**PROBABILITY METHODS IN ENGINEERING**

**ASSIGNMENT # 05**



**Spring 2023**

**CSE-209 Probability Methods In Engineering**

Submitted by: **Shahzad Bangash, Suleman Shah , Ali Asghar**

Registration No. : **21PWCSE1980, 21PWCSE1983, 21PWCSE2059**

Class Section: **C**

“On my honor, as students of University of Engineering and Technology, We have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

**Dr. Amaad Khalil**

**DATE**: 20 **/** June **/** 2023

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Probability Concepts and Implementation in Python**

**Introduction:**

Probability theory is a fundamental branch of mathematics that deals with the study of uncertainty and randomness. It provides a framework for understanding and quantifying the likelihood of events occurring. In this report, we will explore various probability concepts and their implementation in Python.

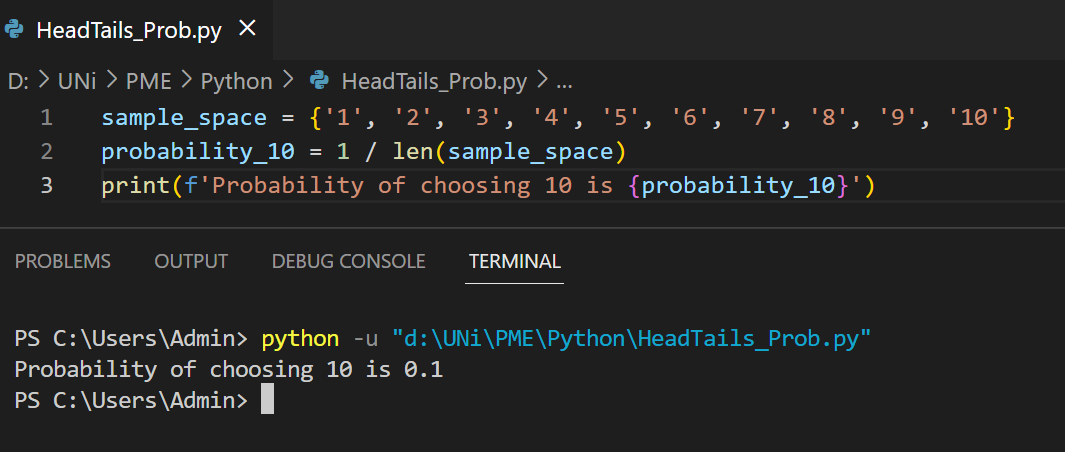
**1. Probability of an Event:**

The probability of an event is a measure of the likelihood of that event occurring. It is represented by a number between 0 and 1, where 0 indicates impossibility and 1 indicates certainty. In Python, we can calculate the probability of an event by dividing the favorable outcomes by the total possible outcomes.

**Example:**

Consider a sample space with elements {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. The probability of choosing '10' can be calculated as follows:

**Code & Output:**



A screen shot of a computer code

Description automatically generated with low confidence

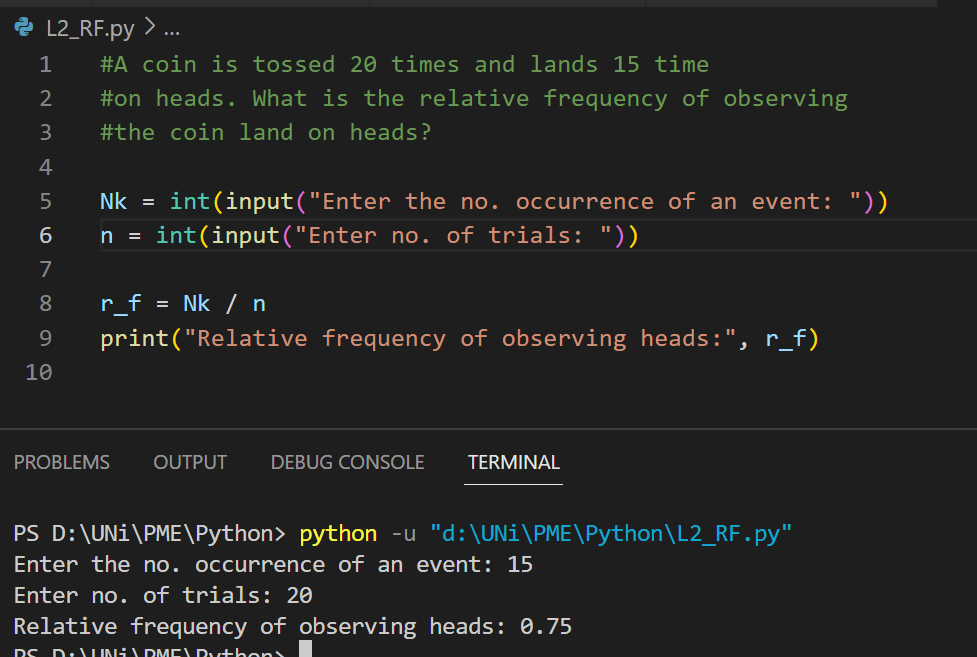
**2. Relative Frequency:**

Relative frequency is a concept used to estimate the probability of an event based on observed data. It is calculated by dividing the number of times the event occurred by the total number of trials or observations. In Python, we can compute the relative frequency using simple arithmetic operations.

**Example:**

Suppose a coin is tossed 20 times, and it lands on heads 15 times. The relative frequency of observing heads can be calculated as follows:

**Code & Output:**



**3. Conditional Probability:**

Conditional probability measures the likelihood of an event occurring given that another event has already occurred. It is calculated by dividing the probability of the intersection of the two events by the probability of the condition event. In Python, we can define a function to calculate the conditional probability.

**Example:**

Consider a scenario where 25% of a class passed both quizzes, and 42% passed the first quiz. We can calculate the percent of those who passed the first quiz and also passed the second quiz using the conditional probability formula.

**Code and Output:**

**A picture containing text, screenshot, software, font

Description automatically generated**

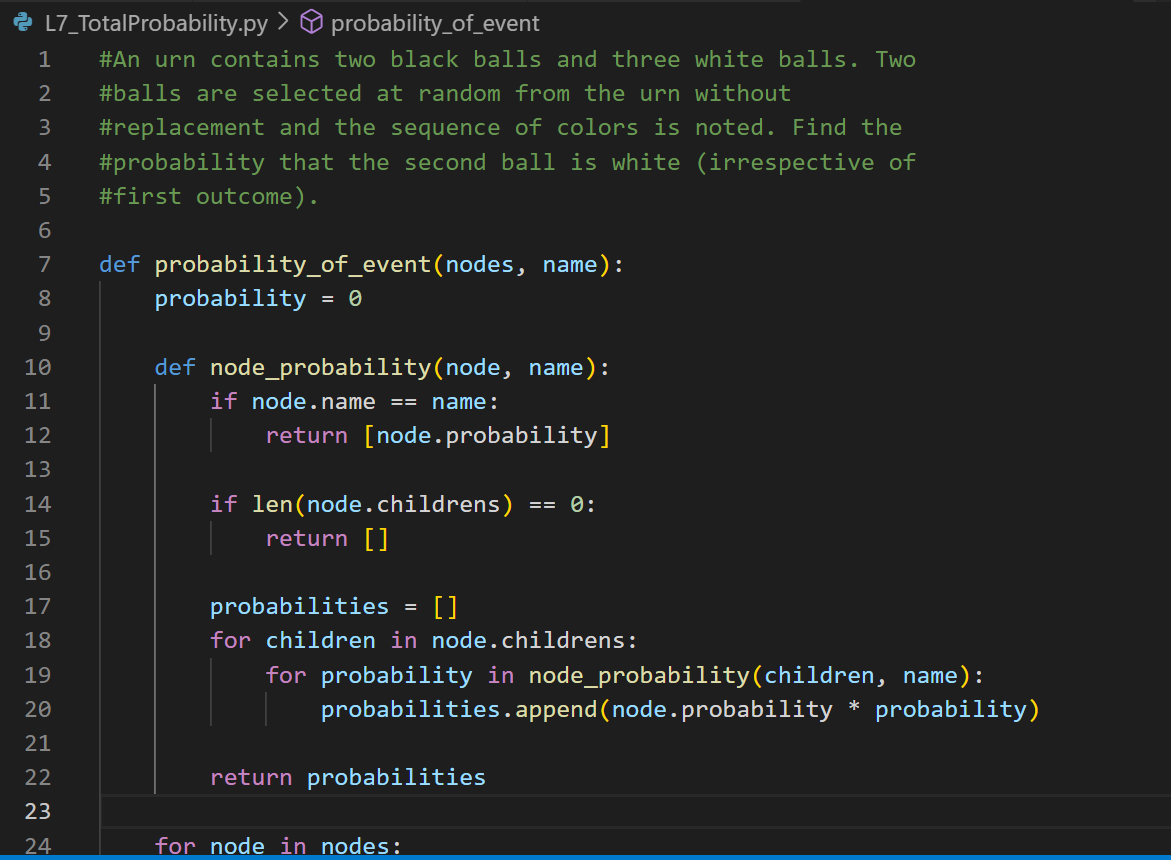
**4. Total Probability:**

The total probability theorem allows us to calculate the probability of an event by considering all possible ways it can occur. In Python, we can define a function to compute the total probability of an event by summing the probabilities of all relevant outcomes.

**Example:**

An urn contains two black balls and three white balls. Two balls are selected at random from the urn without replacement and the sequence of colors is noted. Find the probability that the second ball is white (irrespective of first outcome).

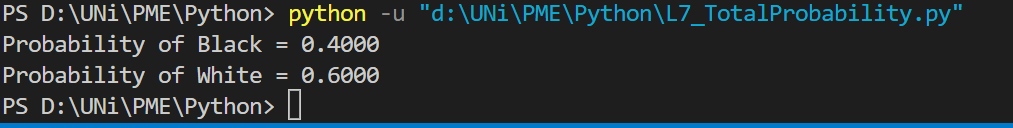
**Code:**



A picture containing text, screenshot, software, display

Description automatically generated

**Output:**



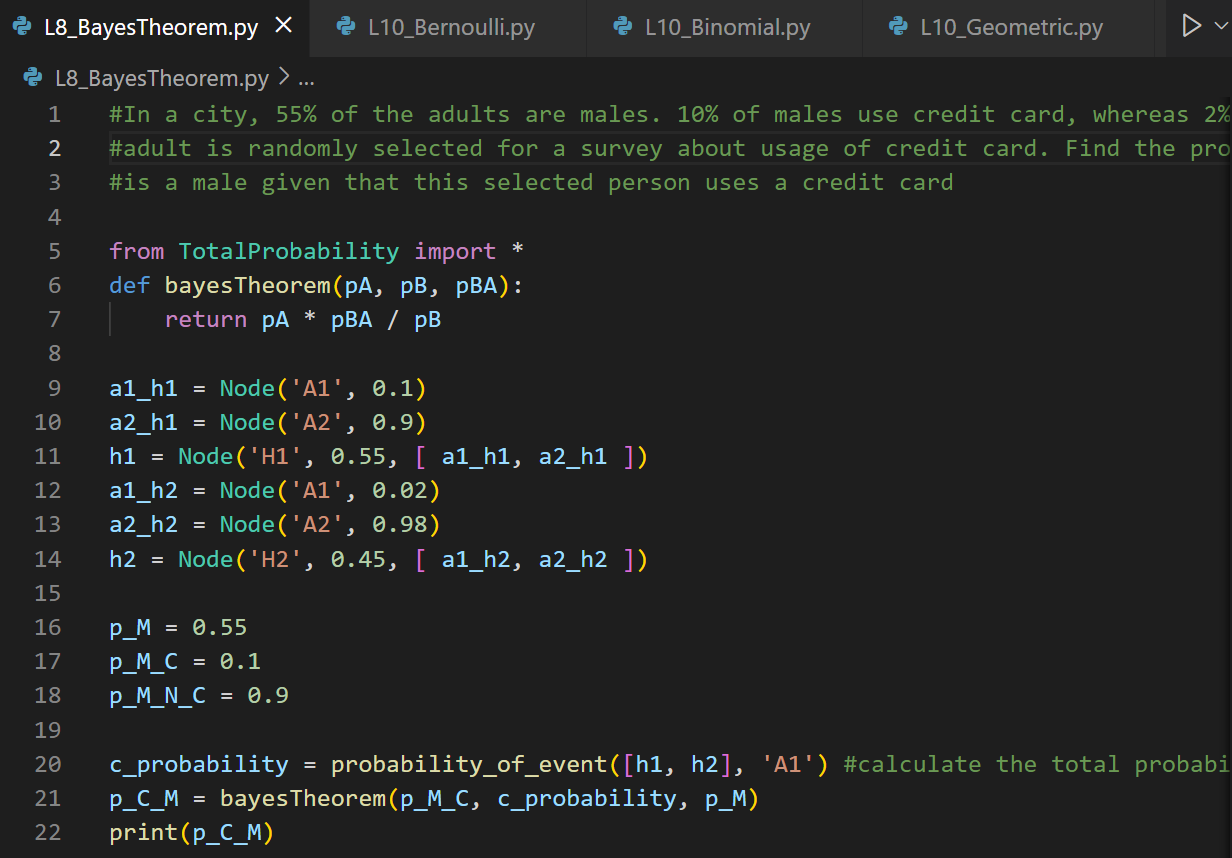
**5. Bayes' Theorem:**

Bayes' theorem provides a way to update the probability of an event based on new evidence or information. It is derived from conditional probability and can be implemented in Python using a function.

**Example:**

In a city, 55% of the adults are males. 10% of males use credit card, whereas 2% of females use credit card. One adult is randomly selected for a survey about usage of credit card. Find the probability that the randomly selected person is a male given that this selected person uses a credit card.

**Code:**



**Output:**

A picture containing text, screenshot, font

Description automatically generated

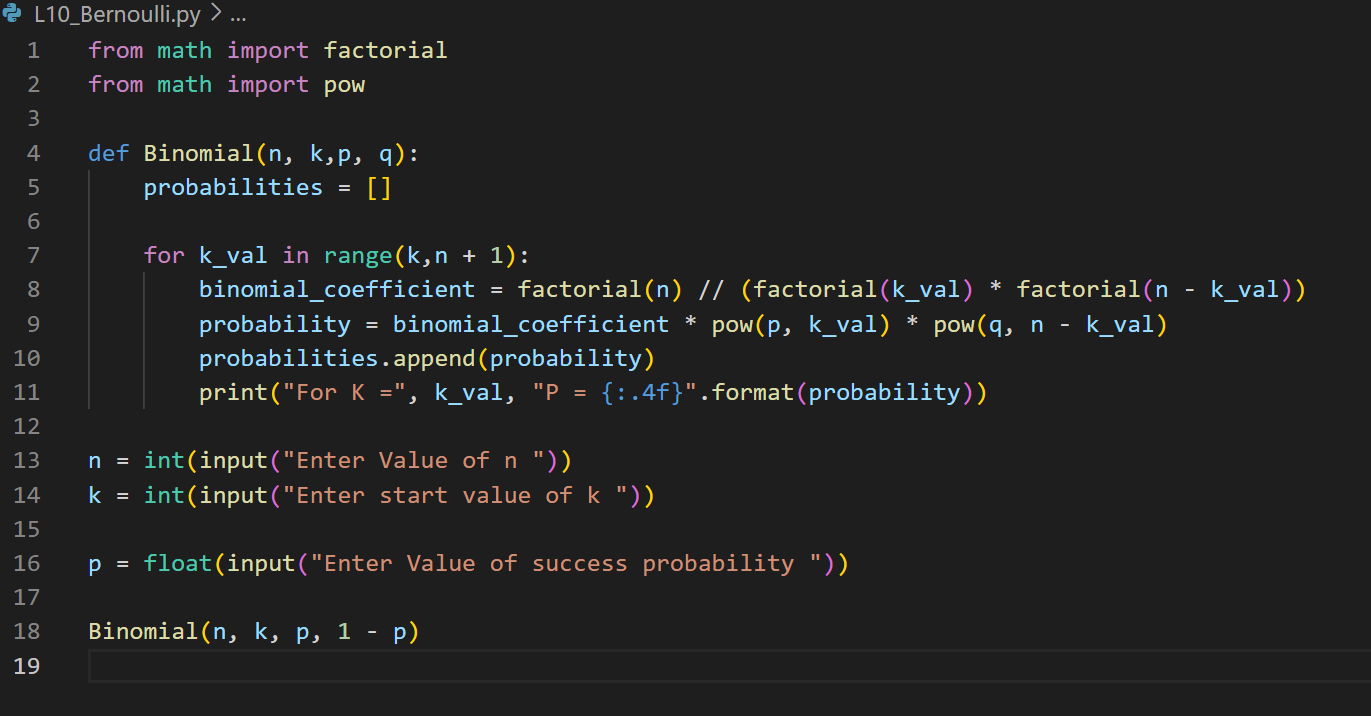
**6. Bernoulli Distribution:**

The Bernoulli distribution models a random experiment with two possible outcomes: success (usually denoted as 1) and failure (usually denoted as 0). It is characterized by a single parameter, p, which represents the probability of success. In Python, we can generate random samples from a Bernoulli distribution using the NumPy library.

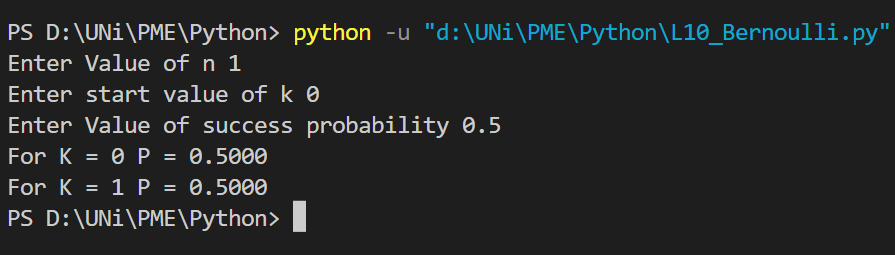
**Example:**

Suppose we want to simulate the outcome of flipping a fair coin, where heads is considered a success. We can use the Bernoulli distribution to model this experiment.

**Code:**



**Output:**

****

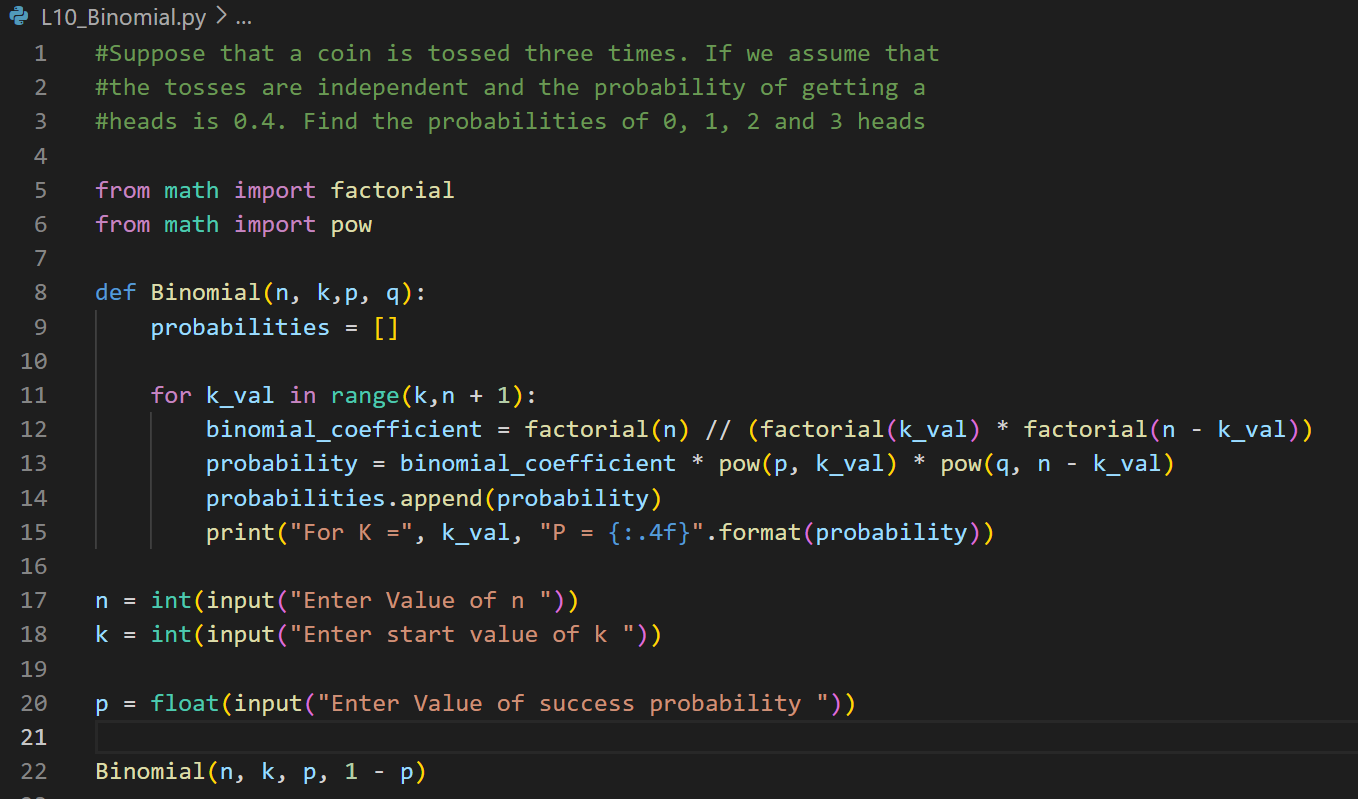
**7. Binomial Distribution:**

The binomial distribution describes the probability of obtaining a certain number of successes in a fixed number of independent Bernoulli trials. In Python, we can calculate the probabilities of different outcomes using the binomial coefficient and the formula for binomial distribution.

**Example:**

Suppose that a coin is tossed three times. If we assume that the tosses are independent and the probability of getting a heads is 0.4. Find the probabilities of 0, 1, 2 and 3 heads

**Code:**



**Output:**

A screen shot of a computer

Description automatically generated with medium confidence

**8. Geometric Distribution:**

The geometric distribution is a probability distribution that models the number of trials needed to achieve the first success in a sequence of independent Bernoulli trials. It is often used to describe situations where a repeated experiment with two possible outcomes (success or failure) is performed until the first success occurs.

**Example:**

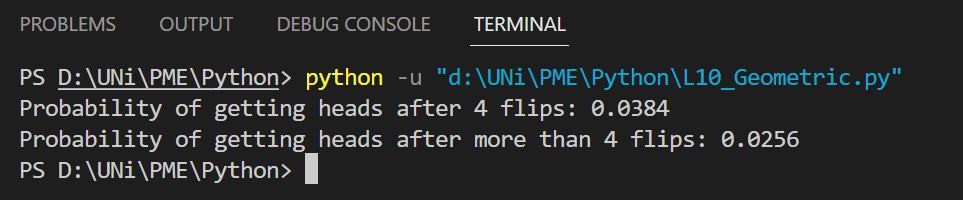
What is the probability that the coin has to be flipped i) 4 times ii) more than 4 times, for getting heads for the first #time? The probability of heads is 0.6 and the probability of tails is 0.4.

**Code:**

**A screen shot of a computer code

Description automatically generated with low confidence**

**Output:**

****