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

Ans-Here is a Python function that does what you described:
```

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

Ans-Here is a Python function that does what you described:

```
def sum_even_numbers(numbers):
    """
    Returns the sum of all even numbers in the input list.
    """
    even_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
    return even_sum
```

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

Ans:-Here is a Python function that does what you described:

```
def reverse_string(input_str):
    """
    Returns the reverse of the input string.
    """
    reversed_str = input_str[::-1]
    return reversed_str
```

You can test this function with a sample string, like this:

```
input_str = "Hello, World!"
result = reverse_string(input_str)
print(result)  # Output: "!dlroW ,olleH"
```

```
!dlroW ,olleH
```

```
def reverse_string(input_str):
    """
    Returns the reverse of the input string.
    """
    reversed_str = "".join(reversed(input_str))
    return reversed_str
```

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

Ans:-

```
def square_numbers(numbers):
    """
    Returns a new list containing the squares of each number in the input list.
    """
    squared_numbers = [num ** 2 for num in numbers]
    return squared_numbers
```

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

Ans:-

```
def is_prime(n):
    """
    Returns True if the input number is prime, False otherwise.
    """
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

def check_primes_up_to_200():
    """
    Prints out all prime numbers from 1 to 200.
```

```
    """
    for num in range(1, 201):
        if is_prime(num):
            print(num)
```

Here's an explanation of how the functions work:

The is_prime function takes an integer n as input and returns True if it's prime, and False otherwise.

We first handle the special case where n is less than or equal to 1, which are not prime numbers.

We then use a for loop to iterate from 2 to the square root of n (inclusive), and check if n is divisible by any of these numbers. If it is, we return False. If the loop completes without finding a divisor, we return True, indicating that n is prime.

The check_primes_up_to_200 function uses the is_prime function to check all numbers from 1 to 200, and prints out the prime numbers.

You can test the check_primes_up_to_200 function by calling it:

```
check_primes_up_to_200()
```

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

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

Ans:-Here is a Python iterator class that generates the Fibonacci sequence up to a specified number of terms:

Double-click (or enter) to edit

```
class FibonacciIterator:
    """
    An iterator class that generates the Fibonacci sequence up to a specified number of terms.
    """
    def __init__(self, num_terms):
        """
        Initializes the iterator with the number of terms to generate.
        """
        self.num_terms = num_terms
        self.current_term = 0
```

```
        self.a, self.b = 0, 1

    def __iter__(self):
        """
        Returns the iterator object itself.
        """
        return self

    def __next__(self):
        """
        Returns the next term in the Fibonacci sequence.
        """
        if self.current_term < self.num_terms:
            result = self.a
            self.a, self.b = self.b, self.a + self.b
            self.current_term += 1
            return result
        else:
            raise StopIteration

# Example usage:
iterator = FibonacciIterator(10)
for term in iterator:
    print(term)
```

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

Ans:-Here is a Python generator function that yields the powers of 2 up to a given exponent:

```
def powers_of_two(exponent):
    """
    Yields the powers of 2 up to the given exponent.
    """
    for i in range(exponent + 1):
        yield 2 ** i

# Example usage:
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.

Ans:- Here is a Python generator function that reads a file line by line and yields each line as a string:

```
def read_file_line_by_line(file_path):
    """
    Yields each line of the file as a string.
    """
    with open(file_path, 'r') as file:
        for line in file:
            yield line.strip()

# Example usage:
file_path = 'example.txt'
```

```
for line in read_file_line_by_line(file_path):
    print(line)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-11-92c6bcf53826> in <cell line: 11>()
      9 # Example usage:
     10 file_path = 'example.txt'
---> 11 for line in read_file_line_by_line(file_path):
     12     print(line)

<ipython-input-11-92c6bcf53826> in read_file_line_by_line(file_path)
      3     Yields each line of the file as a string.
      4     """
----> 5     with open(file_path, 'r') as file:
      6         for line in file:
      7             yield line.strip()

FileNotFoundError: [Errno 2] No such file or directory: 'example.txt'
```

Next steps: Explain error

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

Ans:-Here is an example of how to use a lambda function to sort a list of tuples based on the second element of each tuple:

```
tuples = [(1, 3), (2, 1), (3, 2), (4, 4)]

sorted_tuples = sorted(tuples, key=lambda x: x[1])

print(sorted_tuples)  # Output: [(2, 1), (3, 2), (1, 3), (4, 4)]
```

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

The lambda function is equivalent to a regular function defined as:

```
def sort_key(tup):
    return tup[1]
```

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

Ans:-Here is a Python program that uses map() to convert a list of temperatures from Celsius to Fahrenheit:

```
def celsius_to_fahrenheit(celsius):
    """
    Converts a temperature from Celsius to Fahrenheit.
    """
    return (celsius * 9/5) + 32

temperatures_celsius = [0, 10, 20, 30, 40]

temperatures_fahrenheit = list(map(celsius_to_fahrenheit, temperatures_celsius))

print(temperatures_fahrenheit)  # Output: [32.0, 50.0, 68.0, 86.0, 104.0]
```

```
[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.

Ans:-Here is a Python program that uses filter() to remove all the vowels from a given string:

```
def remove_vowels(char):
    """
    Returns True if the character is not a vowel, False otherwise.
    """
    vowels = 'aeiouAEIOU'
    return char not in vowels

input_string = "Hello, World!"

result = ''.join(filter(remove_vowels, input_string))

print(result)  # Output: "Hll, Wrld!"
```

```
Hll, Wrld!
```

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

Order Number

34587

98762

77226

88112

Book Title and Author

Learning Python, Mark Lutz

Programming Python, Mark Lutz

Head First Python, Paul Barry

Einführung in Python3, Bernd Klein

Quantity

4

5

3

3

Price per Item

40.95

56.80

32.95

24.99

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 €.

Ans:-Here is a Python program that solves the problem:

```
# Define the data
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]
]

# Define a function to calculate the total cost
def calculate_total_cost(order):
    quantity = order[2]
    price = order[3]
    total_cost = quantity * price
    if total_cost < 100:
        total_cost += 10
    return (order[0], total_cost)

# Use a list comprehension to apply the function to each order
result = [calculate_total_cost(order) for order in orders]

# Print the result
for order in result:
    print(order)
```

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

12.Write a Python program using lambda and map.

Ans:-Here is a Python program that uses lambda and map to square all the numbers in a list:

```
# Define a list of numbers
numbers = [1, 2, 3, 4, 5]

# Use lambda and map to square all the numbers
squared_numbers = list(map(lambda x: x ** 2, numbers))
```

```
# Print the result
print(squared_numbers)
```

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

You can also use lambda and map to perform more complex operations, such as filtering a list of numbers to only include the even numbers:

```
numbers = [1, 2, 3, 4, 5]
even_numbers = list(map(lambda x: x, filter(lambda x: x % 2 == 0, numbers)))
print(even_numbers)
```

```
[2, 4]
```