Q1. What is a probability distribution, exactly? If the values are meant to be random, how can you predict them at all?

ANSWER.

A probability distribution is a mathematical function that describes the likelihood of various outcomes or events occurring in a given set of possible outcomes. In simpler terms, it defines the probabilities associated with different values or events within a certain domain.

Probability distributions are used to model uncertainty and randomness in various fields, including statistics, probability theory, machine learning, and simulation. They provide a way to quantify and analyze the uncertainty inherent in real-world phenomena and make predictions about the likelihood of different outcomes.

There are two main types of probability distributions: discrete and continuous.

1. Discrete Probability Distribution: In a discrete probability distribution, the set of possible outcomes is countable and finite or countably infinite. Each outcome has a finite probability associated with it. Examples of discrete probability distributions include the binomial distribution, Poisson distribution, and geometric distribution.

2. Continuous Probability Distribution: In a continuous probability distribution, the set of possible outcomes is uncountable and infinite. The probabilities are assigned to intervals rather than individual points. Examples of continuous probability distributions include the normal (Gaussian) distribution, exponential distribution, and uniform distribution.

Q2. Is there a distinction between true random numbers and pseudo-random numbers, if there is one? Why are the latter considered “good enough”?

ANSWER.

Yes, there is a distinction between true random numbers and pseudo-random numbers.

While pseudo-random numbers are sufficient for many applications, there are scenarios where true randomness is essential, such as cryptographic applications where security depends on the unpredictability of the random numbers. In such cases, true random number sources, such as hardware random number generators, are used to ensure the highest level of randomness and unpredictability.

Q3. What are the two main factors that influence the behaviour of a "normal" probability distribution?

ANSWER.

The two main factors that influence the behavior of a "normal" probability distribution are the mean (μ) and the standard deviation (σ).

1. Mean (μ): The mean of a normal distribution represents the central tendency or average of the distribution. It is the point around which the data is centered. In a normal distribution, the mean determines the location of the peak or highest point of the distribution. If the mean shifts to the right, the distribution shifts to the right as well, and if the mean shifts to the left, the distribution shifts to the left.

2. Standard Deviation (σ): The standard deviation of a normal distribution measures the spread or dispersion of the data around the mean. It quantifies the average distance of data points from the mean. A larger standard deviation indicates greater variability or spread of the data, resulting in a wider and flatter distribution. Conversely, a smaller standard deviation indicates less variability and results in a narrower and taller distribution.

Q4. Provide a real-life example of a normal distribution.

ANSWER.

A real-life example of a normal distribution is human height.

In many populations, human height follows a normal distribution, with the majority of individuals clustered around the average height, and fewer individuals at the extremes of shortness and tallness. This distribution is often represented by a bell-shaped curve, with the mean height at the center of the curve.

Q5. In the short term, how can you expect a probability distribution to behave? What do you think will happen as the number of trials grows?

ANSWER.

In the short term, a probability distribution may exhibit randomness and variability, but as the number of trials grows or the sample size increases, the behavior of the distribution tends to stabilize, and the observed outcomes converge towards the expected values predicted by the distribution.

Q6. What kind of object can be shuffled by using random.shuffle?

ANSWER.

`random.shuffle()` operates in place, meaning it modifies the original sequence rather than returning a new shuffled sequence. Additionally, `random.shuffle()` only shuffles the elements within the sequence; it does not create a new sequence with shuffled elements. Therefore, the object must be mutable for the shuffling operation to take place.

Q7. Describe the math package's general categories of functions.

ANSWER.

The `math` module in Python provides a wide range of mathematical functions for performing various mathematical operations. These functions can be broadly categorized into several general categories based on their functionalities:

1. Basic Arithmetic Functions: This category includes functions for basic arithmetic operations such as addition, subtraction, multiplication, division, exponentiation, and square root. Examples include `math.add()`, `math.subtract()`, `math.multiply()`, `math.divide()`, `math.pow()`, and `math.sqrt()`.

2. Trigonometric Functions: Trigonometric functions operate on angles and include functions such as sine, cosine, tangent, inverse sine, inverse cosine, and inverse tangent. Examples include `math.sin()`, `math.cos()`, `math.tan()`, `math.asin()`, `math.acos()`, and `math.atan()`.

3. Hyperbolic Functions:Hyperbolic functions are analogs of trigonometric functions for hyperbolas and include functions such as hyperbolic sine, hyperbolic cosine, and hyperbolic tangent. Examples include `math.sinh()`, `math.cosh()`, and `math.tanh()`.

4.Exponential and Logarithmic Functions: This category includes functions for exponentiation and logarithms. Exponential functions raise a given base to a specified power, while logarithmic functions compute the logarithm of a number with respect to a given base. Examples include `math.exp()`, `math.log()`, `math.log10()`, and `math.log2()`.

5. Constants: The `math` module also provides several mathematical constants, such as π (pi), Euler's number (e), and the value of the mathematical constant for the base of natural logarithms (math.e).

6. Rounding and Absolute Value Functions:Functions in this category include those for rounding numbers to the nearest integer, obtaining the absolute value of a number, and truncating fractional parts. Examples include `math.ceil()`, `math.floor()`, and `math.fabs()`.

7. Special Functions: Special mathematical functions include factorial, gamma function, error function, and beta function. Examples include `math.factorial()`, `math.gamma()`, `math.erf()`, and `math.beta()`.

Q8. What is the relationship between exponentiation and logarithms?

ANSWER.

Exponentiation and logarithms are inverse operations of each other and are closely related mathematical concepts. Understanding their relationship can provide insights into solving exponential and logarithmic equations and performing various mathematical operations.

Q9. What are the three logarithmic functions that Python supports?

ANSWER.

In Python's `math` module, there are three main logarithmic functions that are commonly used:

1. Natural Logarithm (ln): The natural logarithm function calculates the logarithm of a number to the base \(e\), where \(e\) is Euler's number (approximately equal to 2.71828). In Python, the natural logarithm function is represented as `math.log()`.

```python

import math

x = 10

natural\_log = math.log(x)

print("Natural Logarithm of", x, "is:", natural\_log)

```

2. Common Logarithm (log base 10):The common logarithm function calculates the logarithm of a number to the base 10. In Python, the common logarithm function is represented as `math.log10()`.

```python

import math

x = 100

common\_log = math.log10(x)

print("Common Logarithm of", x, "is:", common\_log)

```

3. Custom Base Logarithm:Python also supports logarithms to any arbitrary base. The base of the logarithm is specified as the second argument to the `math.log()` function.

```python

import math

x = 8

base = 2

custom\_log = math.log(x, base)

print("Logarithm of", x, "with base", base, "is:", custom\_log)

```