Solutions to Sample Midterm A

Learning Objective:

- Create Python code to automate a given task.

Instructions:

The midterm tests your mastery of skills taught in Weeks 1-5, which culminates in creating simulation models using Python and algorithmic thinking. The exam is 80 minutes, and is open notes but closed computer. You can bring paper notes or books of any kind, but no computers, tablets, or cell phones are allowed. Before the exam is graded, do not share the questions or your solutions with anyone else, including students of other sections. When you turn in the exam, you must also hand in all scrap paper that you wrote on. Do not use a cell phone, tablet or computer in the classroom before all of the exams are handed in. **Any violation of academic integrity will result in a zero grade for the midterm for everyone involved.**

As long as you fulfill all the specifications described in the problem description, it doesn't matter how you solve the problem or how efficient is your code. However, if you cannot solve a problem, you may get partial credits for submitting whatever you have, including any parts of the four steps of algorithmic thinking.

Note: All three questions are motivated by the issue of procuring toilet paper, which was a challenging problem in many parts of the world at the beginning of COVID-19.

Q1. Estimating Household Demand for Toilet Paper (7 Points)

Suppose your household has k persons. When a person is healthy, the amount of toilet paper used by the person in a given day is distributed uniformly between 5 and 10 sheets, independent of other persons and independent of other days. (Note that it is possible that a person uses a fractional number of sheets.) However, when a person is sick, the parameters of the uniform distribution are 20 and 40 sheets respectively. Assume for simplicity that on any given day, a person is sick with probability p and healthy with probability $1-p$, and health is independent across days and across persons.

Write a function "demand_estimate" with the following input arguments:

- **days**: the number of days to simulate demand. You can assume that this is a positive integer.
- **k** (default value 1): the number of persons in your household. You can assume that this is a positive integer.
- **p** (default value .02): the probability of being sick on a given day. You can assume that this is a number between 0 and 1.

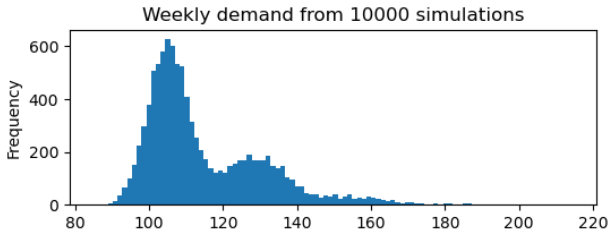
Your function should return one number representing one random sample of the total amount of toilet paper used by your household during the given period of days. See the sample outputs below for illustrations.

```
[1]: # Final Solution
     from numpy.random import default_rng
     def demand_estimate(days,k=1,p=.02):
         rng=default_rng()
         total=0
         for i in range(days):
             sick=rng.binomial(k,p)
             for j in range(sick):
                 total+=rng.uniform(20,40)
             for j in range(k-sick):
                 total+=rng.uniform(5,10)
         return total
```

```
[2]: # Test code 1
     demand_estimate(7)
```

```
48.609976093943594
```

```
[15]: # Test code 2
      import pandas as pd
      import matplotlib.pyplot as plt
      estimates=pd.Series([demand_estimate(7, 2, 0.03) for i in range(10000)])
      estimates.plot(kind='hist',title='Weekly demand from 10000␣
 ↪simulations',bins=100,figsize=(6,2))
      plt.show()
      print(f'Mean: {estimates.mean():.2f} \tStd: {estimates.std():.2f}')
```




```
Mean: 114.81    Std: 16.08
```

Q2. Simulating the Availability of Toilet Paper (8 Points)

Suppose a certain retail store receives s packages of toilet paper from the store's sole supplier every Monday morning before the store opens. Packages of toilet paper are sold on a first-come-first-serve basis at a fixed price, and the store opens for business every day. This question asks you to estimate the availability of toilet paper given a forecast of daily demand.

Write a function called "supply_simulation" with the following input arguments:

- **demand**: a list specifying the number of packages of toilet paper that would be sold each day if the store had infinite supply. You may assume that every number in the list is non-negative, and that the list represents a sequence of consecutive days in which the day at position 0 is a Monday.
- **s**: a positive integer representing the number of packages of toilet paper the store receives every Monday before the store opens. This supply is immediately made available to shoppers when the store opens.

You may assume that before Day 0, the store has zero inventory carried over from the past. In general, unsold packages can be carried over to the future with no limits on storage capacity. If the store has insufficient inventory to serve all demand in a day, the unserved customers buy from somewhere else.

Your function should return two numbers:

- **served_proportion**: The proportion of demand that is served. This is the total number of packages sold divided by the total demand during the given period.
- **good_days_proportion**: The proportion of days in which all demand is served. Define a day to be "good" if all demand for that day is served. The desired number is the number of good days divided by the total number of days in the given period.

Hint: You can determine if a day is a Monday by checking the remainder of the position when divided by 7. Since position 0 is a Monday, then position `t` is a Monday if and only if the remainder of `t` when divided by 7 is 0. In other words, position `t` corresponds to a Monday if and only if `t%7==0` evaluates to True.

See the sample outputs below for illustrations.

[4]:
```
# Final Solution
def supply_simulation(demand,s):
    inventory=0
    total_served=0
    good_days=0
    num_days=len(demand)
    for day in range(num_days):
        if day%7==0:
            inventory+=s
        served=min(inventory,demand[day])
        if served==demand[day]:
            good_days+=1
        total_served+=served
        inventory-=served
    served_proportion=total_served/sum(demand)
    good_days_proportion=good_days/num_days
    return served_proportion,good_days_proportion
```

[5]:
```
# Test code 1
s,g=supply_simulation([20,35,60,20,10,30,50,30,20,10],100)
print('Proportion of total demand that is served: ',s)
print('Proportion of days in which all demand is served: ',g)
```

```
Proportion of total demand that is served:  0.5614035087719298
Proportion of days in which all demand is served:  0.5
```

[6]:
```
# Test code 2
demand=([10]*7)+([20]*3)
supply_simulation(demand,70)
```

```
(1.0, 1.0)
```

[7]:
```
# Test code 3
demand=([20]*10)
supply_simulation(demand,70)
```

```
(0.65, 0.6)
```

Q3. Distance Travelled to Buy Toilet Paper (9 Points)

Suppose that you are on a quest to buy toilet paper for your household. There is a list of stores that you plan to visit in a given order, and you will come home as soon as you find a store that has toilet paper in stock. The question asks you to build a simulation model of the total distance you travel on your quest.

Write a function called "travel_distance" with the following input arguments:

- **locations**: a list in which each element represents the location of a distinct retails store, given as a two-element list of the x and y coordinates. For example, if the list is `[[3,4],[-3,4],[-8,6]]`, then the first store you plan to visit is at location $(x, y) = (3, 4)$, the second store at $(-3, 4)$, and the third store at $(-8, 6)$. Assume that your home is located at $(x, y) = (0, 0)$.
- **probabilities**: a list of numbers between 0 and 1 (inclusive). You can assume that the length of this list is the same as the length of the list "locations", and each element represents the probability that the corresponding store has toilet paper in stock.

For simplicity, assume that you travel the minimum distance between any two points, so that the distance between coordinates (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Moreover, whether toilet paper is in stock is independent across stores. To do square roots, you can do for example:

```
import math
math.sqrt(5)
```

Your function should return one number, which is the total distance travelled on your trip, starting from home and counting all distances travelled until you return to home. When you find that toilet paper is not in stock at a store, you travel directly to the next store. You return to home only after you have found toilet paper, or after you have gone through all of the stores (in which case you return home empty handed). See the sample outputs for illustrations. Note that the returned number is deterministic if all probabilities are either 0 or 1 (as in test codes 1-4), but should be random otherwise (as in test code 5).

[8]:
```
# Final Solution
import math
from numpy.random import default_rng
def distance(point1,point2):
    return math.sqrt((point1[0]-point2[0])**2+(point1[1]-point2[1])**2)

def travel_distance(locations, probabilities):
    rng=default_rng()
    home=[0,0]
    n=len(locations)
    total_distance=0
    last_point=home
    for i in range(n):
        current_point=locations[i]
        prob=probabilities[i]
        total_distance+=distance(last_point,current_point)
        if rng.binomial(1,prob) or i==n-1:
            total_distance+=distance(current_point,home)
            break
        last_point=current_point
    return total_distance
```

[9]:
```
# Test code 1
travel_distance([[3,4],[-3,4],[-8,6]],[0,1,1])
```

```
16.0
```

[10]:
```
# Test code 2
print(travel_distance([[3,4],[-3,4],[-8,6]],[1,1,0]))
```

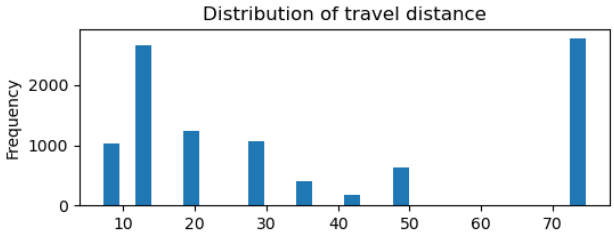
```
10.0
```

[11]:
```
# Test code 3
travel_distance([[3,4],[-3,4],[-8,6]],[0,0,0])
```

```
26.385164807134505
```

[14]:
```
# Test code 4
locations=[[3,2],[5,4],[7,-1],[5,-8],[-5,-4],[-6,2],[-8,4],[-20,10]]
probabilities=[0.1,0.3,0.2,0.2,0.1,0.05,0.2,0.05]
import pandas as pd
import matplotlib.pyplot as plt
distances=pd.Series([travel_distance(locations,probabilities) for i in␣
↪range(10000)])
distances.plot(kind='hist',bins=30,title='Distribution of travel␣
↪distance',figsize=(6,2))
plt.show()
```

```
print('Mean:',round(distances.mean(),2))
print('Standard deviation:',round(distances.std(),2))
```




```
Mean: 35.54
Standard deviation: 26.34
```