Python Final Assignment (20CE155)

Link - https://github.com/settler-av/CE259-PIP (https://github.com/settler-av/CE259-PIP)

1. Descending frequency

In [4]:

```
test_str = input("Enter the string")
freq = dict()
for i in test_str:
    if i in freq:
        freq[i] = freq[i] + 1
    else:
        freq[i] = 1
for key in sorted(freq, key=freq.get, reverse=True):
    print("{} {}".format(str(key),str(freq[key])))
```

```
Enter the stringadnanvahora
a 4
n 2
d 1
v 1
h 1
o 1
r 1
```

2. Find Min, Max, mean, SD, Variance

In [5]:

```
import numpy as np
string = "10 50 80 70 49 23 11 4"
arr = string.split(" ")
# concert list int
arr = np.array(arr, dtype=int)
print("Min: {}".format(np.min(arr)))
print("Max: {}".format(np.max(arr)))
print("Mean: {}".format(np.mean(arr)))
print("SD: {}".format(np.std(arr)))
print("Variance: {}".format(np.var(arr)))
```

```
Min: 4
Max: 80
Mean: 37.125
SD: 27.25086007817001
Variance: 742.609375
```

3. You are given an integer array height of length n there are n vertical lines drawn such that the two endpoints of the line are (i,0) and (i, height[i]). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Return the maximum amount of water that can be contained.

In [6]:

```
def maxArea(A,le):
    #code here
    Pairs = []
    for i in range(le):
        for j in range(le):
            width = (j - i)
            pair = [A[i], A[j]]
            height = min(pair)
            Pairs.append(width*height)
    return max(Pairs)

for _ in range(0,int(input())):

    n = int(input())
    l = list(map(int,input().split()))

    print(maxArea(l,n))
```

```
1
5
1 2 5 4 4
8
```

4. Given list of integers, Write a program to print the count of all possible unique combination of nubers whose sum is equal to K

In [3]:

```
# Python 3 implementation of the approach

# Function to find all unique combination of
# given elements such that their sum is K
def unique_combination(l, sum, K, local, A):
    # If a unique combination is found
    if (sum == K):
        print("{", end="")
        for i in range(len(local)):
            if (i != 0):
                print(" ", end="")
            print(local[i], end="")
            if (i != len(local) - 1):
                print(", ", end="")
        print("}")
        return
    # For all other combinations
    for i in range(l, len(A), 1):
        # Check if the sum exceeds K
        if (sum + A[i] > K):
            continue
        # Check if it is repeated or not
        if (i > l and
                A[i] == A[i - 1]):
            continue
        # Take the element into the combination
        local.append(A[i])
        # Recursive call
        unique_combination(i + 1, sum + A[i],
                           K, local, A)
        # Remove element from the combination
        local.remove(local[len(local) - 1])

# Function to find all combination
# of the given elements
def Combination(A, K):
    # Sort the given elements
    A.sort(reverse=False)
    local = []
    unique_combination(0, 0, K, local, A)

if __name__ == '__main__':
    A = [10, 1, 2, 7, 6, 1, 5]
    K = 8
    Combination(A, K)
```

```
{1, 1, 6}
{1, 2, 5}
{1, 7}
{2, 6}
```

5. Explaing about differnt types of exeption in python

Exception hierarchy

```
Exception
    ├── ArithmeticError
    │   ├── FloatingPointError
    │   ├── OverflowError
    │   └── ZeroDivisionError
    ├── AssertionError
    ├── AttributeError
    ├── BufferError
    ├── EOFError
    ├── ImportError
    │   └── ModuleNotFoundError
    ├── LookupError
    │   ├── IndexError
    │   └── KeyError
    ├── MemoryError
    ├── NameError
    │   └── UnboundLocalError
    ├── OSError
    │   ├── BlockingIOError
    │   ├── ChildProcessError
    │   ├── ConnectionError
    │   │   ├── BrokenPipeError
    │   │   ├── ConnectionAbortedError
    │   │   ├── ConnectionRefusedError
    │   │   └── ConnectionResetError
    │   ├── FileExistsError
    │   ├── FileNotFoundError
    │   ├── InterruptedError
    │   ├── IsADirectoryError
    │   ├── NotADirectoryError
    │   ├── PermissionError
    │   ├── ProcessLookupError
    │   └── TimeoutError
    ├── ReferenceError
    ├── RuntimeError
    │   ├── NotImplementedError
    │   └── RecursionError
    ├── StopAsyncIteration
    ├── StopIteration
    ├── SyntaxError
    │   └── IndentationError
    │       └── TabError
    ├── SystemError
    ├── TypeError
    ├── ValueError
    │   └── UnicodeError
    │       ├── UnicodeDecodeError
    │       ├── UnicodeEncodeError
    │       └── UnicodeTranslateError
    └── Warning
        ├── BytesWarning
        ├── DeprecationWarning
```

```
├── FutureWarning
├── ImportWarning
├── PendingDeprecationWarning
├── ResourceWarning
├── RuntimeWarning
├── SyntaxWarning
├── UnicodeWarning
└── UserWarning
```

In [7]:

```
"""
1. Exceptions are classes and they can be used just like all other classes.
2. ValueError and TypeError are some of the most commonly used exceptions.
3. The try and except keywords can be used for attempting to do something and then doing so
4. It's possible to raise exceptions with the raise keyword. This is also known as throwing
5. Raise exceptions if they are meant to be displayed for programmers and use sys.stderr an
"""
while True:
    try:
        number = int(input("Enter a number: "))
        break
    except ValueError:
        print("That's not a valid number! Try again.")

print("Your number doubled is:", number * 2)




# These are here so you can change them to customize the program
# easily.
default_greeting = "Hello World!"
filename = "greeting.txt"


import sys

def askyesno(question):
    while True:
        answer = input(question + ' (y or n) ')
        if answer == 'Y' or answer == 'y':
            return True
        if answer == 'N' or answer == 'n':
            return False

def greet():
    with open(filename, 'r') as f:
        for line in f:
            print(line.rstrip('\n'))

try:
    greet()
except OSError:
    print("Cannot read '%s'!" % filename, file=sys.stderr)
    if askyesno("Would you like to create a default greeting file?"):
        with open(filename, 'w') as f:
            print(default_greeting, file=f)
        greet()
```

```
Enter a number: 2
Your number doubled is: 4
Hello World!
```

6. Complete Django tutorial (Part 1 to 7) from the official

website

https://www.djangoproject.com/ (https://www.djangoproject.com/)

Link of the work: https://github.com/settler-av/CE259-PIP/tree/master/Final%20Assignment/ (https://github.com/settler-av/CE259-PIP/tree/master/Final%20Assignment/)

7. Write a Django code to send an email with attachment

Link of the work: https://github.com/settler-av/CE259-PIP/tree/master/Final%20Assignment/Email-with-django (https://github.com/settler-av/CE259-PIP/tree/master/Final%20Assignment/Email-with-django)

8. Program to demonstrate Overriding of base Class method in derived class

---

In [10]:

```
# Python program to demonstrate
# method overriding


# Defining parent class
class Parent():

    # Constructor
    def __init__(self):
        self.value = "Inside Parent"

    # Parent's show method
    def show(self):
        print(self.value)

# Defining child class
class Child(Parent):

    # Constructor
    def __init__(self):
        self.value = "Inside Child"

    # Child's show method
    def show(self):
        print(self.value)


# Driver's code
obj1 = Parent()
obj2 = Child()

obj1.show()
obj2.show()
```

```
Inside Parent
Inside Child
```

9. Write pythonic code to create a function named move_rectangle that takes an object of Reactangle and two numbers called dx and dy. It should move the rectangle by the given amount.

In [11]:

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        self.corner = {'x': 0, 'y': 0}
    def display(self):
        print(self.corner['x'], self.corner['y'])

    def move_rectangle(self, dx, dy):
        self.corner['x'] += dx
        self.corner['y'] += dy



r1 = Rectangle(10, 20)
r1.display()
r1.move_rectangle(5, 10)
r1.display()
```

```
0 0
5 10
```