1. Write a Python function that takes a list of numbers as input and returns the sum of all even numbers in the list

```
def sum_of_even_numbers(numbers):
  sum_even = 0
  for number in numbers:
    if number % 2 == 0:
      sum_even += number
  return sum_even
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sum_even = sum_of_even_numbers(numbers)
print(sum_even)
```

```
30
```

2. Create a Python function that accepts a string and returns the reverse of that string.

```
def reverse_string(input_string):
  return input_string[::-1]
string_to_reverse = "Kesava Murthy"
reversed_string = reverse_string(string_to_reverse)
reversed_string
```

```
'yhtruM avaseK'
```

3. Implement a Python function that takes a list of integers and returns a new list containing the squares of each number.

```
def squares_of_numbers(numbers):
  squares = []
  for number in numbers:
    squares.append(number * number)
  return squares
numbers = [1, 2, 3, 4, 5]
squares = squares_of_numbers(numbers)
squares
```

```
[1, 4, 9, 16, 25]
```

4. Write a Python function that checks if a given number is prime or not from 1 to 200.

```
def is_prime(number):
  if number <= 1:
    return False
```

```
    for i in range(2, int(number ** 0.5) + 1):
      if number % i == 0:
        return False
    return True
for number in range(1, 201):
    if is_prime(number):
      print(number, "is a prime number")
```

```
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
41 is a prime number
43 is a prime number
47 is a prime number
53 is a prime number
59 is a prime number
61 is a prime number
67 is a prime number
71 is a prime number
73 is a prime number
79 is a prime number
83 is a prime number
89 is a prime number
97 is a prime number
101 is a prime number
103 is a prime number
107 is a prime number
109 is a prime number
113 is a prime number
127 is a prime number
131 is a prime number
137 is a prime number
139 is a prime number
149 is a prime number
151 is a prime number
157 is a prime number
163 is a prime number
167 is a prime number
173 is a prime number
179 is a prime number
181 is a prime number
191 is a prime number
193 is a prime number
197 is a prime number
199 is a prime number
```

5. Create an iterator class in Python that generates the Fibonacci sequence up to a specified number of terms.

```
class FibonacciIterator:
    def __init__(self, num_terms):
        self.num_terms = num_terms
        self.a = 0
        self.b = 1
        self.count = 0
    def __iter__(self):
        return self
    def __next__(self):
        if self.count < self.num_terms:
            fib_number = self.a
            self.a, self.b = self.b, self.a + self.b
            self.count += 1
            return fib_number
        else:
            raise StopIteration
num_terms = 10
fib_iterator = FibonacciIterator(num_terms)
for number in fib_iterator:
    print(number)
```

```
0
1
1
2
3
5
8
13
21
34
```

6. Write a generator function in Python that yields the powers of 2 up to a given exponent.

```
def powers_of_two(exponent):
  power = 0
  while power <= exponent:
    yield 2 ** power
    power += 1
for power in powers_of_two(5):
  print(power)
```

```
1
2
4
8
```

```
16
32
```

7. Implement a generator function that reads a file line by line and yields each line as a string.

```
def read_file_line_by_line(file_path):
  with open(file_path, 'r') as file:
    for line in file:
      yield line
```

8. Use a lambda function in Python to sort a list of tuples based on the second element of each tuple.

```
list_of_tuples = [(1, 5), (3, 2), (2, 8), (4, 1)]
sorted_list = sorted(list_of_tuples, key=lambda x: x[1])
sorted_list
```

```
[(4, 1), (3, 2), (1, 5), (2, 8)]
```

9. Write a Python program that uses `map()` to convert a list of temperatures from Celsius to Fahrenheit.

```
def celsius_to_fahrenheit(celsius):
  return (celsius * 9/5) + 32
celsius_temperatures = [0, 10, 20, 30, 40]
fahrenheit_temperatures = list(map(celsius_to_fahrenheit, celsius_temperatures))
fahrenheit_temperatures
```

```
[32.0, 50.0, 68.0, 86.0, 104.0]
```

10. Create a Python program that uses `filter()` to remove all the vowels from a given string.

```
def remove_vowels(string):
  vowels = "aeiouAEIOU"
  return "".join(filter(lambda char: char not in vowels, string))
string = "Kesava Murthy"
string_without_vowels = remove_vowels(string)
string_without_vowels
```

```
'Ksv Mrthy'
```

11) Imagine an accounting routine used in a book shop. It works on a list with sublists, which look like this:

Order Number	Book Title and Author	Quantity	Price per Item
34587	Learning Python, Mark Lutz	4	40.95
98762	Programming Python, Mark Lutz	5	56.80
77226	Head First Python, Paul Barry	3	32.95
88112	Einführung in Python3, Bernd Klein	3	24.99

```
orders = [
    [34587, "Learning Python, Mark Lutz", 4, 40.95],
    [98762, "Programming Python, Mark Lutz", 5, 56.80],
    [77226, "Head First Python, Paul Barry", 3, 32.95],
    [88112, "Einführung in Python3, Bernd Klein", 3, 24.99]
]
total_costs = [(order[0], order[2] * order[3]) for order in orders]
for order_number, total in total_costs:
    print(f"Order Number: {order_number}, Total Cost: ${total:.2f}")
```

```
Order Number: 34587, Total Cost: $163.80
Order Number: 98762, Total Cost: $284.00
Order Number: 77226, Total Cost: $98.85
Order Number: 88112, Total Cost: $74.97
```

12.Write a Python program, which returns a list with 2-tuples. Each tuple consists of the order number and the product of the price per item and the quantity. The product should be increased by 10,- € if the value of the order is smaller than 100,00 €.

```
orders = [
    [34587, "Learning Python, Mark Lutz", 4, 40.95],
    [98762, "Programming Python, Mark Lutz", 5, 56.80],
    [77226, "Head First Python, Paul Barry", 3, 32.95],
    [88112, "Einführung in Python3, Bernd Klein", 3, 24.99]
]
order_totals = []
for order in orders:
    order_number = order[0]
    quantity = order[2]
    price_per_item = order[3]
    total_cost = quantity * price_per_item
    if total_cost < 100:
        total_cost += 10
    order_totals.append((order_number, total_cost))
print(order_totals)
```

```
[(34587, 163.8), (98762, 284.0), (77226, 108.85000000000001), (88112, 84.97)]
```

13.Write a Python program using lambda and map

```
orders = [
    [34587, "Learning Python, Mark Lutz", 4, 40.95],
    [98762, "Programming Python, Mark Lutz", 5, 56.80],
    [77226, "Head First Python, Paul Barry", 3, 32.95],
    [88112, "Einführung in Python3, Bernd Klein", 3, 24.99]
]
order_totals = list(map(lambda order: (order[0], order[2] * order[3] + (10 if order
print(order_totals)
```

```
[(34587, 163.8), (98762, 284.0), (77226, 108.85000000000001), (88112, 84.97)]
```