

Part A – Function Basics (20 points)

1. Define a function `triple(number)` that returns $3 \times$ the number.

Code: `def triple(number):`
 `return 3 * number`

2. Define a function `is_even(num)` that returns `True` if even, otherwise `False`.

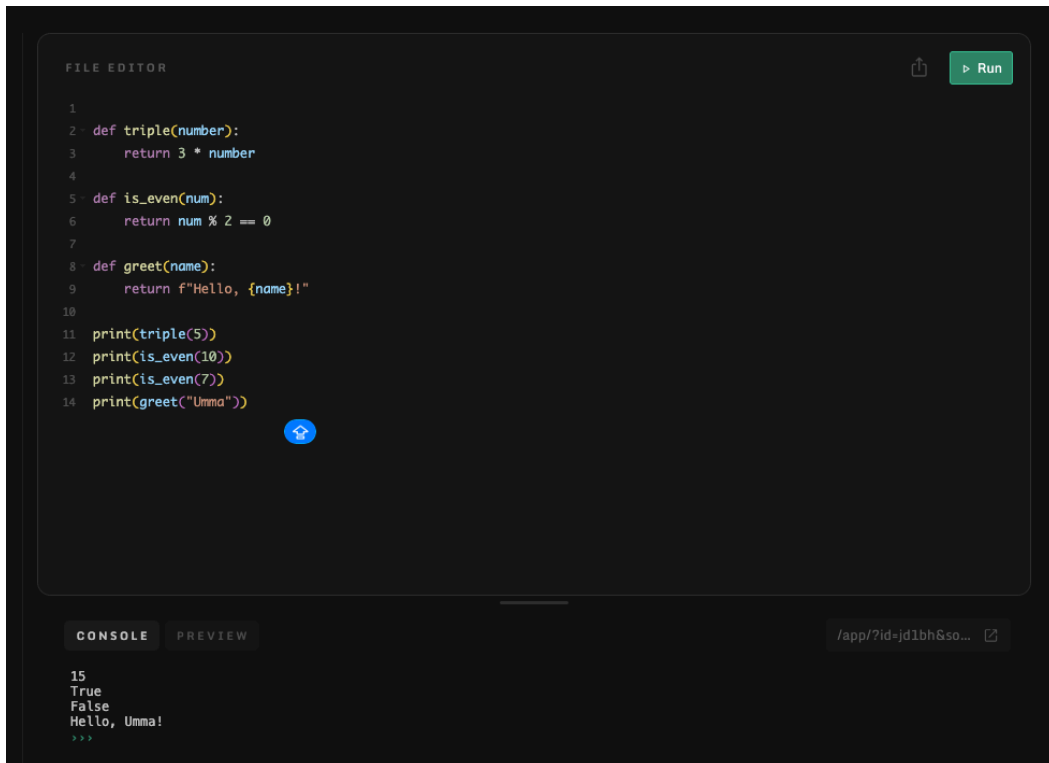
Code: `def is_even(num):`
 `return num % 2 == 0`

3. Define a function `greet(name)` that returns a greeting message.

Code: `def greet(name):`
 `return f"Hello, {name}!"`

4. Test all three functions by printing the results.

Code: `print(triple(5))`
`print(is_even(10))`
`print(is_even(7))`
`print(greet("Umma"))`



The screenshot shows a code editor with a dark theme. The code is as follows:

```
1
2 def triple(number):
3     return 3 * number
4
5 def is_even(num):
6     return num % 2 == 0
7
8 def greet(name):
9     return f"Hello, {name}!"
10
11 print(triple(5))
12 print(is_even(10))
13 print(is_even(7))
14 print(greet("Umma"))
```

Below the code editor, there is a console output showing the results of the execution:

```
15
True
False
Hello, Umma!
```

Part B – Functions With Lists (25 points)

temperatures = [72, 68, 75, 80, 79, 66, 71]

1. Write `above_75(temp)` that returns `True` if `temp > 75`.

Code: `temperature = [72,68,75,80,79,66,71]`

`def above_75(temp):`

`return temp > 75`

2. Use a for loop and the function to create a list of `hot_days`.

Code: `hot_days = []`

`for t in temperatures:`

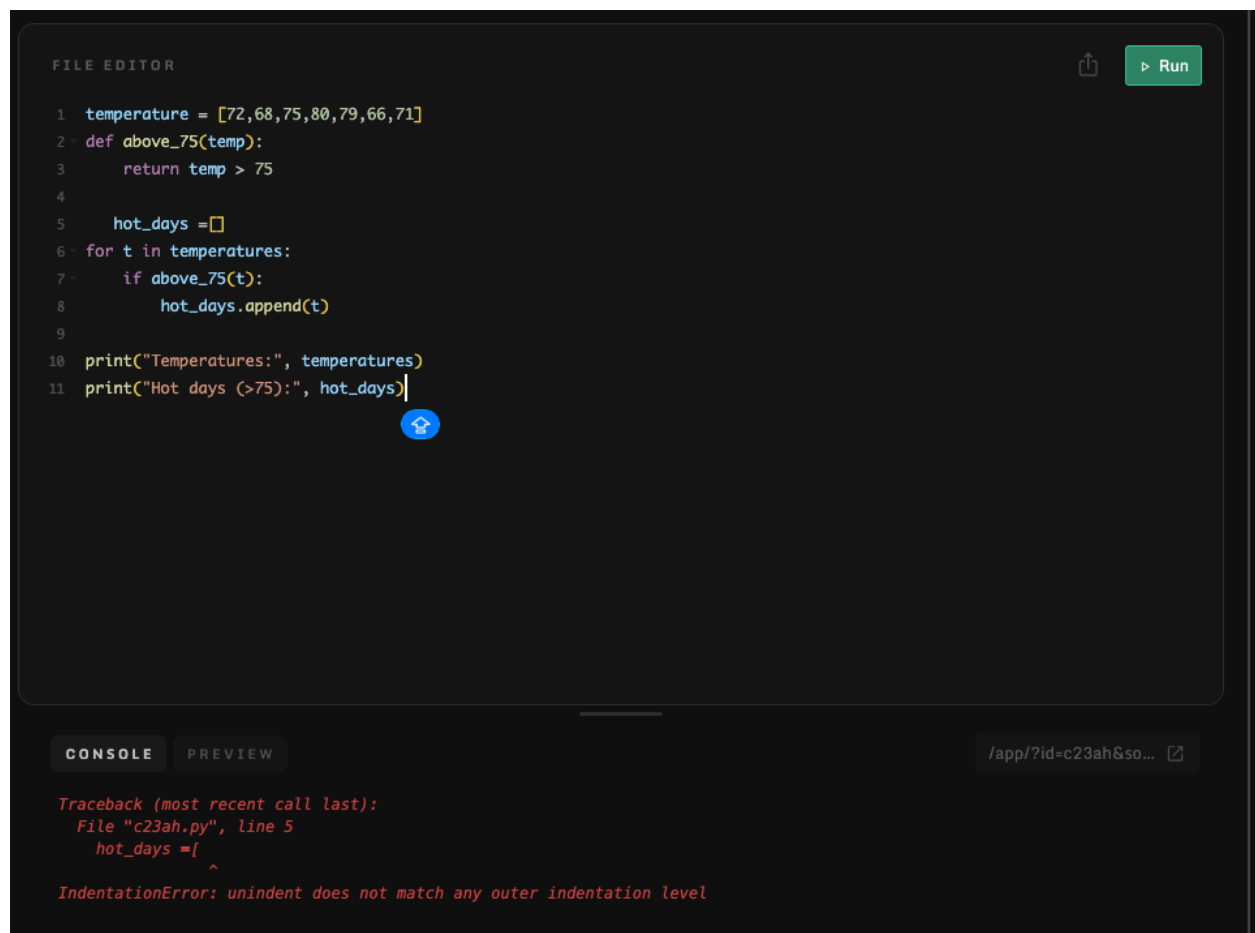
`if above_75(t):`

`hot_days.append(t)`

3. Print original temperatures and `hot_days`.

Code: `print("Temperatures:", temperatures)`

`print("Hot days (>75):", hot_days)`



The screenshot shows a code editor with the following Python code:

```
1 temperature = [72,68,75,80,79,66,71]
2 def above_75(temp):
3     return temp > 75
4
5 hot_days = []
6 for t in temperatures:
7     if above_75(t):
8         hot_days.append(t)
9
10 print("Temperatures:", temperatures)
11 print("Hot days (>75):", hot_days)
```

The console shows a `Traceback (most recent call last):` error:

```
File "c23ah.py", line 5
hot_days = [
^
IndentationError: unindent does not match any outer indentation level
```

Part C – Functions + Dictionaries + Conditionals (25 points)

```
students = { 'Alice': 85, 'Ben': 92, 'Carla': 78, 'David': 88 }
```

1. Write `letter_grade(score)` returning A, B, C, D, or F.

Code : `def letter_grade(score):`

```
    if score >= 90:
```

```
        return "A"
```

```
    elif score >= 80:
```

```
        return "B"
```

```
    elif score >= 70:
```

```
        return "C"
```

```
    elif score >= 60:
```

```
        return "D"
```

```
    else:
```

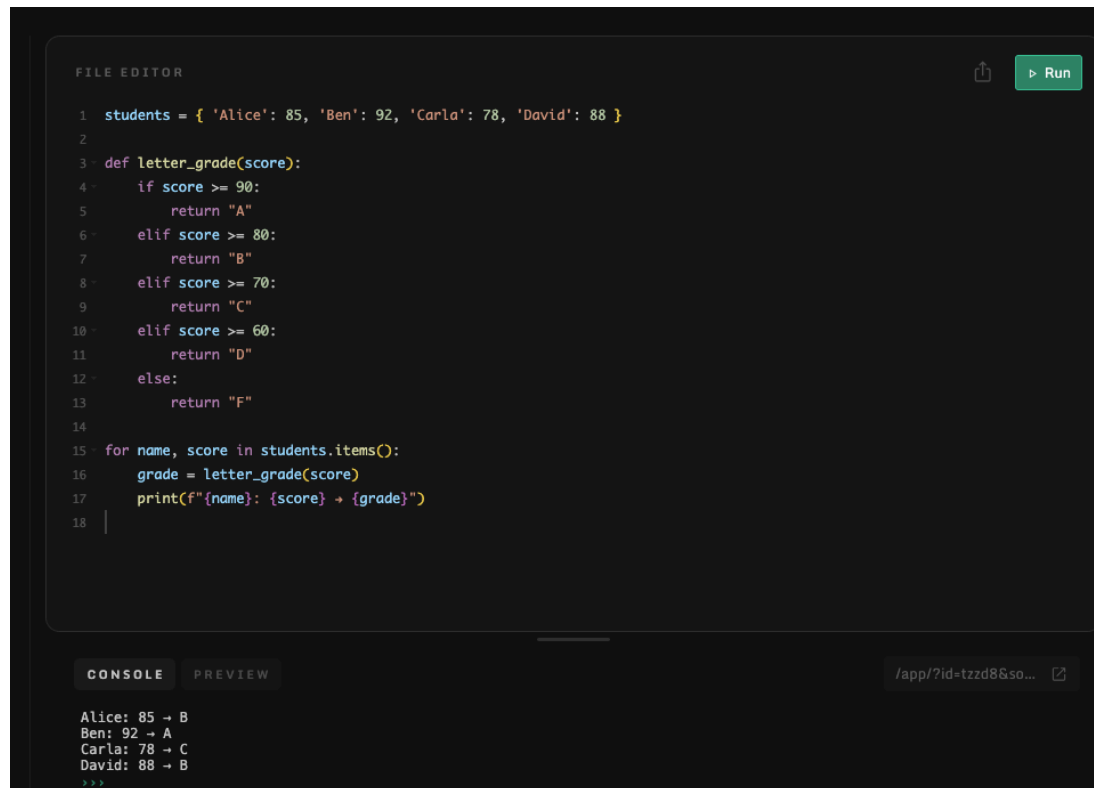
```
        return "F"
```

2. Loop through `students` and print `Name: score → letter`

Code: `for name, score in students.items():`

```
    grade = letter_grade(score)
```

```
    print(f'{name}: {score} → {grade}')
```



The screenshot shows a code editor with a dark theme. At the top right, there is a 'Run' button. The code is as follows:

```
1 students = { 'Alice': 85, 'Ben': 92, 'Carla': 78, 'David': 88 }
2
3 def letter_grade(score):
4     if score >= 90:
5         return "A"
6     elif score >= 80:
7         return "B"
8     elif score >= 70:
9         return "C"
10    elif score >= 60:
11        return "D"
12    else:
13        return "F"
14
15 for name, score in students.items():
16     grade = letter_grade(score)
17     print(f'{name}: {score} → {grade}')
18
```

At the bottom, there is a 'CONSOLE' tab showing the output:

```
Alice: 85 → B
Ben: 92 → A
Carla: 78 → C
David: 88 → B
```

Part D – While Loop + Functions (20 points)

Write `find_max(numbers)` using a while loop to find the largest number.

Use `nums = [5, 9, 2, 17, 3, 11]` and print the maximum.

Code: `def find_max(numbers):`

```
    i = 0
```

```
    max_value = numbers[0]
```

```
    while i < len(numbers):
```

```
        if numbers[i] > max_value:
```

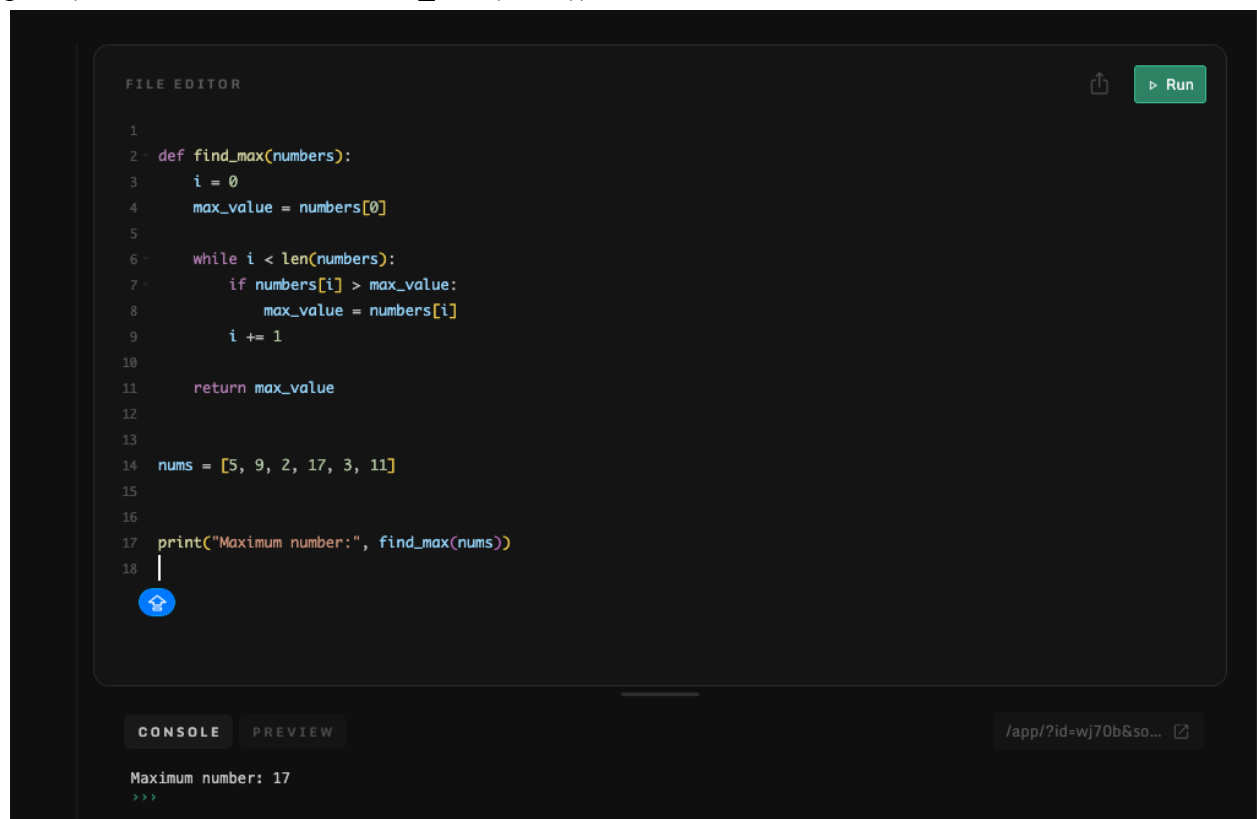
```
            max_value = numbers[i]
```

```
        i += 1
```

```
    return max_value
```

```
nums = [5, 9, 2, 17, 3, 11]
```

```
print("Maximum number:", find_max(nums))
```



The screenshot shows a code editor with a dark theme. The code is as follows:

```
1
2 def find_max(numbers):
3     i = 0
4     max_value = numbers[0]
5
6     while i < len(numbers):
7         if numbers[i] > max_value:
8             max_value = numbers[i]
9         i += 1
10
11     return max_value
12
13
14 nums = [5, 9, 2, 17, 3, 11]
15
16
17 print("Maximum number:", find_max(nums))
18
```

At the bottom, there is a console output showing "Maximum number: 17".

Part E – Create Your Own Function (10 points)

Create a function for a real - world task (e.g., convert minutes, count vowels).
Must include at least one argument and a conditional or loop.

Code: `def count_vowels(text):`

`vowels = "aeiouAEIOU"`

`count = 0`

`for char in text:`

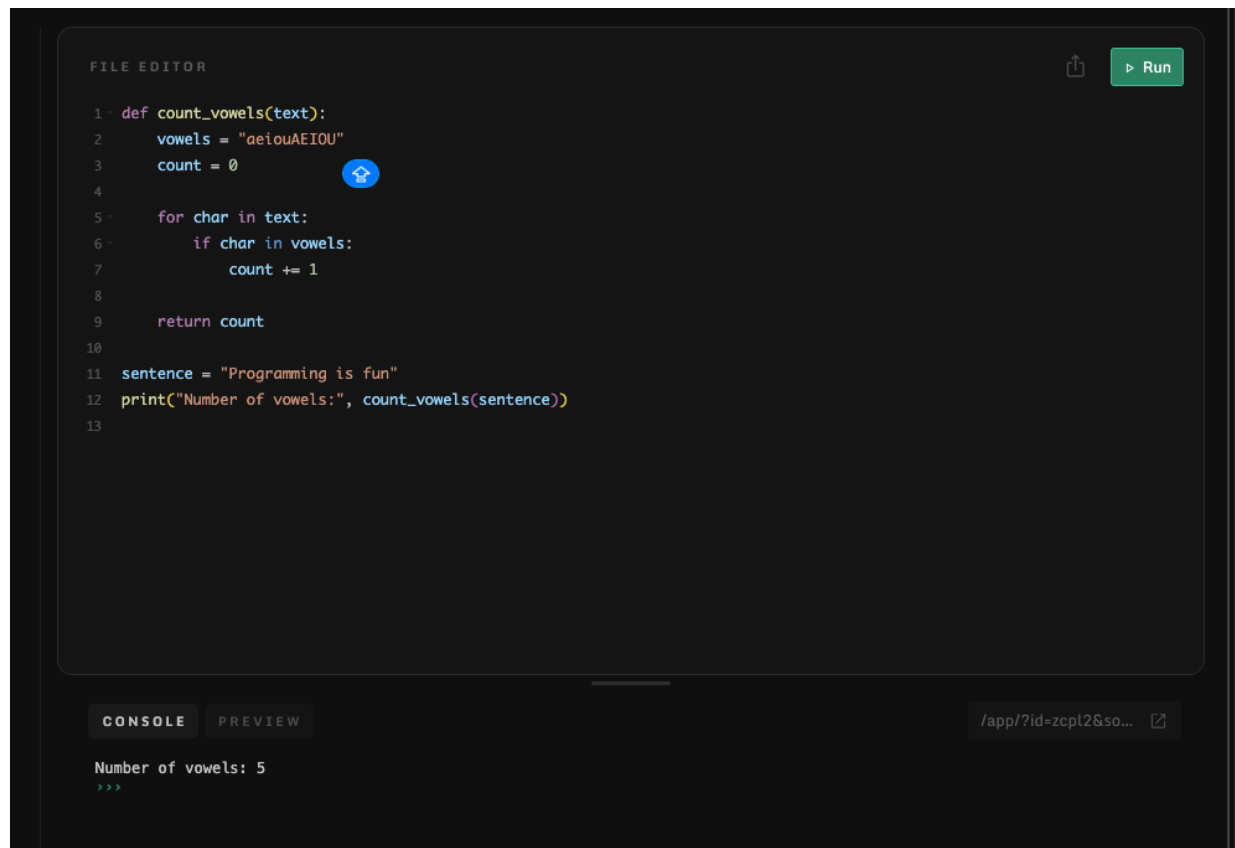
`if char in vowels:`

`count += 1`

`return count`

`sentence = "Programming is fun"`

`print("Number of vowels:", count_vowels(sentence))`



The screenshot shows a code editor with a dark theme. The editor has a 'FILE EDITOR' tab at the top left and a 'Run' button at the top right. The code is as follows:

```
1 def count_vowels(text):
2     vowels = "aeiouAEIOU"
3     count = 0
4
5     for char in text:
6         if char in vowels:
7             count += 1
8
9     return count
10
11 sentence = "Programming is fun"
12 print("Number of vowels:", count_vowels(sentence))
13
```

Below the editor, there is a 'CONSOLE' tab and a 'PREVIEW' tab. The console shows the output of the code:

```
Number of vowels: 5
>>>
```

There is also a URL bar at the bottom right showing `/app/?id=zcpt2&so...`.

Bonus – Optional (+10 points)

Write `reverse_list(lst)` using a while loop to return a reversed list

Code: `def reverse_list(lst):`

`reversed_lst = []`

`i = len(lst) - 1`

`while i >= 0:`

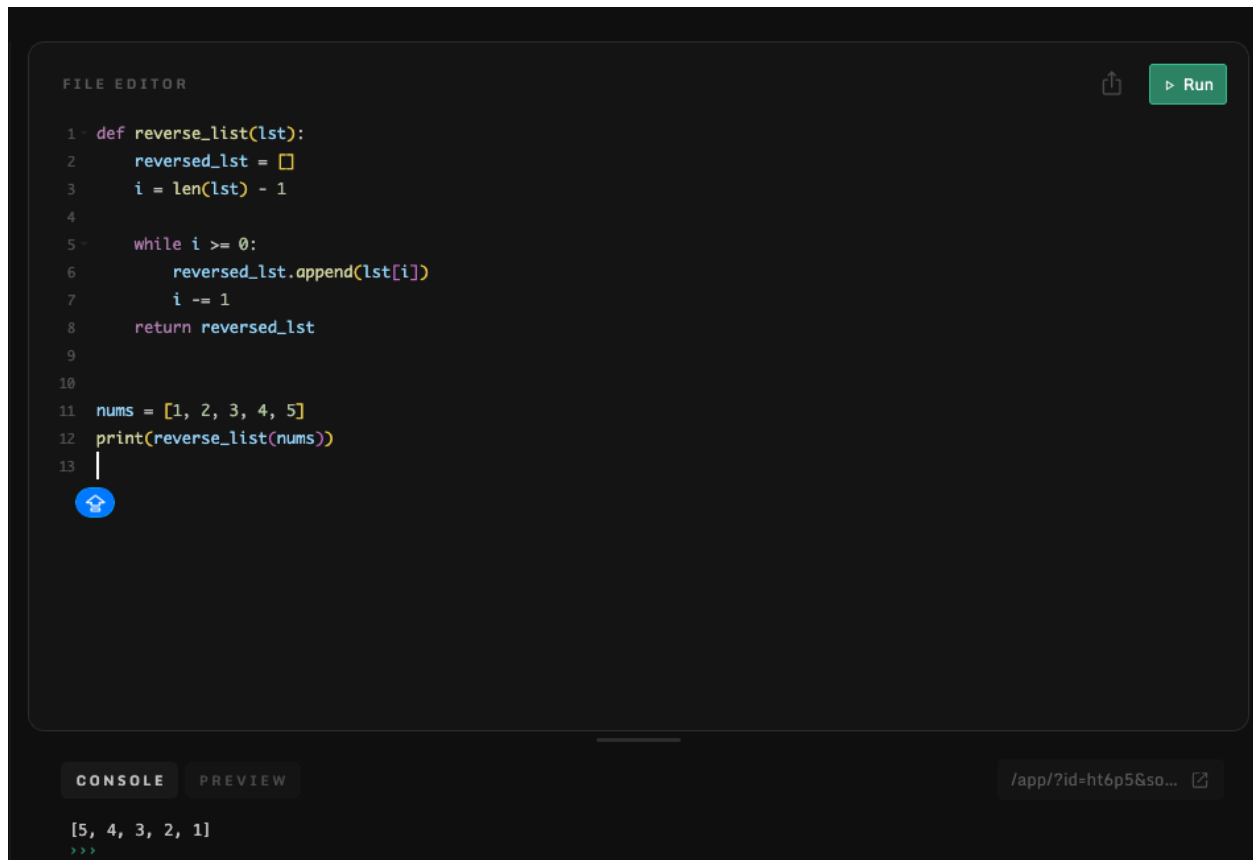
`reversed_lst.append(lst[i])`

`i -= 1`

`return reversed_lst`

`nums = [1, 2, 3, 4, 5]`

`print(reverse_list(nums))`



The screenshot shows a code editor with a dark theme. The code is as follows:

```
1 def reverse_list(lst):
2     reversed_lst = []
3     i = len(lst) - 1
4
5     while i >= 0:
6         reversed_lst.append(lst[i])
7         i -= 1
8     return reversed_lst
9
10
11 nums = [1, 2, 3, 4, 5]
12 print(reverse_list(nums))
13
```

Below the code editor, there is a console output showing the result of the execution:

```
[5, 4, 3, 2, 1]
```

The interface includes a 'Run' button in the top right corner and a 'Console' tab at the bottom left.