PYTHON LAB FILE

SUBMITTED BY:

NAME: PRIYANSHI RAI

SAP ID: 500108262

BATCH:32(B. TECH CSE)

Practical No 1: Python Installation and starting with python

Q1)

```
# installed python
```

Q2)

```
a) print("Hello Everyone !!!")
      b) print("Hello \n World")
      c) print("Hello \n \t World")
      d) print("Rohit's date of birth is 12\\05\\1999")
```

Q3)

```
 x = "Hello"
      print(x)
```

Q4)

```
num = 4244
pii=3.14
strng = "Hello World!"
boolean = True
print(num)
print(pii)
print(strng)
print(boolean)
```

Q5)

```
a = "Priyanshi"
b = "Rai"
name = a + " " + b
print(name)
```

Q6)

```
first_name = "Priyanshi"
last_name = "Rai"
nickname = "Priii"
print(first_name, "(", nickname, ")", last_name)
```

Q7)

```
name = "NIKUNJ BANSAL"
sap_id = "500069944"
dob = "13 Oct 1999"
address = "UPES \n \t \t Bidholi Campus"
pincode = "248007"
programme = "AI & ML"
semester = "2"
print("NAME :", name)
print("SAP ID :", sap_id)
print("DATE OF BIRTH :", dob)
print("ADDRESS :", address)
print("\t \t Pincode :", pincode)
print("Programme :", programme)
print("Semester :", semester)
```

Practical No 2:Use of input statements, operators

Q1)

```
x = 9
y = 7
addition = x + y
multiplication = x * y
division = x / y
subtraction = x - y
print("Addition: ", addition)
print("Multiplication: ", multiplication)
print("Division: ", division)
print("Subtraction: ", subtraction)
```

Q2)

```
import math
radius = float(input("Enter the radius of the circle: "))
area = math.pi * (radius**2)
print("Area of the circle is: ", area)
```

Q3)

```
x = 4
y = 3
result = (x+y) ** 2
print(result)
```

Q4)

```
import math
a = float(input("Enter the length of side a: "))
b = float(input("Enter the length of side b: "))
c = math.sqrt(a**2 + b**2)
print("Length of the hypotenuse is: ", c)
```

Q5)

```
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time in years: "))
simple_interest = (principal * rate * time) / 100
print("Simple interest is: ", simple_interest)
```

Q6)

```
import math
a = float(input("Enter the length of side a: "))
b = float(input("Enter the length of side b: "))
c = float(input("Enter the length of side c: "))
s = (a + b + c) / 2
area = math.sqrt(s*(s-a)*(s-b)*(s-c))
print("Area of the triangle is: ", area)
```

Q7)

```
seconds = int(input("Enter the number of seconds: "))
hours = seconds // 3600
seconds = seconds % 3600
minutes = seconds // 60
seconds = seconds % 60
print("Time is: ", hours, " hours, ", minutes, " minutes, and ", seconds, " seconds.")
```

Q8)

```
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
a = a + b
b = a - b
a = a - b
print("After swapping, a = ", a, " and b = ", b)
```

Q9)

```
n = int(input("Enter the value of n: "))
sum = 0
for i in range(1, n+1):
    sum += i
print("Sum of first ", n, " natural numbers is: ", sum)
```

Q10)

```
# Bitwise AND
print("Bitwise AND Truth Table:")
print("a\tb\ta & b")
print("--------------")
for a in [0, 1]:
    for b in [0, 1]:
        print(a, "\t", b, "\t", a & b)

# Bitwise OR
print("\n Bitwise OR Truth Table:")
print("a\tb\ta | b")
print("--------------")
for a in [0, 1]:
    for b in [0, 1]:
        print(a, "\t", b, "\t", a | b)

# Bitwise XOR
print("\n Bitwise XOR Truth Table:")
print("a\tb\ta ^ b")
print("--------------")
for a in [0, 1]:
    for b in [0, 1]:
        print(a, "\t", b, "\t", a ^ b)
```

Q11)

```
num = int(input("Enter a number: "))
n = int(input("Enter the number of bits to shift: "))
left_shift = num << n
right_shift = num >> n
```

```
print("Left shift value is: ", left_shift)
print("Right shift value is: ", right_shift)
```

Q12)

```
seq = [10, 20, 56, 78, 89]
num = int(input("Enter a number to search: "))
if num in seq:
    print(num, " is present in the sequence.")
else:
    print(num, " is not present in the sequence.")
```

Q13)

```
string = input("Enter a string: ")
char = input("Enter a character to search: ")
if char in string:
    print(char, " is present in the string.")
else:
    print(char, " is not present in the string.")
```

Practical No 3:Conditional Statements

Q1)
```
num = int(input("Enter a number: "))
if num % 3 == 0 and num % 5 == 0:
    print(num, " is divisible by 3 and 5 both.")
else:
    print(num, " is not divisible by 3 and 5 both.")
```

Q2)
```
num = int(input("Enter a number: "))
if num % 5 == 0:
    print(num, " is a multiple of 5.")
else:
    print(num, " is not a multiple of 5.")
```

Q3)
```
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
if a > b:
    print(a, " is the greatest number.")
elif b > a:
    print(b, " is the greatest number.")
else:
    print("Numbers are equal.")
```

Q4)
```
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
c = int(input("Enter the third number: "))
if a > b and a > c:
    print(a, " is the greatest number.")
elif b > a and b > c:
    print(b, " is the greatest number.")
elif c > a and c > b:
    print(c, " is the greatest number.")
else:
    print("No two values are the same.")
```

Q5)
```
import math
a = float(input("Enter the coefficient of x^2: "))
b = float(input("Enter the coefficient of x: "))
c = float(input("Enter the constant term: "))
discriminant = b ** 2 - 4 * a * c
if discriminant > 0:
    root1 = (-b + math.sqrt(discriminant)) / (2 * a)
    root2 = (-b - math.sqrt(discriminant)) / (2 * a)
    print("The roots are real and distinct.")
    print("Root 1 is: ", root1)
    print("Root 2 is: ", root2)
```

```
elif discriminant == 0:
    root = -b / (2 * a)
    print("The roots are real and equal.")
    print("Root is: ", root)
else:
    real_part = -b / (2 * a)
    imaginary_part = math.sqrt(-discriminant) / (2 * a)
    print("The roots are imaginary.")
    print("Root 1 is: ", real_part, "+", imaginary_part, "i")
    print("Root 2 is: ", real_part, "-", imaginary_part, "i")
```

Q6)

```
year = int(input("Enter a year: "))
if year % 4 == 0 and year % 100 != 0 or year % 400 == 0:
    print(year, " is a leap year.")
else:
    print(year, " is not a leap year.")
```

Q7)

```
day = int(input("Enter the day: "))
month = int(input("Enter the month: "))
year = int(input("Enter the year: "))
if year % 4 == 0 and year % 100 != 0 or year % 400 == 0:
    leap_year = True
else:
    leap_year = False
if month == 2:
    if leap_year:
        max_days = 29
    else:
        max_days = 28
```

Q8)

```
name = input("Enter the name of the student: ")
roll_number = input("Enter the roll number of the student: ")
sapid = input("Enter the SAP ID of the student: ")
semester = input("Enter the semester of the student: ")
course = input("Enter the course of the student: ")
subject_marks = {}

for i in range(5):
    subject_name = input(f"Enter the name of subject {i+1}: ")
    subject_marks[subject_name] = int(input(f"Enter the marks of subject {i+1}: "))

total_marks = sum(subject_marks.values())
percentage = (total_marks/500)*100
cgpa = percentage/10
if cgpa >= 9.1:
    grade = "O (Outstanding)"
elif cgpa >= 8.1:
    grade = "A+"
elif cgpa >= 7.1:
    grade = "A"
elif cgpa >= 6.1:
    grade = "B+"
elif cgpa >= 5.1:
    grade = "B"
elif cgpa >= 3.5:
    grade = "C+"
else:
    grade = "F"

print("\n\nGrade Sheet")
print(f"Name: {name}")
print(f"Roll Number: {roll_number}")
print(f"SAP ID: {sapid}")
print(f"Semester: {semester}")
print(f"Course: {course}")
print("Subject Name: Marks")
for subject, marks in subject_marks.items():
    print(f"{subject}: {marks}")
print(f"Percentage: {percentage:.2f}%")
print(f"CGPA: {cgpa:.1f}")
print(f"Grade: {grade}")
```

Practical No 4: Loops

Q1)

```
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1, num + 1):
       factorial = factorial * i
   print("The factorial of", num, "is", factorial)
```

Q2)

```
num = int(input("Enter a number: "))
order = len(str(num))
sum = 0
temp = num
while temp > 0:
   digit = temp % 10
   sum += digit ** order
   temp //= 10
if num == sum:
   print(num,"is an Armstrong number")
else:
   print(num,"is not an Armstrong number")
```

Q3)

```
n = int(input("Enter the number of terms: "))
n1, n2 = 0, 1
count = 0
if n <= 0:
   print("Please enter a positive integer")
elif n == 1:
   print("Fibonacci sequence upto", n, ":")
   print(n1)
else:
   print("Fibonacci sequence:")
   while count < n:
       print(n1)
       nth = n1 + n2
       n1 = n2
       n2 = nth
       count += 1
```

Q4)

```
num = int(input("Enter a number: "))
if num > 1:
   for i in range(2, num):
       if (num % i) == 0:
           print(num, "is not a prime number")
```

```
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

Q5)

```
num = int(input("Enter a number: "))
temp = num
reverse = 0
while num > 0:
    digit = num % 10
    reverse = reverse * 10 + digit
    num //= 10
if temp == reverse:
    print(temp, "is a palindrome number")
else:
    print(temp, "is not a palindrome number")
```

Q6)

```
num = int(input("Enter a number: "))
sum = 0

while num > 0:
    digit = num % 10
    sum += digit
    num //= 10
print("The sum of digits is", sum)
```

Q7)

```
count = 0
for i in range(1, 101):
    if i % 5 == 0 or i % 7 == 0:
        print(i)
        count += 1
print("Total count:", count)
```

Q8)

```
string = input("Enter a string: ")
uppercase = ""
for char in string:
    if char.islower():
        uppercase += char.upper()
    else:
        uppercase += char
print("Original string:", string)
print("Uppercase string:", uppercase)
```

Q9)

```
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num/2) + 1):
        if num % i == 0:
            return False
    return True
for i in range(1, 101):
    if is_prime(i):
        print(i)
```

Q10)

```
num = int(input("Enter a number: "))
for i in range(1, 11):
    print(f"{num} * {i} = {num * i}")
```

Practical No 5:String and Sets

Q1)
```
string = input("Enter a string: ")
count = 0
for char in string:
    if char.isupper():
        count += 1
print("Number of capital letters:", count)
```

Q2)
```
string = input("Enter a string: ")
count = 0
vowels = "aeiouAEIOU"
for char in string:
    if char in vowels:
        count += 1
print("Total number of vowels:", count)
```

Q3)
```
sentence = input("Enter a sentence: ")
words = sentence.split()
for word in words:
    print(word)
```

Q4)
```
string = input("Enter a string: ")
substring = input("Enter a substring: ")
count = 0
for i in range(len(string)):
    if string[i:i+len(substring)] == substring:
        count += 1
print("Number of times substring occurs:", count)
```

Q5)
```
string = input("Enter a string: ")
alphabets = "abcdefghijklmnopqrstuvwxyz"
counts = {}
for char in alphabets:
    counts[char] = string.lower().count(char)
print("Occurrences of each alphabet:", counts)
```

Q6)
```
sentence = input("Enter a sentence: ")
words = sentence.split()
unique_words = set(words)
print("Number of unique words:", len(unique_words))
```

Q7)
```
n = int(input("Enter the number of fruits in each set: "))
s1 = set()
```

```
s2 = set()
print("Enter fruits in set 1:")
for i in range(n):
    s1.add(input())
print("Enter fruits in set 2:")
for i in range(n):
    s2.add(input())
common_fruits = s1.intersection(s2)
print("Common fruits:", common_fruits) # part(a)

unique_fruits = s1.difference(s2)
print("Fruits only in set 1:", unique_fruits) #part(b)

all_fruits = s1.union(s2)
print("Count of all fruits:", len(all_fruits))  #part(c)
```

Q8)

```
S1 = {"Red", "yellow", "orange", "blue"}
S2 = {"violet", "blue", "purple"}
print("Intersection of S1 and S2:", S1.intersection(S2))
print("Union of S1 and S2:", S1.union(S2))
print("Difference of S1 and S2:", S1.difference(S2))
print("Difference of S2 and S1:", S2.difference(S1))
print("Symmetric difference of S1 and S2:", S1.symmetric_difference
```

Practical No 6:Lists, tuples, dictionary

Q1)

```
n = int(input("Enter the number of values: "))
values = []
for i in range(n):
    value = int(input("Enter a value (between 0-3): "))
    if value < 0 or value > 3:
        print("Invalid value entered, please try again.")
        continue
    values.append(value)
count = [0] * 4
for value in values:
    count[value] += 1
for i in range(4):
    print(f"{i}: {count[i]}")
```

Q2)

```
n = int(input("Enter the number of values: "))
values = []
for i in range(n):
    value = int(input(f"Enter value {i+1}: "))
    values.append(value)
avg = sum(values) / n
print("Average:", avg)
```

Q3)

```
n = int(input("Enter the number of students: "))
scores = []
for i in range(n):
    score = int(input(f"Enter score for student {i+1}: "))
    scores.append(score)

scores = list(set(scores))
scores.sort(reverse=True)
print("Runner-up score:", scores[1])
```

Q4)

```
n = int(input("Enter the number of persons: "))
persons = {}
for i in range(n):
    name = input(f"Enter name of person {i+1}: ")
    city = input(f"Enter city of person {i+1}: ")
    persons[name] = city
print("Names:", list(persons.keys()))
print("Cities:", list(set(persons.values())))
print("Person details:")
for name, city in persons.items():
    print(f"{name}: {city}")
city_count = {}
for city in persons.values():
    city_count[city] = city_count.get(city, 0) + 1
```

```
print("City counts:", city_count)
```

Q5)

```
n = int(input("Enter the number of movies: "))
movies = []
for i in range(n):
    movie = {}
    movie["name"] = input("Enter name of movie: ")
    movie["year"] = int(input("Enter year of release: "))
    movie["director"] = input("Enter name of director: ")
    movie["production_cost"] = int(input("Enter production cost: "))
    movie["collection"] = int(input("Enter collection made: "))
    movies.append(movie)

# a) print all movie details
print("All movie details:")
for movie in movies:
        print(movie)

# b) display name of movies released before 2015
print("\nMovies released before 2015:")
for movie in movies:
    if movie["year"] < 2015:
        print(movie["name"])

# c) print movies that made a profit
print("\nMovies that made a profit:")
for movie in movies:
    if movie["collection"] > movie["production_cost"]:
        print(movie["name"])

# d) print movies directed by a particular director
director_name = input("\nEnter name of director to search: ")
print("Movies directed by", director_name, ":")
for movie in movies:
    if movie["director"] == director_name:
        print(movie["name"])
```

Practical No 7:Functions

Q1)
```
def find_max_min(numbers):
    max_num = numbers[0]
    min_num = numbers[0]
    for num in numbers:
        if num > max_num:
            max_num = num
        elif num < min_num:
            min_num = num
    return max_num, min_num
```

Q2)
```
def sum_of_cubes(n):
    if n == 0:
        return 0
    else:
        return (n-1)**3 + sum_of_cubes(n-1)
```

Q3)
```
def print_numbers(n):
    if n == 1:
        print(1)
    else:
        print_numbers(n-1)
        print(n)
```

Q4)
```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)
n = 10
for i in range(n):
    print(fibonacci(i))
```

Q5)
```
volume_of_cone = lambda r, h: (1/3) * 3.14 * r**2 * h
```

Q6)
```
max_min_tuple = lambda lst: (max(lst), min(lst))
```

Q7)
```
# Keyword argument example
def greet(name, greeting):
    print(f"{greeting}, {name}!")
greet(name="Alice", greeting="Hello")
greet(greeting="Hi", name="Bob")

# Default argument example
```

```
def repeat(text, times=2):
   print(text * times)
repeat("hello")
repeat("world", 3)

# Variable length argument example
def sum_all(*numbers):
   result = 0
   for number in numbers:
      result += number
   return result
print(sum_all(1, 2, 3))
print(sum_all(4, 5, 6, 7, 8))
```

In the above program, we define three functions:

1. “greet” - This function takes two arguments name and greeting, and prints a greeting message using those arguments. We call this function using <u>keyword arguments</u>, which allows us to pass the arguments in any order we want, as long as we specify the argument names.
2. “repeat” - This function takes one mandatory argument text, and one optional argument times, which defaults to 2 if not specified. The function prints the text argument repeated times number of times. We call this function with and without specifying the times argument, to show how <u>default arguments</u> work.
3. “sum_all” - This function takes a variable number of arguments, using the * operator before the argument name. This means that we can pass any number of arguments to the function, and they will be collected into a tuple called numbers. The function then loops over this tuple and sums all the numbers. We call this function with different numbers of arguments to show how <u>variable length arguments</u> work.

Practical No 8:File handling and Exception Handling

Q1) (A)
```
with open("name.txt", "r") as file:
    names = file.readlines()
    count = len(names)
    print("Total names:", count)
```

(B)
```
with open("name.txt", "r") as file:
    names = file.readlines()
    count = 0
    vowels = ['a', 'e', 'i', 'o', 'u']
    for name in names:
        if name[0].lower() in vowels:
            count += 1
    print("Names starting with vowel:", count)
```

(C)
```
with open("name.txt", "r") as file:
    names = file.readlines()
    longest_name = max(names, key=len).strip()
    print("Longest name:", longest_name)
```

Q2)
```
numbers = [10, 20, 30, 40, 50]
with open("numbers.txt", "w") as file:
    for num in numbers:
        file.write(str(num) + "\n")
with open("numbers.txt", "r") as file:
    numbers = file.readlines()
    numbers = [int(num.strip()) for num in numbers]
max_num = max(numbers)
print("Max number:", max_num)
average = sum(numbers) / len(numbers)
print("Average of numbers:", average)
count = len([num for num in numbers if num > 100])
print("Numbers greater than 100:", count)
```

Q3)
(A)
```
with open("city.txt", "r") as file:
    cities = file.readlines()
    print("City details:")
    for city in cities:
        city_data = city.split()
        name, population, area = city_data[0], float(city_data[1]), float(city_data[2])
        print(name, population, area)
```

(B)
```
with open("city.txt", "r") as file:
    cities = file.readlines()
    print("Cities with population more than 10 Lakhs:")
    for city in cities:
        city_data = city.split()
        name, population, area = city_data[0], float(city_data[1]), float(city_data[2])
        if population > 10:
            print(name)
```

(C)

```
with open("city.txt", "r") as file:
    cities = file.readlines()
    area_sum = sum([float(city.split()[2]) for city in cities])
    print("Sum of areas of all cities:", area_sum)
```

Q4)

```
n = int(input())
for i in range(n):
    try:
        a, b = input().split()
        print(int(a) // int(b))
    except ValueError:
        print("Error Code: invalid literal for int() with base 10:", a)
    except ZeroDivisionError:
        print("Error Code: integer division or modulo by zero")
```

Practical No 9: Classes and object

Q1)

```
class Student:
    def __init__(self, name, sap_id, phy_marks, chem_marks, math_marks):
        self.name = name
        self.sap_id = sap_id
        self.phy_marks = phy_marks
        self.chem_marks = chem_marks
        self.math_marks = math_marks

    def display_details(self):
        print("Name:", self.name)
        print("SAP ID:", self.sap_id)
        print("Marks in Physics:", self.phy_marks)
        print("Marks in Chemistry:", self.chem_marks)
        print("Marks in Maths:", self.math_marks)
        print("\n")
for i in range(3):
    name = input("Enter Name: ")
    sap_id = input("Enter SAP ID: ")
    phy_marks = float(input("Enter Marks in Physics: "))
    chem_marks = float(input("Enter Marks in Chemistry: "))
    math_marks = float(input("Enter Marks in Maths: "))
    student = Student(name, sap_id, phy_marks, chem_marks, math_marks)
    student.display_details()
```

Q2)

```
class Student:
    def __init__(self, name, sap_id, phy_marks, chem_marks, maths_marks):
        self.name = name
        self.sap_id = sap_id
        self.phy_marks = phy_marks
        self.chem_marks = chem_marks
        self.maths_marks = maths_marks

    def display(self):
        print("Name:", self.name)
        print("SAP ID:", self.sap_id)
        print("Physics Marks:", self.phy_marks)
        print("Chemistry Marks:", self.chem_marks)
        print("Maths Marks:", self.maths_marks)

    def marks_percentage(self):
        total_marks = self.phy_marks + self.chem_marks + self.maths_marks
        percentage = (total_marks / 300) * 100
        return percentage

    def result(self):
        if self.phy_marks >= 40 and self.chem_marks >= 40 and self.maths_marks >= 40:
            print("Result: Pass")
        else:
            print("Result: Fail")

def class_average(students):
    total_percentage = 0
    for student in students:
        total_percentage += student.marks_percentage()
    return total_percentage / len(students)
```

Q3)

```
#single
class A:
    def method(self):
        print("Method of A called")
class B(A):
    pass
b = B()
b.method()
#multiple
class A:
    def method(self):
        print("Method of A called")
class B:
    def method(self):
        print("Method of B called")
class C(A, B):
    pass
c = C()
c.method()
#multi level
class A:
    def method(self):
        print("Method of A called")
class B(A):
    def method2(self):
        print("Method of B called")
class C(B):
    pass
b=B()
b.method()
c = C()
c.method2()
#hierarchical
class A:
    def method(self):
        print("Method of A called")
class B(A):
    pass
class C(A):
    pass
b=B()
b.method()
c = C()
c.method()
```

Q4)

```
class A:
    def method(self):
        print("Method of A called")
class B(A):
    def method(self):
        print("Method of B called")
b=B()
b.method()
```

Q5)

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)
    def __str__(self):
        return "Point(x={}, y={})".format(self.x, self.y)
p1 = Point(10, 20)
p2 = Point(12, 15)
p3 = p1 + p2
print(p3)
```

Practical No 10: Data Analysis and Visualization

Q1)

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
sum_arr = np.sum(arr)
print(sum_arr)
```

Q2)

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
sum_rows = np.sum(arr, axis=1)
sum_cols = np.sum(arr, axis=0)
second_max=second_max = np.amax(arr[arr != max_element])
print("Array:\n", arr)
print("Sum of all rows:", sum_rows)
print("Sum of all columns:", sum_cols)
print("Second maximum element in the array:", second_max)
```

Q3)

```
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
print(np.matmul(A, B)
```

Q4)

```
import pandas as pd
data = {'X': [78, 85, 96, 80, 86],
        'Y': [84, 94, 89, 83, 86],
        'Z': [86, 97, 96, 72, 83]}
df = pd.DataFrame(data)
print(df)
```

Q5)

```
import pandas as pd
import numpy as np
exam_data = {'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],'name': ['Anastasia', 'Dima', 'Katherine',
'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],'qualify': ['yes', 'no', 'yes', 'no',
'no', 'yes', 'yes', 'no', 'no', 'yes'],
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
}
df = pd.DataFrame(exam_data, index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'])
first_three_rows = df.loc[['a', 'b', 'c']]
print(first_three_rows)
```

Q6)

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'A': [1, 2, np.nan, 4], 'B': [5, np.nan, np.nan, 8], 'C': [9, 10, 11, 12]})
df.fillna(0, inplace=True)
print(df)
```

Q7)

```
import matplotlib.pyplot as plt
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]
plt.plot(x, y)
plt.title("Linear graph", fontsize=25, color="green")
plt.ylabel('Y-Axis')
plt.xlabel('X-Axis')
plt.ylim(0, 80)
plt.xticks(x, labels=["one", "two", "three", "four"])
plt.show()
```