) Write a python program to save a text file in reverse order (Individual words will not get reversed)

```
def reverseorderfile():
    f1 = open("output1.txt", "w")
    with open("file.txt", "r") as myfile:
        data = myfile.read()
        reversed_data = data[::-1]
        f1.write(reversed_data)
        f1.close()
reverseorderfile()
```

Output:-

```
input.txt
Line 1
Line 2
Line 3
output.txt
Line 3
Line 2
Line 1
```

) Write a Python program to remove and print every third number from a list of numbers until the list becomes empty.

```
def removeandprintthirdelement(number):
    index = 2
    while nums:
        index = index % len(nums)
        print(nums.pop(index))
        index = index + 2
removeandprintthirdelement([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

Output:-

```
3
6
9
2
7
1
8
5
10
4
```

) Write a python program to create all possible strings by using 'a', 'e', 'i', 'o', 'u'. Use the characters exactly once.

```
import random
def possible_strings(list):
    characters = ['a', 'e', 'i', 'o', 'u']
    random.shuffle(characters)
    print(''.join(characters))
possible_.
import random
def possible_strings(characters):
    random.shuffle(characters)
    print(''.join(characters))
possible_strings(['a', 'e', 'i', 'o', 'u'])
```

Output:-

```
aeiou
aeoiu
aeuio
aeuoi
aioeu
....
```

94) Write a python program to check if every word starting with vowels have 'an' before it.

Ans)

```
def checkvowel(sentence):
    words = sentence.split()
    for i in range(1, len(words)):
        if words[i].lower() in 'aeiou' and words[i-1].lower():
            print("This sentence has every wood with vowels has an before it")

checkvowel("This is an apple")
```

Output:-

This sentence has word starting vowels has an bef

95) Write a python program to print a 2D list in a spiral

Ans)

```
def spiralmatrix(matrix):
    resultmatrix = []
    while matrix:
        resultmatrix = resultmatrix + matrix.pop(0)
        if matrix and matrix[0]:
            for row in matrix:
                resultmatrix.append(row.pop())
        if matrix:
            resultmatrix = resultmatrix + matrix.pop()[::-1]
        if matrix and matrix[0]:
            for row in matrix[::-1]:
                result.append(row.pop(0))
    return result
print(spiralmatrix([[1,2,3],[4,5,6],[7,8,9]]))
```

Output:-

[[1, 2, 3],
[6, 9, 8],
[7, 4, 5]

96) Write a python program to print the intersection between two lists.

Ans)

```
def intersection(list1, list2):
    return list(set(list1) & set(list2))

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

Output:

[3, 4

7) Write a Python program to generate a pseudo random string with templating support. (i.e. Input : "a___string"; a possible output string : a12agstring)

```
import random
import string

def generaterandomstr(inputstring):
    result = []
    for i in inputstring:
        if i == "_":
            result.append(random.choice(string.ascii_letters + string.digits))
        else:
            result.append(i)
    return ''.join(result)

print(generaterandomstr("a___string"))
```

Output :-

```
a5qwgstring
```

) Write a Python program to calculate permutations & combinations (using & w/o using library)

Using library :-

```
import math
def permutations(n, r):
    return math.perm(n, r)

def combinations(n, r):
    return math.comb(n, r)

print(permutations(5, 3))
print(combinations(5, 3))
```

Output :-

```
Permutation : 60
Combination : 10
```

Without library :-

```
def factorial(number):
    if(number == 1):
        return 1
    return number * factorial(number - 1)
def permutation(n, r):
    value = factorial(n) / factorial(n - r)
    return value
```

```
def combination(n,r):
    value = factorial(n) / factorial(r) * factorial(n-r)
    return value

print(permutation(5,3))
Print(combination(5,3))
```

Q9) Write a python program to simulate a student regi
with trivial operations (Records saved into a text file)

Ans)
```
import json
def addstudent(student_id, student_name, file_name='students
    student = {'ID': student_id, 'Name': student_name}
    with open(file_name, 'a') as f:
        f.write(json.dumps(student) + '\n')

def readstudent(file_name='students.txt'):
    with open(file_name, 'r') as f:
        students = f.readlines()
    return [json.loads(student) for student in students]

addstudent(1,"Alice")
Print(readstudent())
```

Output
Students.tx
{ID: 1, 'Name

Q10) Write a python program to make a browser histor
storage with forward and backward moving op

Ans)
```
class BrowserHistory:
    def __init__(self):
        self.history = []
        self.current_index = -1

    def visit(self, url):
        self.history = self.history[:self.current_index+
        self.history.append(url)
        self.current_index = self.current_index + 1
        print(f"Visited: {url}")
```

```
def back(self):
    if self.current_index > 0:
        self.current_index = self.current_index - 1;
        print(f"Back to: {self.history[self.current_index]}")

    else:
        print("No previous page").

def forward(self):
    if self.current_index < len(self.history) - 1:
        self.current_index = self.current_index + 1
        print(f"Forward to: {self.history[self.current_index]}")

    else:
        print("No forward page.")

browser = BrowserHistory()
browser.visit("google.com")
browser.visit("github.com")
browser.back()
browser.forward()
browser.visit("stackoverflow.com")
browser.back()
browser.back()
browser.forward()
```

output

```
visit: google.com
visit: github.com
Back: google.com
Forward: github.com
Back: google.com
visit: stackoverflow.com
Back: github.com
Back: google.com
Forward: github.com
```

) Write a python program to get sum of squares of list of integers.

```
) def sum_of_squares(list):
    result = 0
    for i in list:
        result = result + i**2
    return result

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

output

```
Sum of squares: 30
```

Q12) With a given integral number n, write a program to generate a dictionary that contains (i, i*i) such that is an integral number between 1 and n (both included), and then the program should print the dictionary.

Ans)
```
def generate square dict(n):
    dict = {}
    for i in range(1, n+1):
        dict[i] = i*i

print(generatesquaredict(5))
```

Output
{1:1, 2:4, 3:9, 4:

Q13) Write a program that accepts a sequence of comma separated 4 digit binary numbers as its input and check whether they are divisible by 5 or not.

Ans)
```
def checkbinarydivby5(binaries):
    div_by_5 = [b for b in binaries if int(b,2) %5==0]
    return div_by_5

binaries = input("Enter comma-separated binary no:").split(",")
result = checkbinarydivby5(binaries)
print(result)
```

Output
1010,1111,1100,100
Divisible by 5: 1010

Q14) Write a program that accepts a sentence and the number of letters and digits.

Ans)
```
def countletterdigits(sentence):
    letters = sum(c.isalpha() for c in sentence)
    digits = sum(c.isdigit() for c in sentence)
    return letters, digits

sentence = "Hello World! 123"
letters, digits = countletterdigits(sentence)
print(f"Letters: {letters}, Digits: {digits}")
```

Output
Letters
Number
Digits

Q15) A website requires the users to input a username and password to register. Write a program to check the validity of password input by users.
The following are the criteria for checking the password:-

1.) At least 1 letter between [a-z]
2.) At least 1 number between [0-9]
3.) At least 1 letter between [A-Z]
4.) At least 1 character from [$ # @]
5.) Minimum length of transaction password: 6
6.) Maximum length of transaction password: 12

Your program should accept a sequence of comma separated passwords and will check them according to the above criteria.

```
def validatepassword(password):
    if not (6 <= len(password) <= 12):
        return False

    has_lower = False
    has_upper = False
    has_digit = False
    has_special = False
    special_characters = {'$', '#', '@'}

    for char in password:
        if char.islower():
            has_lower = True

        elif char.isupper():
            has_upper = True

        elif char.isdigit():
            has_digit = True

        elif char in special_characters:
            has_special = True
    return has_lower and has_upper and has_digit and has_special
```

Output

"A1#abc, B2@Cde, abcd1234, A12@abc"

Valid passwords:
A1#abc, B2@Cde, A12@abc

```
def check_passwords(passwords):
    valid_passwords = [pwd for pwd in passwords if validate_password(pwd)]
    return valid_passwords

passwords = input("Enter comma-separated passwords: ").split(',')
valid = check_passwords(passwords)
print(valid)
```

Q16) A rudimentary calculator (with exception handling)

Ans)

```
def calculator():
    try:
        num1 = float(input("Enter the first no."))
        operator = input("Enter operator +, -, *, /: ")
        num2 = float(input("Enter second number: "))

        if operator == '+':
            result = num1+num2

        elif operator == '-':
            result = num1-num2

        elif operator == '*':
            result = num1*num2

        elif operator == '/':
            result = num1/num2

        else:
            print("Invalid operator")
        print(f"Result: {result}")

    except ValueError:as e:
        print(f"Error: {e}")
    except ZeroDivisionError:
        print("Division by 0 is not allowed")
calculator()
```

Output

```
Enter the first no 5
Enter operator +, -, *, /:
Enter second no 3
The result is: 11
```

17) Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle. Within the class rectangle write down the following functions:

i) create_rectangle()
Input parameters x, y, width, height
Return value: instance of Rectangle
Operation: Create a new instance of Rectangle

ii) str_rectangle()
Input parameters: rect, ~~dx, dy~~
Return value: ~~None~~ string
Operation: convert given Rectangle instance into string of form (x, y, width, height)

iii) shift_rectangle()
Input parameters: rect, dx, dy
Return value: None
Operation: change the x and y coordinates of the given Rectangle instance.

iv) offset_rectangle()
Input parameters: rect, dx, dy
Return value: instance of Rectangle
Operation: create a new Rectangle instance which is ~~offset~~ from the given instance in x and y coordinates by dx and dy respectively.

```
class Rectangle:
def __init__(self, x, y, width):
    self.x = x
    self.y = y
    self.width = width
    self.height = height
def area(self):
    return self.width * self.height
def create_rectangle(x, y, width, height):
    return Rectangle(x, y, width, height)
def str_rectangle(rect):
    return f"({rect.x}, {rect.y}, {rect.width}, {rect.height})"
```

```
def shift_rectangle(rect,dx,dy):
    rect.x = rect.x + dx
    rect.y = rect.y + dy
def offset_rectangle(rect, dx,dy):
    return Rectangle(rect.x + dx, rect.y + dy, rect.width, rect.he

rect = Rectangle.create_rectangle(0,0,10,5)
print(Rectangle.str_rectangle(rect))
Rectangle.shift_rectangle(rect, 2,3)
print(Rectangle.str_rectangle(rect))
new_rect = Rectangle.offset_rectangle(rect, 1,1)
print(Rectangle.str_rectangle(new_rect))
```

Output:-

(0, 0, 10, 5)
After shifting
(2, 3, 10, 5)

Q28) Write a simple Python class named Student and display its type. Also, display the __dict__ attribute keys and the value of the __module__ attribute of the Student class.

Ans)
```
class Student:
    pass

student = Student()
print(type(student))
print(student.__dict__)
print(Student.__module__)
```

Output

```
<class '__main__.Student'>
{}
__main__
```

Q29) Write a Python class named Student with two attributes student_id, student_name. Add a new attribute student_class display the entire attribute and their values of the said class. Now remove the student_name attribute and display the entire attribute with values.

Ans)
```
class Student:
    def __init__(self, student_id, student_name):
        self.student_id = student_id
        self.student_name = student_name
        self.student_class = None
```

```
ent = student(1, "John")
ent.student_class = "10th Grade"
(student.__dict__)
student.student_name
(student.__dict__)
```

Output

```
{'student_id': 1, student_name': "John Doe",
 'student_class': "10th Grade"}
After removal of student_name
{'student_id': 1, 'student_class':
'10th Grade'}
```

Create a Bus child class that inherits from the Vehicle class. The default fare charge of any vehicle is seating capacity * 100. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the final amount = total fare + 10% of the total fare.

```
Vehicle:
def __init__(self, seating_capacity):
    self.seating_capacity = seating_capacity

def fare(self):
    return self.seating_capacity * 100

Bus(Vehicle):
def fare(self):
    base_fare = super().fare
    return base_fare + 0.1 * base_fare

= Bus(50)
(bus.fare())
```

Output

```
Bus fare: 5500
```