```
from datetime import date    (1b)
name = input("Enter the name of the person: ")
birth_year = int(input("Enter year of birth: "))
current_year = date.today().year
age = current_year - birth_year
if age >= 60:
    print(f"{name} is {age} years old and is a Senior Citizen.")
else:
    print(f"{name} is {age} years old and is NOT a Senior Citizen.")
name = input("Enter the name of the student: ")      (1a)
usn = input("Enter the USN of the student: ")
m1 = int(input("Enter marks in Subject 1: "))
m2 = int(input("Enter marks in Subject 2: "))
m3 = int(input("Enter marks in Subject 3: "))
total = m1 + m2 + m3
percentage = total / 3
print("\n--- Student Details ---")
print(f"Name       : {name}")
print(f"USN        : {usn}")
print(f"Marks      : {m1}, {m2}, {m3}")
print(f"Total      : {total}")
print(f"Percentage : {percentage:.2f}%")
n = int(input("Enter how many Fibonacci numbers to print: "))      ( 2a)
a, b = 0, 1
print(f"\nFibonacci sequence with {n} terms:")
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

```
def factorial(n):                                    ( 2b)
   return 1 if n == 0 or n == 1 else n * factorial(n - 1)
n = int(input("Enter value of n: "))
r = int(input("Enter value of r (0 ≤ r ≤ n): "))
if r < 0 or r > n:
   print("Invalid input! r should be between 0 and n.")
else:
   nCr = factorial(n) // (factorial(r) * factorial(n - r))
   print(f"{n}C{r} = {nCr}")
```

```
from math import sqrt          ( 3)
n = int(input("Enter how many numbers: "))
numbers = []
for i in range(n):
   num = float(input(f"Enter number {i+1}: "))
   numbers.append(num)
mean = sum(numbers) / n
variance = sum((x - mean) ** 2 for x in numbers) / n
std_dev = sqrt(variance)
print(f"\nNumbers: {numbers}")
print(f"Mean          : {mean:.2f}")
print(f"Variance      : {variance:.2f}")
print(f"Standard Deviation: {std_dev:.2f}")
```

```
num_str = input("Enter a multi-digit number: ")                ( 4)
print(f"The number entered is: {num_str}")
print("Digit Frequencies:")
for digit in sorted(set(num_str)):
   print(f"Digit {digit} occurs {num_str.count(digit)} time(s)")
```

```
import os                          (  5 )
import string
filename = input("Enter the filename: ")
if not os.path.isfile(filename):
   print(f"File '{filename}' does not exist.")
   exit()
with open(filename, "r") as file:
   text = file.read()
for ch in string.punctuation:
   text = text.replace(ch, " ")
words = text.lower().split()
word_freq = {}
for word in words:
   if word in word_freq:
      word_freq[word] += 1
   else:
      word_freq[word] = 1
sorted_words = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)
print("\nTop 10 Most Frequent Words:")
print("-" * 35)
for word, count in sorted_words[:10]:
   print(f"{word} : {count} time(s)")
import sys
def DivExp(a,b):
   assert a>0,"a should be greater than 0"
   try:c=a/b
   except ZeroDivisionError:
      print("value of b cannot be zero")
      sys.exit(0)
```

```
  else:
    return c
val1=int(input("enter a value for a:"))
val2=int(input("enter a value for b:"))
val3=DivExp(val1,val2)
print(val1,"/",val2,"=",val3)
```

```
import os                                                    (06)
import sys
# Get the filename from the user
fname = input("Enter the filename whose contents are to be sorted: ")  # sample
file Text.txt
# Check if the file exists
if not os.path.isfile(fname):
  print("File", fname, "doesn't exist")
  sys.exit(0)
# Read the contents of the file
with open(fname, "r") as infile:
  myList = infile.readlines()
# Remove trailing newline characters and create a sorted list
lineList = [line.strip() for line in myList]
lineList.sort()  # Sort the list
# Write sorted contents to a new file sorted.txt
with open("sorted.txt", "w") as outfile:
  for line in lineList:
    outfile.write(line + "\n")
# Check if sorted.txt was created and display its contents
if os.path.isfile("sorted.txt"):
  print("\nFile containing sorted content 'sorted.txt' created successfully.")
  print("sorted.txt contains", len(lineList), "lines.")
  print("Contents of sorted.txt:")
  print("==============================================================")
  with open("sorted.txt", "r") as rdFile:
    for line in rdFile:
      print(line, end="")
```

```
import os                                                    (07)
import sys
import pathlib
import zipfile
dirName = input("Enter Directory name that you want to backup: ")
if not os.path.isdir(dirName):
  print("Directory", dirName, "doesn’t exist")
  sys.exit(0)
curDirectory = pathlib.Path(dirName)
with zipfile.ZipFile("myZip.zip", mode="w") as archive:
  for file_path in curDirectory.rglob("*"):
    archive.write(file_path, arcname=file_path.relative_to(curDirectory))
if os.path.isfile("myZip.zip"):
  print("Archive", "myZip.zip", "created successfully")
else:
  print("Error in creating zip archive")
```

```
import sys                                                              (8)
def DivExp(a, b):
  assert a > 0, "a should be greater than 0"
  try:
    c = a / b
  except ZeroDivisionError:
    print("Value of b cannot be zero")
    sys.exit(0)
    else:
        return c
val1 = int(input("Enter a value for a: "))
val2 = int(input("Enter a value for b: "))
val3 = DivExp(val1, val2)
print(val1, "/", val2, "=", val3)
```

---

(9)

```
class Complex:
    def __init__(self, real, imag):
        self.real = real
        self.imag = imag
    def __add__(self, other):
        real_sum = self.real + other.real
        imag_sum = self.imag + other.imag
        return Complex(real_sum, imag_sum)
    def __str__(self):
        return f"{self.real} + {self.imag}i"
# Input
a = int(input("Enter real part of first complex number: "))
b = int(input("Enter imaginary part of first complex number: "))
c = int(input("Enter real part of second complex number: "))
d = int(input("Enter imaginary part of second complex number: "))
# Create Complex numbers
c1 = Complex(a, b)
c2 = Complex(c, d)
c3 = c1 + c2
print(f"The sum of {c1} and {c2} is: {c3}")
```

---

(10)

```
class Student:
    def __init__(self, name="", usn="", score=None):
        if score is None:
            score = [0, 0, 0, 0]
        self.name = name
        self.usn = usn
        self.score = score
    def getMarks(self):
        self.name = input("Enter student Name: ")
        self.usn = input("Enter student USN: ")
        self.score[0] = int(input("Enter marks in Subject 1: "))
        self.score[1] = int(input("Enter marks in Subject 2: "))
        self.score[2] = int(input("Enter marks in Subject 3: "))
        self.score[3] = self.score[0] + self.score[1] + self.score[2]
    def display(self):
        percentage = self.score[3] / 3
        spcstr = "=" * 81
        print(spcstr)
        print("SCORE CARD DETAILS".center(81))
        print(spcstr)
        print("%15s %12s %8s %8s %8s %8s %12s" % (
            "NAME", "USN", "MARKS1", "MARKS2", "MARKS3", "TOTAL", "PERCENTAGE"))
        print(spcstr)
        print("%15s %12s %8d %8d %8d %8d %12.2f" % (
            self.name, self.usn, self.score[0], self.score[1],
            self.score[2], self.score[3], percentage))
        print(spcstr)
def main():
    s1 = Student()
    s1.getMarks()
    s1.display()
if __name__ == "__main__":
```