  2.5  3.  3.5  4.  4.5  5.  5.5  6.  6.5  7.  7.5
  8.  8.5  9.  9.5 10.]

...Program finished with exit code 0
Press ENTER to exit console.
```

The interface also includes a sidebar with navigation links (IDE, My Projects, Classroom, Learn Programming, Programming Questions, Jobs, Sign Up, Login) and a footer with site information (About, FAQ, Blog, Terms of Use, Contact Us, GDB Tutorial, Credits, Privacy, © 2016 - 2024 GDB Online).