Probability (homework 1).

Problem 1

The Birthday Problem: This is a classic problem that has a nonintuitive answer. Suppose there are N students in a room.

Part a)

What is the probability that at least two of them have the same birthday (month and day)? (Assume that each day is equally likely to be a student's birthday, that there are no sets of twins, and that there are 365 days in the year. Do not include leap years).

Note: Jupyter has two types of cells: Programming and Markdown. Programming is where you will create and run R code. The Markdown cells are where you will type out expalantions and mathematical expressions. Here is a document on Markdown some basic markdwon syntax. Also feel free to look at the underlying markdown of any of the provided cells to see how we use markdown.

$$P(\text{At least two have same birthday}) = ?$$
$$= \text{YOUR ANSWER HERE}$$

Answer:

To begin with, we suppose that N is the number of people in a room, then:

(✓) If $N > 365$, then by the pigeon principle is necessary that at least 2 people of the set of N people in the room have the same birthday.

(✓) If $0 \leq N \leq 365$, then the number of possible data is 365^N.

This is true because the multiplication principle.

$$\underset{\substack{1\\(\text{student})}}{\underline{365}} \times \underset{\substack{2\\(\text{student})}}{\underline{365}} \times \underset{\substack{3\\(\text{student})}}{\underline{365}} \times \underset{\substack{4\\(\text{student})}}{\underline{365}} \times \cdots \times \underset{\substack{N\\(\text{student})}}{\underline{365}} = 365^N.$$

Also, the probability that no person has birthday the same day is:

$$\frac{365 \times 364 \times \cdots \times (365-(N-1))}{365^N} = \frac{\frac{(365)!}{(365-N)!}}{(365)^N}$$

therefore, the probability that at least 2 people in the room have the same birthday is:

$$\boxed{P_N = 1 - \frac{\frac{(365)!}{(365-N)!}}{(365)^N} = \frac{(365)^N - \frac{(365)!}{(365-N)!}}{(365)^N}}$$

Part b)

How large must N be so that the probability that at least two of them have the same birthday is at least 1/2?

YOUR ANSWER HERE

Answer:

In this part, we need to find the minimum positive integer N such that:

$$\boxed{P_N \geq \frac{1}{2}}.$$

For this we will use the following code:

```
def permutation(n,m):
    result=1
    for i in range(m):
        result = result * (n-i)
    return result

def P(N):
    return ((365**N)-permutation(365,N))/((365**N))

N=1
variable=True
while variable==True:
    if P(N)>=0.5:
        variable=False
        break
    N = N+1

print(N)
```

```
C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.2032.0_x64__qbz5n2kfra8p0\python3.10.exe
23
```

And the result is $N \geq 23$.

Part c)

Plot the number of students on the x-axis versus the probability that at least two of them have the same birthday on the y-axis.

YOUR ANSWER HERE

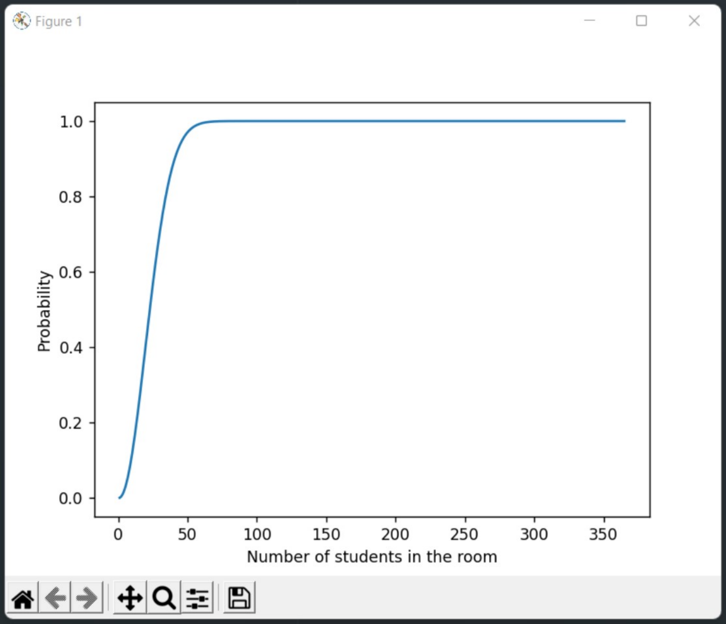
Answer.

```
import matplotlib.pyplot as plt


def permutation(n,m):

def P(N):
    return ((365**N)-permutation(365,N))/((365**N))


list_1=[i for i in range(1,365+1)]
list_2=[P(i) for i in range(1,365+1)]
plt.plot(list_1,list_2)
plt.xlabel("Number of students in the room")
plt.ylabel("Probability",)
plt.show()
```



Problem 2

One of the most beneficial aspects of R, when it comes to probability, is that it allows us to simulate data and random events. In the following problem, you are going to become familiar with these simulation functions and techniques.

Part a)

Let X be a random variable for the number rolled on a fair, six-sided die. How would we go about simulating X?

Start by creating a list of numbers [1, 6]. Then use the `sample()` function with our list of numbers to simulate **a single** roll of the die, as in simulate X. We would recommend looking at the documentation for `sample()`, found here, or by executing `?sample` in a Jupyter cell.

Answer.

First of all, it is important to note that $X : \{1,2,3,4,5,6\} \longrightarrow \mathbb{R}$ is defined as:

$$\begin{cases} X(i) = i, \\ \text{for all } i \in \{1,2,3,4,5,6\}. \end{cases}$$

And using the sample( ) function we have the following simulating of X.

```
# Your Code Here
numbers = 1:6
sample(x = numbers, 1, replace = TRUE)
```

5

```
# Your Code Here
numbers = 1:6
sample(x = numbers, 1, replace = TRUE)
```

4

```
# Your Code Here
numbers = 1:6
sample(x = numbers, 1, replace = TRUE)
```

2

Part b)

In our initial problem, we said that X comes from a fair die, meaning each value is equally likely to be rolled. Because our die has 6 sides, each side should appear about $1/6^{th}$ of the time. How would we confirm that our simulation is fair?

What if we generate multiple instances of X? That way, we could compare if the simulated probabilities match the theoretical probabilities (i.e. are all $1/6$).

Generate 12 instances of X and calculate the proportion of occurances for each face. Do your simulated results appear to come from a fair die? Now generate 120 instances of X and look at the proportion of each face. What do you notice?

Note: Each time you run your simulations, you will get different values. If you want to guarantee that your simulation will result in the same values each time, use the `set.seed()` function. This function will allow your simulations to be reproducable.

Answer.

```
# Your Code Here
numbers = 1:6
sample_of_12=sample(x = numbers, size = 12, replace = TRUE)
sample_of_120=sample(x = numbers, size = 120, replace = TRUE)
```

```
print(sample_of_12)
```

```
 [1] 2 3 3 2 6 2 5 6 3 4 4 2
```

```
print(sample_of_120)
```

```
  [1] 1 3 1 5 4 5 2 4 4 5 4 5 1 4 5 5 5 6 3 1 5 2 5 3 3 4 1 1 5 1 4 2 4 5 6 2 2
 [38] 2 2 1 2 6 3 1 5 6 6 1 6 5 4 5 3 1 1 1 3 1 5 2 1 3 6 6 2 1 4 4 5 5 2 2 1 1
 [75] 4 2 1 1 3 4 5 4 6 2 5 4 5 6 2 5 6 1 6 5 4 5 4 6 6 3 5 1 3 4 2 4 2 2 1 2 4
[112] 2 4 4 5 2 4 4 4 4
```

(✓) Results of 12 instances of X.

```
{2: 0.3333, 3: 0.25, 4: 0.1667, 5: 0.0833, 6: 0.1667}
```

(✓) Results of 120 instances of X

```
{1: 0.1917, 2: 0.175, 3: 0.0917, 4: 0.2167, 5: 0.2083, 6: 0.1167}
```

And the theorical proportion of occurances for each face is $\frac{1}{6} = 0,1666\ldots$

Thus, we could note that the proportion of occurances for each face is close to the theorical value, if the number of instances of X is big enough.

Part c)

What if our die is not fair? How would we simulate that?

Let's assume that Y comes from an unfair six-sided die, where $P(Y = 3) = 1/2$ and all other face values have an equal probability of occuring. Use the `sample()` function to simulate this situation. Then display the proportion of each face value, to confirm that the faces occur with the desired probabilities. Make sure that n is large enough to be confident in your answer.

Answer.

In this case, we define $S=\{1,2,3,4,5,6\}$,

$$\begin{aligned} y: S &\to \mathbb{R} \\ i &\mapsto y(i)=i \end{aligned}$$ and a probability

$p: \{A / A \subseteq S\} \to \mathbb{R}$ described as

$$P\{\{i\}\} = \begin{cases} \frac{1}{10} & \text{if } i \neq 3, \\ \frac{1}{2} & \text{if } i = 3. \end{cases}$$

then using the Sample( ) function to simulate the previous situation, we have:

```r
# Your Code Here
sample_of_unfair=sample(c(1,2,3,4,5,6), 1000, replace = TRUE, prob = c(1/10,1/10,1/2,1/10,1/10,1/10))
```

```r
sample_of_unfair
print(sample_of_unfair)
```

```
   [1] 3 3 5 3 3 3 6 4 3 3 3 5 4 1 3 2 6 2 3 2 4 3 6 5 3 1 3 3 1 2 3 3 3 3 3 3 3
  [38] 3 3 5 6 4 3 5 3 4 6 3 4 2 3 2 3 1 6 4 3 4 3 3 3 4 2 2 3 3 3 3 4 3 3 3 3 3
  [75] 3 4 4 3 4 6 3 5 3 3 2 2 2 3 3 6 3 1 3 3 4 3 2 3 2 2 3 3 1 4 3 1 3 3 4 3 5
 [112] 3 5 2 6 3 1 1 2 2 3 3 3 3 3 2 6 3 4 3 4 6 4 3 2 5 4 4 3 5 1 5 3 3 1 2 3 2
 [149] 4 6 3 3 5 3 5 3 4 3 6 4 3 3 3 3 4 2 3 6 6 3 3 3 2 3 3 3 3 3 2 1 3 4 3 4 2
 [186] 3 3 5 4 4 3 3 4 3 2 2 3 1 2 5 3 3 2 3 3 4 1 2 1 3 3 1 4 3 3 5 4 2 6 3 5 1
 [223] 2 4 5 1 4 3 6 3 3 4 3 3 6 2 3 5 2 6 3 3 1 3 3 3 3 5 5 3 2 3 4 3 6 5 6 1 3
 [260] 3 2 4 2 3 4 3 3 3 3 3 3 3 3 3 1 3 3 3 6 4 2 4 3 4 5 5 1 3 6 3 2 3 4 1 2 2
 [297] 3 3 2 6 2 3 3 2 3 3 2 3 4 6 3 3 1 3 3 4 3 5 3 5 3 6 5 6 3 3 6 3 2 3 1 5 3
 [334] 1 3 3 3 3 3 6 3 5 3 3 3 2 6 5 1 4 3 3 2 3 3 3 6 3 6 4 4 3 1 6 5 3 3 1 6 3
 [371] 3 1 3 6 1 3 1 5 3 2 5 3 3 3 5 5 2 1 5 3 4 3 3 3 3 3 1 5 3 3 3 3 5 3 3 3 1
 [408] 3 3 6 3 5 5 6 3 6 1 1 4 3 2 3 2 4 5 3 3 2 3 6 3 3 2 4 2 5 3 3 3 2 3 3 1 3
 [445] 4 4 3 3 3 3 2 3 3 3 6 3 1 1 4 6 3 6 4 4 3 3 3 3 1 5 3 3 1 5 3 3 3 3 5 3 3
 [482] 3 3 4 3 3 3 6 2 3 3 2 6 1 3 3 3 1 3 6 3 3 6 2 1 1 5 3 3 4 5 3 4 1 3 4 4 4
 [519] 5 3 5 6 3 2 6 3 3 3 4 3 6 4 3 6 5 3 5 6 3 3 2 3 3 3 3 2 3 3 6 5 3 3 3 3 3
 [556] 3 3 3 3 4 6 3 3 2 3 4 3 5 6 3 3 3 2 3 2 6 3 5 5 1 4 4 4 2 5 1 4 3 6 3 6 2
 [593] 3 2 2 3 3 4 3 2 4 4 2 3 3 3 3 3 2 3 2 3 3 3 3 1 3 6 3 2 1 4 5 5 4 1 3 1 4
 [630] 2 4 3 1 2 6 1 3 3 1 2 2 3 2 3 3 6 3 6 2 3 2 4 3 3 3 3 3 3 4 3 3 3 3 5 3 2
 [667] 3 1 6 3 6 3 3 3 3 5 3 4 3 3 5 1 6 3 1 4 4 3 4 1 3 5 6 3 6 6 3 1 4 3 3 4 3
 [704] 3 4 3 3 3 3 1 2 6 3 3 3 3 6 3 3 6 3 3 4 1 2 1 3 1 4 1 4 1 5 6 5 2 3 3 4 3
 [741] 3 4 3 5 5 3 4 3 5 2 2 3 1 3 1 3 6 3 3 3 1 6 3 3 3 5 3 2 1 2 3 4 3 3 3 3 1
 [778] 3 3 6 3 3 1 5 3 3 3 5 6 3 3 3 3 3 3 4 3 3 3 3 2 5 4 6 2 6 4 6 5 3 5 3 1 2
 [815] 3 3 3 3 4 3 3 3 1 3 3 2 3 3 3 2 6 3 6 3 3 3 3 3 2 2 3 3 6 1 1 3 3 5 3 3 3
 [852] 3 1 3 2 3 3 4 6 5 1 3 3 6 3 3 1 6 3 5 3 1 3 3 3 1 4 3 3 3 5 2 4 5 2 1 4 4
 [889] 5 6 2 4 3 5 4 3 3 2 3 2 4 3 3 3 3 3 1 3 4 5 6 3 4 4 3 5 4 4 3 6 3 2 3 3 3
 [926] 3 3 3 3 3 3 3 5 2 4 3 6 3 3 5 2 3 4 3 3 4 4 3 3 3 1 3 3 5 1 3 3 4 4 4 3 3
 [963] 1 4 3 3 3 3 3 1 3 5 3 6 3 3 1 3 3 3 4 5 3 3 1 2 6 2 3 3 5 3 3 3 3 1 6 3 3
[1000] 3
```

Thus the results of 1000 instances of y are:

```
{1: 0.091, 2: 0.108, 3: 0.502, 4: 0.12, 5: 0.088, 6: 0.091}
```