Python Final Assignment (20CE155)

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

1. Descending frequency

In [2]:

```python
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 stringadnan
a 2
n 2
d 1
```

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

In [3]:

```python
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 [4]:
```python
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
2
1 1
1
```

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 [5]:
```python
# 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 [6]:

```python
"""
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 d
4. It's possible to raise exceptions with the raise keyword. This is also known as t
5. Raise exceptions if they are meant to be displayed for programmers and use sys.st
"""
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: 5
Your number doubled is: 10
Hello World!
```

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

https://www.djangoproject.com/

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

Manage.py

In [ ]:

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'djangoTut.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__':
    main()
```

models.py

In [ ]:

```
from django.db import models
import datetime

from django.db import models
from django.utils import timezone

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    def __str__(self):
            return self.question_text


class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.question_text

    def was_published_recently(self):
        now = timezone.now()
        return now - datetime.timedelta(days=1) <= self.pub_date <= now# 7. Write a
```

urls.py

In [ ]:

```
from django.urls import path

from . import views

app_name = 'polls'
urlpatterns = [
    path('', views.IndexView.as_view(), name='index'),
    path('<int:pk>/', views.DetailView.as_view(), name='detail'),
    path('<int:pk>/results/', views.ResultsView.as_view(), name='results'),
    path('<int:question_id>/vote/', views.vote, name='vote'),
]
```

Views.py

In [ ]:

```
from django.shortcuts import render, get_object_or_404
from django.http import HttpResponse, Http404, HttpResponseRedirect
from django.urls import reverse
from django.views import generic

from .models import Choice,Question


class IndexView(generic.ListView):
    template_name = 'polls/index.html'
    context_object_name = 'latest_question_list'

    def get_queryset(self):
        """Return the last five published questions."""
        return Question.objects.order_by('-pub_date')[:5]


class DetailView(generic.DetailView):
    model = Question
    template_name = 'polls/detail.html'


class ResultsView(generic.DetailView):
    model = Question
    template_name = 'polls/results.html'


def vote(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    try:
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        # Redisplay the question voting form.
        return render(request, 'polls/detail.html', {
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:
        selected_choice.votes += 1
        selected_choice.save()
        # Always return an HttpResponseRedirect after successfully dealing
        # with POST data. This prevents data from being posted twice if a
        # user hits the Back button.
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

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

7. Email with Django

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

In [ ]:

```
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.views import View

from django.core.mail import EmailMessage

from django.conf import settings
from .forms import EmailForm

class EmailAttachementView(View):
    form_class = EmailForm
    template_name = 'emailattachment.html'

    def get(self, request, *args, **kwargs):
        form = self.form_class()
        return render(request, self.template_name, {'email_form': form})

    def post(self, request, *args, **kwargs):
        form = self.form_class(request.POST, request.FILES)

        if form.is_valid():
            subject = form.cleaned_data['subject']
            message = form.cleaned_data['message']
            email = form.cleaned_data['email']
            files = request.FILES.getlist('attach')

            try:
                mail = EmailMessage(subject, message, settings.EMAIL_HOST_USER, [ema
                for f in files:
                    mail.attach(f.name, f.read(), f.content_type)
                mail.send()
                return render(request, self.template_name, {'email_form': form, 'err
            except:
                return render(request, self.template_name, {'email_form': form, 'err

        return render(request, self.template_name, {'email_form': form, 'error_messa

    # Single File Attachment
    # def post(self, request, *args, **kwargs):
    #     form = self.form_class(request.POST, request.FILES)

    #     if form.is_valid():

    #         subject = form.cleaned_data['subject']
    #         message = form.cleaned_data['message']
    #         email = form.cleaned_data['email']
    #         attach = request.FILES['attach']

    #         try:
    #             mail = EmailMessage(subject, message, settings.EMAIL_HOST_USER, [e
    #             mail.attach(attach.name, attach.read(), attach.content_type)
    #             mail.send()
    #             return render(request, self.template_name, {'email_form': form, 'e
    #         except:
    #             return render(request, self.template_name, {'email_form': form, 'e

    #     return render(request, self.template_name, {'email_form': form, 'error_mes
```

In [ ]:

```
from django.urls import path
from emailattachment.views import EmailAttachementView

urlpatterns = [
    path('', EmailAttachementView.as_view(), name='emailattachment')

]
```

In [ ]:

```
from django import forms

class EmailForm(forms.Form):
    email = forms.EmailField()
    subject = forms.CharField(max_length=100)
    attach = forms.FileField(widget=forms.ClearableFileInput(attrs={'multiple': True
    message = forms.CharField(widget = forms.Textarea)
```

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

---

In [7]:

```python
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
```