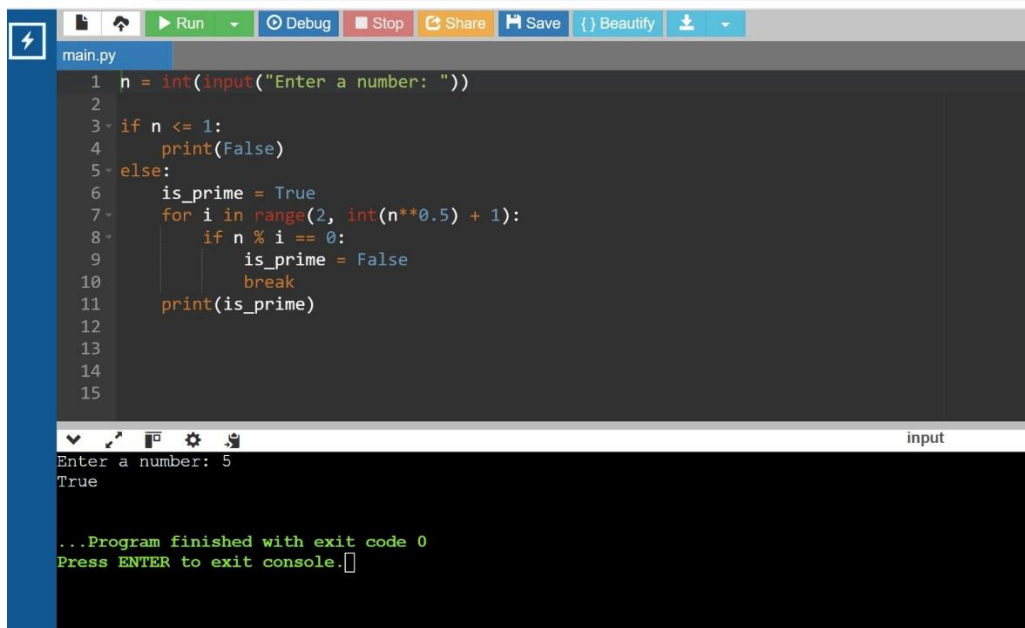


PYTHON DEVELOPER

Task 2

9. Prime Number



The screenshot shows a Python IDE with a file named 'main.py'. The code is a prime number checker. It prompts the user to 'Enter a number: ', reads the input, and checks if it's a prime. For the input '5', it prints 'True'. The console output shows the program finished with exit code 0.

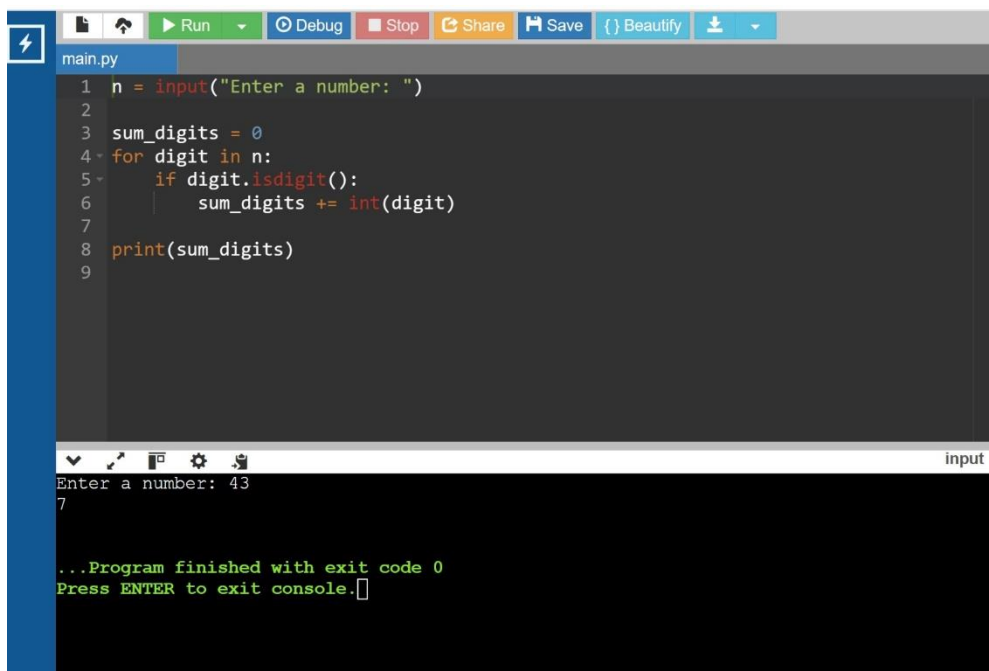
```
1 n = int(input("Enter a number: "))
2
3 if n <= 1:
4     print(False)
5 else:
6     is_prime = True
7     for i in range(2, int(n**0.5) + 1):
8         if n % i == 0:
9             is_prime = False
10            break
11    print(is_prime)
12
13
14
15
```

input

Enter a number: 5
True

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

10. Sum of Digits



The screenshot shows a Python IDE with a file named 'main.py'. The code calculates the sum of digits of a number. It prompts the user to 'Enter a number: ', reads the input, and prints the sum of its digits. For the input '43', it prints '7'. The console output shows the program finished with exit code 0.

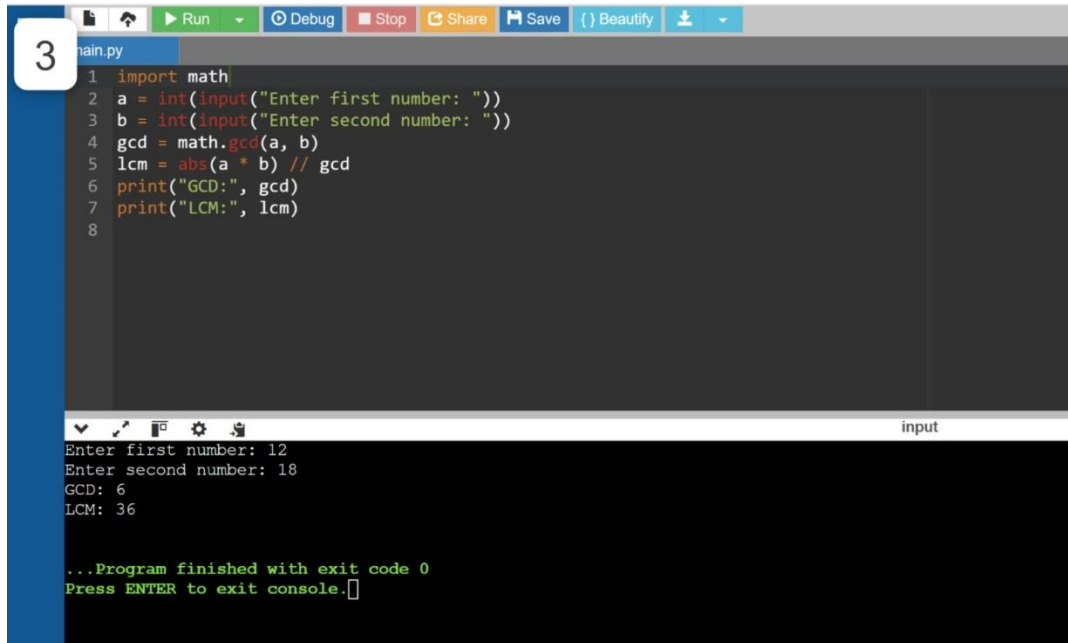
```
1 n = input("Enter a number: ")
2
3 sum_digits = 0
4 for digit in n:
5     if digit.isdigit():
6         sum_digits += int(digit)
7
8 print(sum_digits)
9
```

input

Enter a number: 43
7

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

11. LCM and GCD



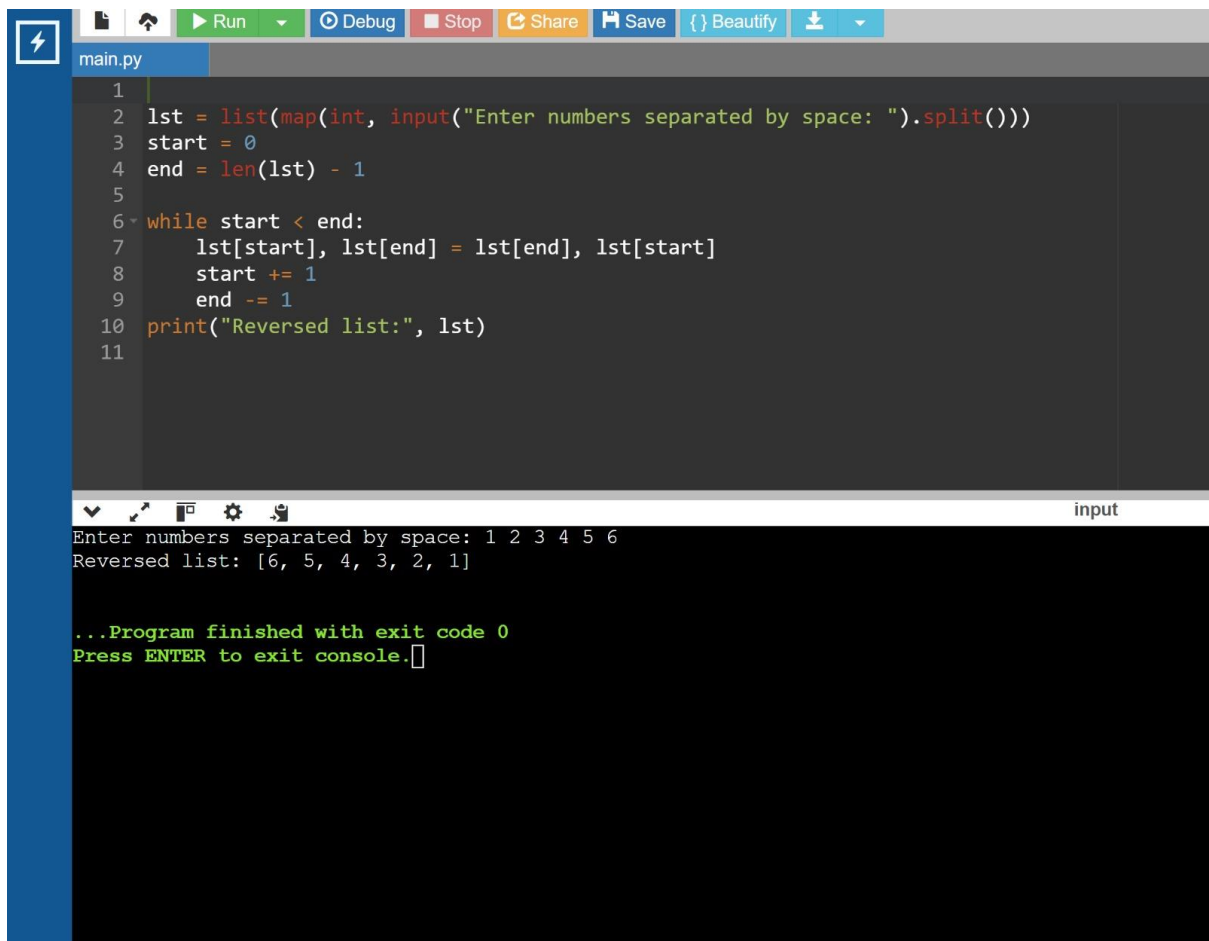
The screenshot shows a Python IDE with a file named 'main.py'. The code calculates the Least Common Multiple (LCM) and Greatest Common Divisor (GCD) of two numbers. The user has entered 12 for the first number and 18 for the second number. The output shows GCD: 6 and LCM: 36. The program finished with exit code 0.

```
1 import math
2 a = int(input("Enter first number: "))
3 b = int(input("Enter second number: "))
4 gcd = math.gcd(a, b)
5 lcm = abs(a * b) // gcd
6 print("GCD:", gcd)
7 print("LCM:", lcm)
8
```

Enter first number: 12
Enter second number: 18
GCD: 6
LCM: 36

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

12. List Reversal



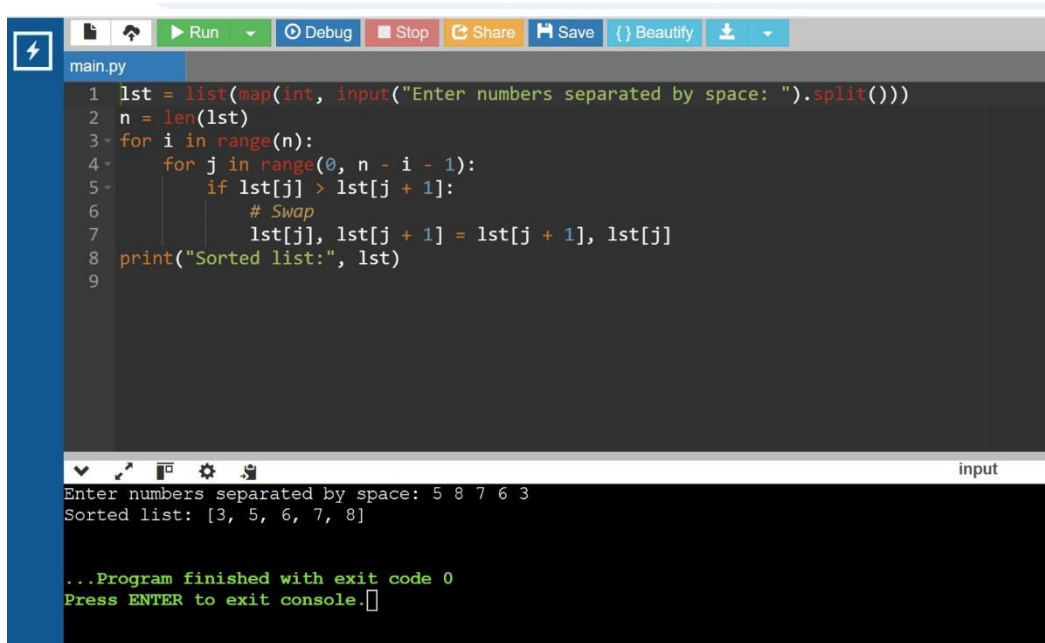
The screenshot shows a Python IDE with a file named 'main.py'. The code reverses a list of numbers entered by the user. The user has entered '1 2 3 4 5 6'. The output shows the reversed list: [6, 5, 4, 3, 2, 1]. The program finished with exit code 0.

```
1
2 lst = list(map(int, input("Enter numbers separated by space: ").split()))
3 start = 0
4 end = len(lst) - 1
5
6 while start < end:
7     lst[start], lst[end] = lst[end], lst[start]
8     start += 1
9     end -= 1
10 print("Reversed list:", lst)
11
```

Enter numbers separated by space: 1 2 3 4 5 6
Reversed list: [6, 5, 4, 3, 2, 1]

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

13. Sort a List



The screenshot shows a Python IDE with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, Beautify, and a dropdown menu. The editor window, titled 'main.py', contains the following Python code:

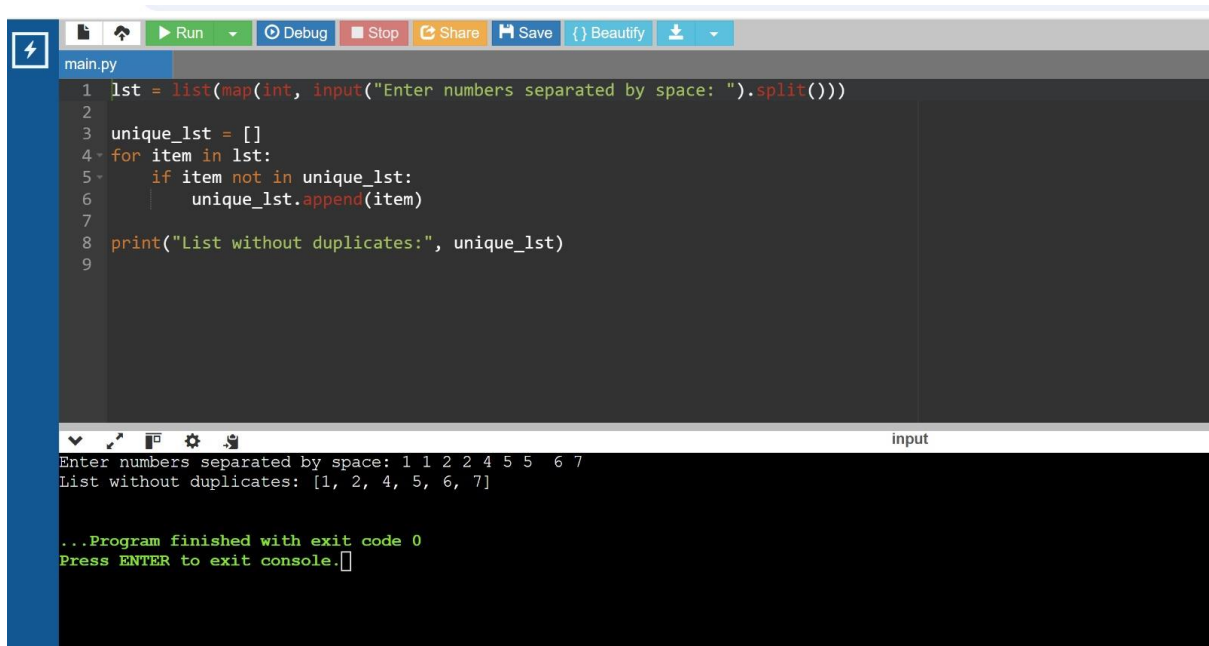
```
1 lst = list(map(int, input("Enter numbers separated by space: ").split()))
2 n = len(lst)
3 for i in range(n):
4     for j in range(0, n - i - 1):
5         if lst[j] > lst[j + 1]:
6             # Swap
7             lst[j], lst[j + 1] = lst[j + 1], lst[j]
8 print("Sorted list:", lst)
9
```

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

```
Enter numbers separated by space: 5 8 7 6 3
Sorted list: [3, 5, 6, 7, 8]

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

14. Remove Duplicates



The screenshot shows a Python IDE with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, Beautify, and a dropdown menu. The editor window, titled 'main.py', contains the following Python code:

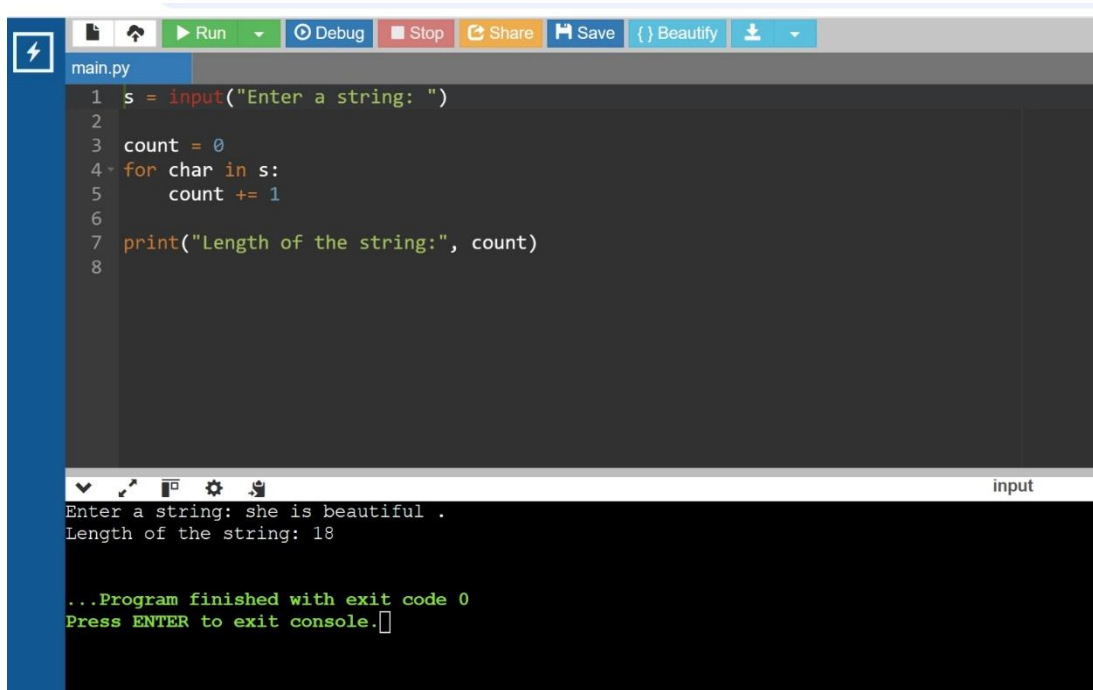
```
1 lst = list(map(int, input("Enter numbers separated by space: ").split()))
2
3 unique_lst = []
4 for item in lst:
5     if item not in unique_lst:
6         unique_lst.append(item)
7
8 print("List without duplicates:", unique_lst)
9
```

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

```
Enter numbers separated by space: 1 1 2 2 4 5 5 6 7
List without duplicates: [1, 2, 4, 5, 6, 7]

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

15. String Length

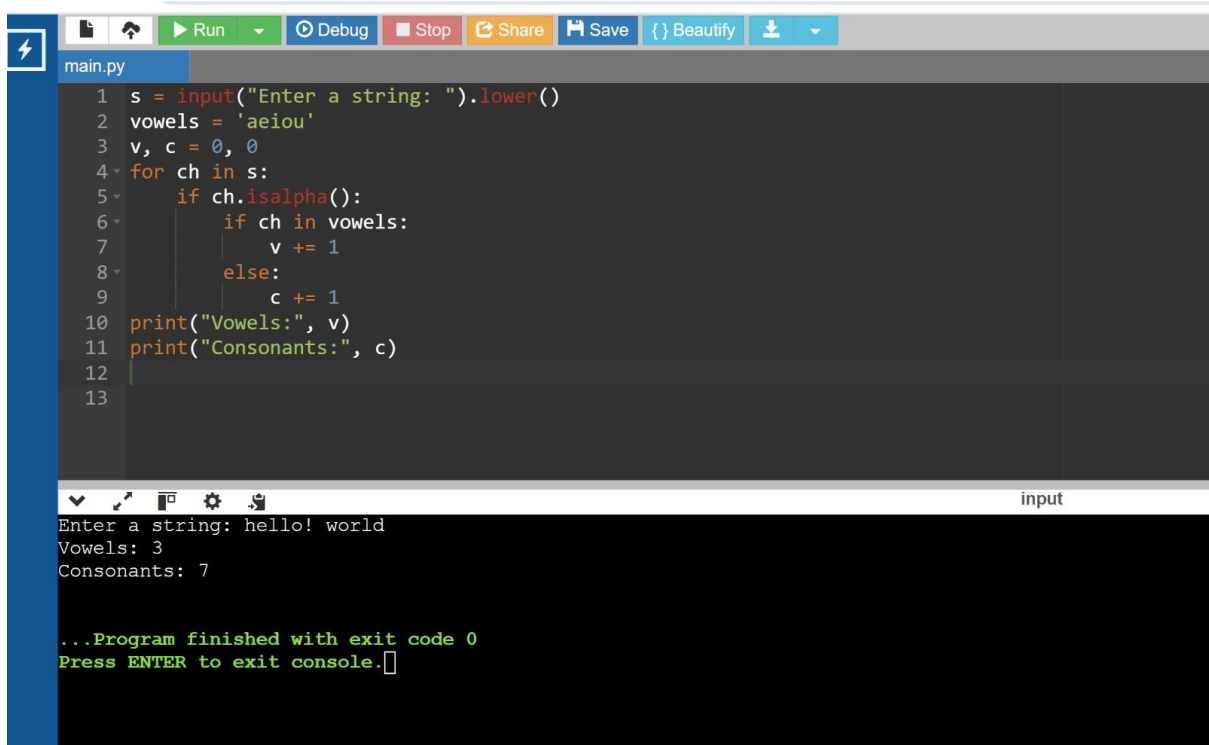


The screenshot shows a code editor with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, Beautify, and a dropdown menu. The file name 'main.py' is visible. The code in the editor is as follows:

```
1 s = input("Enter a string: ")
2
3 count = 0
4 for char in s:
5     count += 1
6
7 print("Length of the string:", count)
8
```

Below the code editor is a console window. It shows the program's execution with the input 'she is beautiful .' and the output 'Length of the string: 18'. The console also displays the message '...Program finished with exit code 0' and 'Press ENTER to exit console.'

6. Count Vowels and Consonants;



The screenshot shows a code editor with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, Beautify, and a dropdown menu. The file name 'main.py' is visible. The code in the editor is as follows:

```
1 s = input("Enter a string: ").lower()
2 vowels = 'aeiou'
3 v, c = 0, 0
4 for ch in s:
5     if ch.isalpha():
6         if ch in vowels:
7             v += 1
8         else:
9             c += 1
10 print("Vowels:", v)
11 print("Consonants:", c)
12
13
```

Below the code editor is a console window. It shows the program's execution with the input 'hello! world' and the output 'Vowels: 3' and 'Consonants: 7'. The console also displays the message '...Program finished with exit code 0' and 'Press ENTER to exit console.'

2. Maze Generator and Solver

a.

```
main.py
1 import random
2
3 def create_maze(width, height):
4     maze = [[1 for _ in range(width)] for _ in range(height)]
5
6     def carve_passages(x, y):
7         directions = [(2,0), (-2,0), (0,2), (0,-2)]
8         random.shuffle(directions)
9
10        for dx, dy in directions:
11            nx, ny = x + dx, y + dy
12            if 1 <= nx < width-1 and 1 <= ny < height-1 and maze[ny][nx] == 1:
13                maze[ny][nx] = 0
14                maze[y + dy//2][x + dx//2] = 0
15                carve_passages(nx, ny)
16
17    maze[1][1] = 0
18    carve_passages(1, 1)
19    return maze
20
21 def solve_maze(maze):
22     height = len(maze)
23     width = len(maze[0])
24     start = (1, 1)
25     end = (width - 2, height - 2)
26     path = []
27     visited = set()
28
29     def dfs(x, y):
30         if not (0 <= x < width and 0 <= y < height):
31             return False
32         if maze[y][x] == 1 or (x,y) in visited:
33             return False
34         path.append((x,y))
35         visited.add((x,y))
```

b.

```
main.py
35         visited.add((x,y))
36         if (x,y) == end:
37             return True
38         if dfs(x+1, y) or dfs(x-1, y) or dfs(x, y+1) or dfs(x, y-1):
39             return True
40         path.pop()
41         return False
42
43     dfs(*start)
44     return path
45
46 def print_maze(maze, path=None):
47     for y, row in enumerate(maze):
48         line = ""
49         for x, cell in enumerate(row):
50             if path and (x,y) in path:
51                 line += "*"
52             elif cell == 1:
53                 line += "#"
54             else:
55                 line += " "
56         print(line)
57
58 def main():
59     width, height = 21, 21 # Must be odd
60     maze = create_maze(width, height)
61     path = solve_maze(maze)
62     print("Maze:")
63     print_maze(maze)
64     print("\nSolved Path:")
65     print_maze(maze, path)
66
67 if __name__ == "__main__":
68     main()
69
```

c. OUTPUT

```
⚡ ✓ ↗ 📄 ⚙️ 📁
Maze:
#####
# #      #      #
# ###   ###   ###   ###
#      # #      #   #
#####   ###   #####
#      #      #      #
# #   ###   #####   #####
# #      # #      #      #
# ###   # #   #####   #####
# #      # #   #   #   #
# #   ###   # #   #   ###
# #      # #   #   #   #
# #####   # #   #   #   #
#      #   #   #   #   #
#####   # #   #####
#      #      #      #
# #####   #####
#      #      #      #
#   ###   #####   ###
#      #      #      #
#####

Solved Path:
#####
#*#   *****#      #
#*###*###*###   ###   ###
#*****#   ***#      #
#####   ###*#####
#***   #*****#*****#
#*#*###*#####*#####*#
#*#*****#*#####*#####
#*###*#*#*#####*#####
#*#*#*#*#*#   #*#   #
#*#*###*#*#   #   #*###
#*#*****#*#   #   #*#*#
#*#####*#*#   #   #*#*#
#*****#*#*#   #   #*#*#
#####*#*#   #####*#
#      #*#*#*****#
```