

## 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 interface with a dark theme. At the top, there's a 'FILE EDITOR' tab and a 'Run' button. The code area contains the following Python code:

```
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 is a 'CONSOLE' tab and a 'PREVIEW' tab. The console output shows the results of running the code:

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

The URL at the bottom of the console window is /app/?id=jd1bh&so... .

## 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 Python code editor interface. The code in the file editor is as follows:

```
FILE EDITOR
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 code contains a syntax error at line 11, where the closing parenthesis for the print statement is missing. A blue bracket icon is positioned over the closing parenthesis of the second print statement to highlight the error. The editor has tabs for 'FILE EDITOR' and 'CONSOLE'. In the 'CONSOLE' tab, the following traceback is displayed:

```
CONSOLE
Traceback (most recent call last):
File "c23ah.py", line 5
    hot_days =[
               ^
IndentationError: unindent does not match any outer indentation level
```

The URL in the browser bar is /app/?id=c23ah&so... .

### 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}')
```

```
FILE EDITOR  
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 |
```

CONSOLE PREVIEW /app/?id=tzzd8&so... ↻

```
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 interface with a dark theme. The code area contains the provided Python script. At the bottom, the console tab is active, displaying the output of the `print` statement. The preview tab is also visible. A green "Run" button is located in the top right corner of the editor.

```
FILE EDITOR
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
```

CONSOLE PREVIEW /app/?id=wj7Ob&so... ↻

```
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 interface with a dark theme. The top bar has a file icon, a run button labeled 'Run', and a preview button. The left side is labeled 'FILE EDITOR' and contains the Python code. The right side is labeled 'CONSOLE' and shows the output of the code execution.

```
FILE EDITOR
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
```

CONSOLE /app/?id=zcpl2&so... □

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

## 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 interface with a dark theme. The code area contains the provided Python script. The editor has tabs for 'FILE EDITOR' and 'CONSOLE'. In the 'CONSOLE' tab, the output of the script is shown: `[5, 4, 3, 2, 1]`. There is also a preview link: `/app/?id=ht6p5&so...`.

```
FILE EDITOR
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 |
```

CONSOLE PREVIEW /app/?id=ht6p5&so...

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