Practical No 5

Aim: To use lambda functions and higher-order functions like map() in Python for functional programming tasks.

Theory:

Lambda Function

- A **lambda** function is an anonymous, one-line function with no name.

Syntax:

```
lambda arguments: expression
```

Example:

```
square = lambda x: x * x
print(square(5))
```

Higher-Order Functions

- These are functions that take other functions as arguments.
- Applies a function to all items in an iterable and returns a `map` object (which can be converted to a list).

Syntax:

```
map(function, iterable)
```

Example:

```
numbers = [1, 2, 3, 4]
squares = list(map(lambda x: x**2, numbers))
print(squares)
```

Write a Python program using a lambda function with **map() to add** 18% GST **to a list of product prices and display both the original and final prices.**

Program

```
# List of prices before GST
prices = [100, 200, 300, 400]

# Lambda function to add 18% GST
add_gst = lambda price: price + (price * 0.18)

# Using map() to apply GST to all prices
final_prices = list(map(add_gst, prices))
```

```
# Display results
print("Original Prices:", prices)
print("Final Prices with GST (18%):", final_prices)

# Squaring numbers using lambda and map
numbers = [2, 4, 6, 8]
squares = list(map(lambda x: x**2, numbers))
print("Squares:", squares)
```

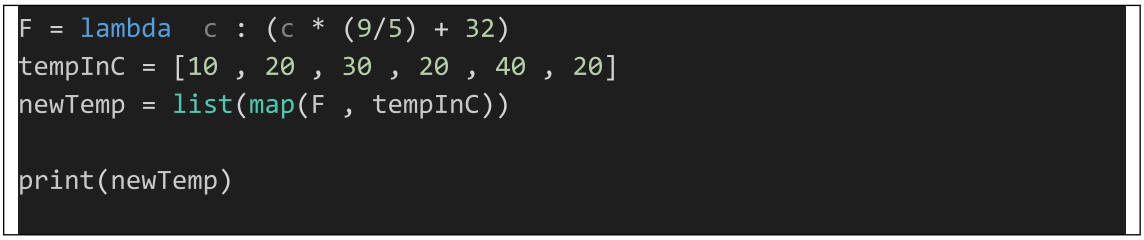
Output

```
Original Prices: [100, 200, 300, 400]
Final Prices with GST (18%): [118.0, 236.0, 354.0, 472.0]
Squares: [4, 16, 36, 64]
```

Problem Statements:

1. Write a program using **lambda** + `map()` to convert a list of temperatures from **Celsius to Fahrenheit**.
   Formula: `F = (C × 9/5) + 32`

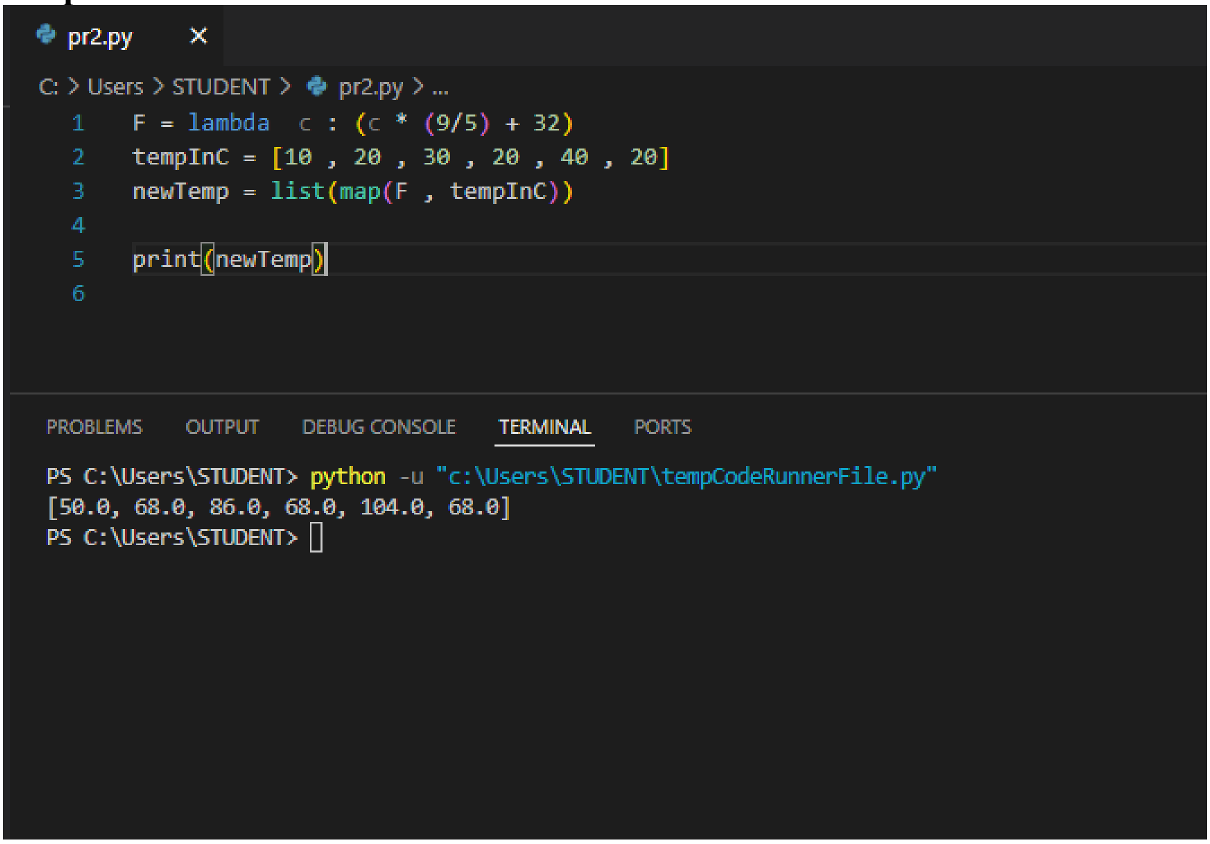
   Program:



```
F = lambda  c : (c * (9/5) + 32)
tempInC = [10 , 20 , 30 , 20 , 40 , 20]
newTemp = list(map(F , tempInC))

print(newTemp)
```

Output:



```
F = lambda  c : (c * (9/5) + 32)
tempInC = [10 , 20 , 30 , 20 , 40 , 20]
newTemp = list(map(F , tempInC))

print(newTemp)
```

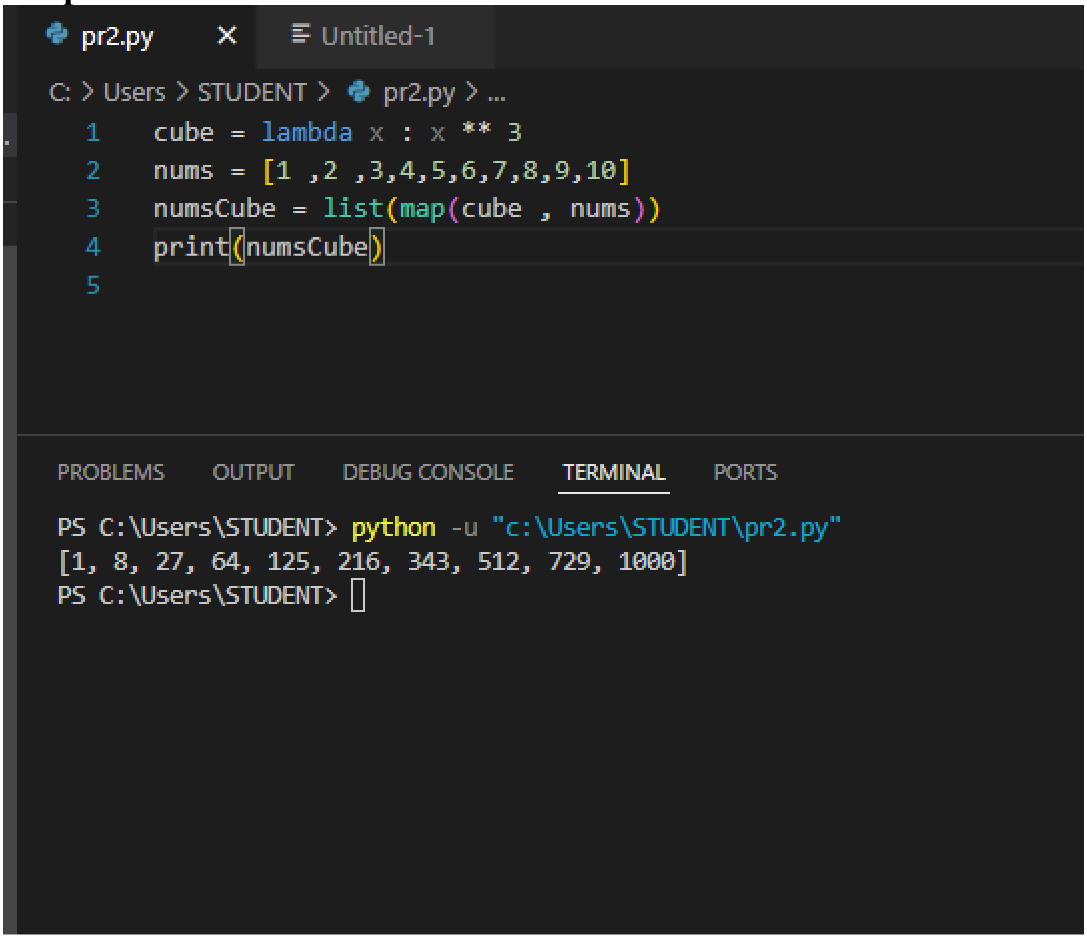
```
PS C:\Users\STUDENT> python -u "c:\Users\STUDENT\tempCodeRunnerFile.py"
[50.0, 68.0, 86.0, 68.0, 104.0, 68.0]
PS C:\Users\STUDENT> 
```

2. Write a Python program using **lambda** + `map()` to return the **cube** of each number in a given list.

Program:

```
cube = lambda x : x ** 3
nums = [1 ,2 ,3,4,5,6,7,8,9,10]
numsCube = list(map(cube , nums))
print(numsCube)
```

Output:



```python
cube = lambda x : x ** 3
nums = [1 ,2 ,3,4,5,6,7,8,9,10]
numsCube = list(map(cube , nums))
print(numsCube)
```

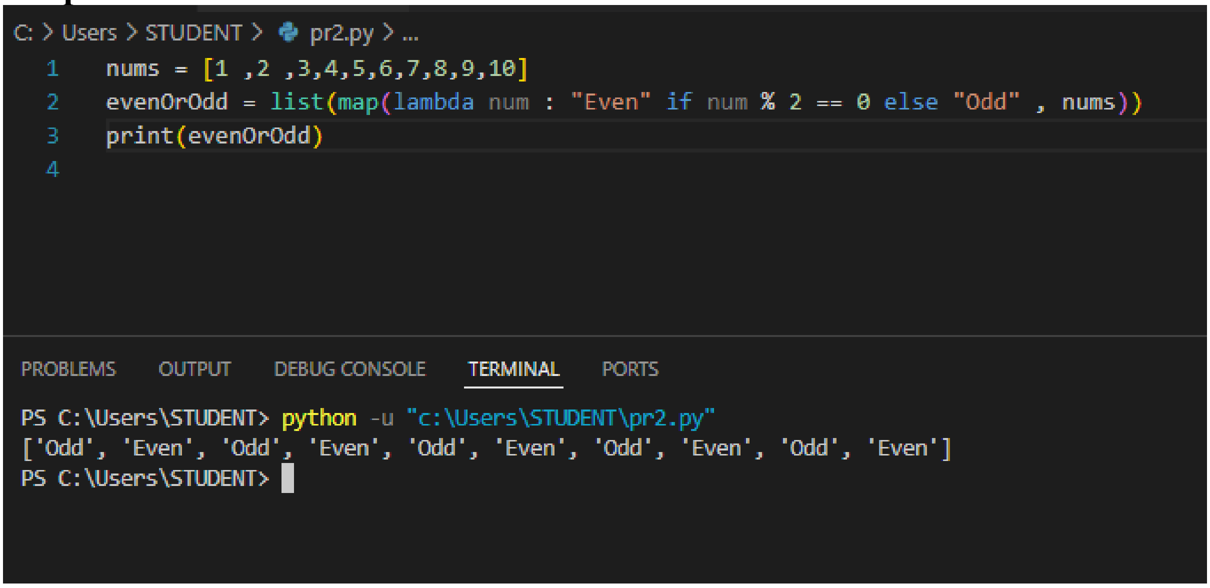
```
PS C:\Users\STUDENT> python -u "c:\Users\STUDENT\pr2.py"
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
PS C:\Users\STUDENT> 
```

3. **Given a list of integers, use a lambda function with** `map()` **to classify each number as Even or Odd.**

Program:

```python
nums = [1 ,2 ,3,4,5,6,7,8,9,10]
evenOrOdd = list(map(lambda num : "Even" if num % 2 == 0 else "Odd" ,
nums))
print(evenOrOdd)
```

Output:



```python
nums = [1 ,2 ,3,4,5,6,7,8,9,10]
evenOrOdd = list(map(lambda num : "Even" if num % 2 == 0 else "Odd" , nums))
print(evenOrOdd)
```

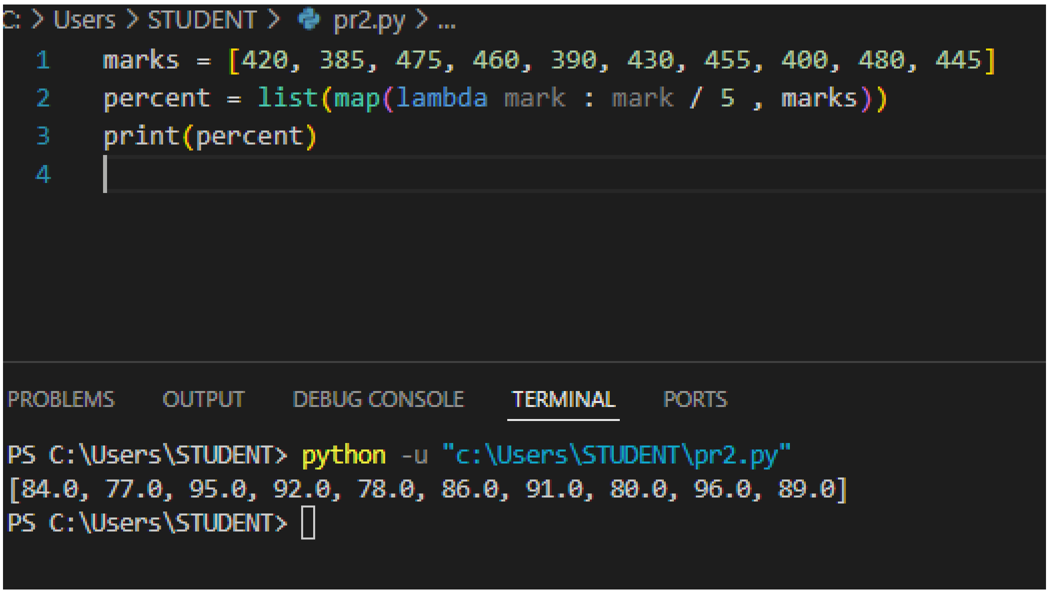
```
PS C:\Users\STUDENT> python -u "c:\Users\STUDENT\pr2.py"
['Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even']
PS C:\Users\STUDENT>
```

4. Given a list of student marks (out of **500**), use a **lambda function with** `map()` to calculate the **percentage** for each student.

Program:

```python
marks = [420, 385, 475, 460, 390, 430, 455, 400, 480, 445]
percent = list(map(lambda mark : mark / 5 , marks))
print(percent)
```

Output:



```python
marks = [420, 385, 475, 460, 390, 430, 455, 400, 480, 445]
percent = list(map(lambda mark : mark / 5 , marks))
print(percent)
```

```
PS C:\Users\STUDENT> python -u "c:\Users\STUDENT\pr2.py"
[84.0, 77.0, 95.0, 92.0, 78.0, 86.0, 91.0, 80.0, 96.0, 89.0]
PS C:\Users\STUDENT>
```

5. Given a list of student marks (out of **100**), use a **lambda function with** `map()` to calculate the **percentage** and assign a **grade** to each student.

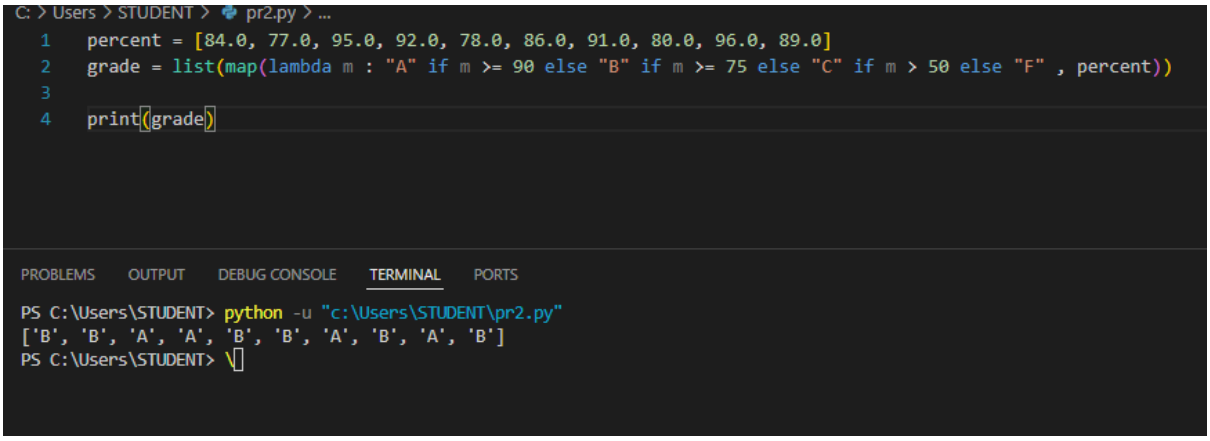
- Grade Rules:
    - ≥ 90 → A
    - 75–89 → B
    - 50–74 → C
    - < 50 → Fail

Program:

```
percent = [84.0, 77.0, 95.0, 92.0, 78.0, 86.0, 91.0, 80.0, 96.0, 89.0]
grade = list(map(lambda m : "A" if m >= 90 else "B" if m >= 75 else "C"
if m > 50 else "F" , percent))

print(grade)
```

Output:



```
percent = [84.0, 77.0, 95.0, 92.0, 78.0, 86.0, 91.0, 80.0, 96.0, 89.0]
grade = list(map(lambda m : "A" if m >= 90 else "B" if m >= 75 else "C" if m > 50 else "F" , percent))

print(grade)
```

```
PS C:\Users\STUDENT> python -u "c:\Users\STUDENT\pr2.py"
['B', 'B', 'A', 'A', 'B', 'B', 'A', 'B', 'A', 'B']
PS C:\Users\STUDENT> \
```

6. Write a Python program using a **lambda function** to calculate the **electricity bill** based on the following rules:

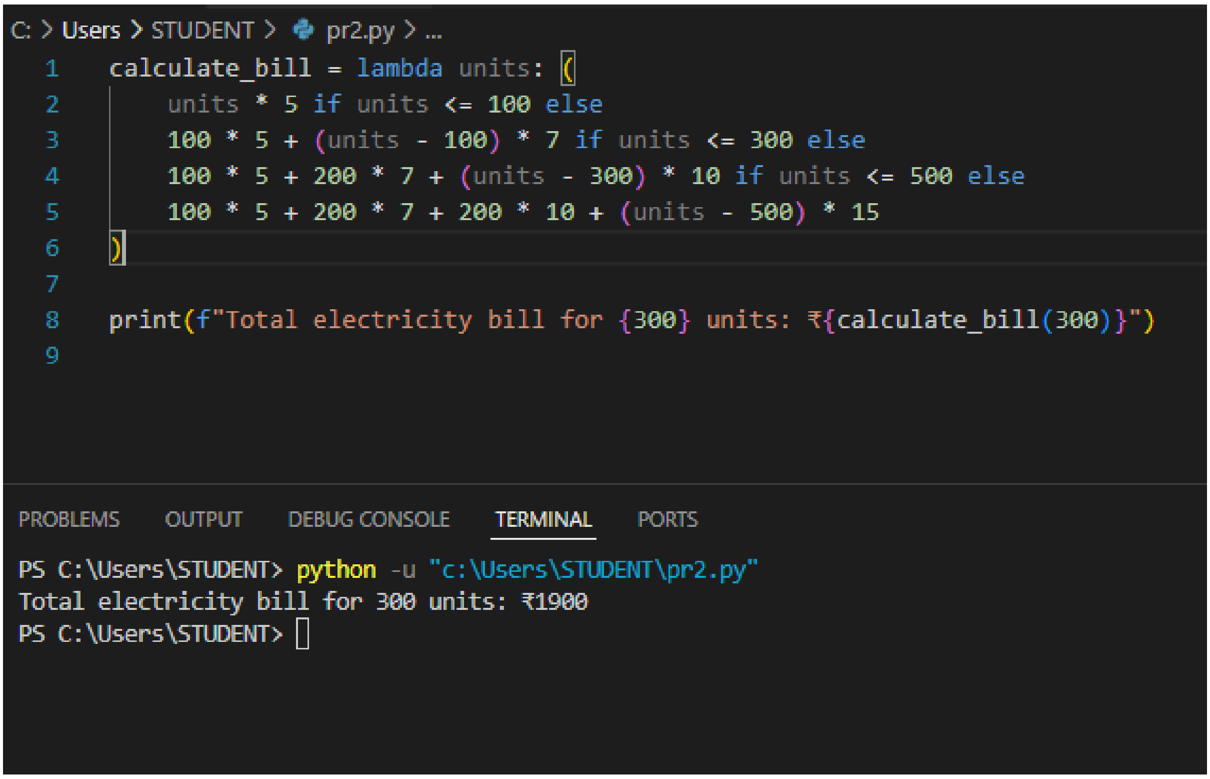
- First 100 units → ₹5 per unit
- Next 200 units (101–300) → ₹7 per unit
- Next 200 units (301–500) → ₹10 per unit
- Above 500 units → ₹15 per unit

Program:

```
calculate_bill = lambda units: (
    units * 5 if units <= 100 else
    100 * 5 + (units - 100) * 7 if units <= 300 else
    100 * 5 + 200 * 7 + (units - 300) * 10 if units <= 500 else
    100 * 5 + 200 * 7 + 200 * 10 + (units - 500) * 15
)

print(f"Total electricity bill for {300} units:
 {calculate_bill(300)}")
```

Output:



```
calculate_bill = lambda units: (
    units * 5 if units <= 100 else
    100 * 5 + (units - 100) * 7 if units <= 300 else
    100 * 5 + 200 * 7 + (units - 300) * 10 if units <= 500 else
    100 * 5 + 200 * 7 + 200 * 10 + (units - 500) * 15
)

print(f"Total electricity bill for {300} units: ₹{calculate_bill(300)}")
```

```
PS C:\Users\STUDENT> python -u "c:\Users\STUDENT\pr2.py"
Total electricity bill for 300 units: ₹1900
PS C:\Users\STUDENT> 
```