All Key Functions in Python's random Module (Detailed Table)

random()	Returns a floating point number between 0.0 (inclusive) and 1.0 (exclusive), following a uniform distribution. It is the core function for generating basic random numbers in Python and forms the basis for many other random operations.	Probability simulations, basic randomness, scaling random values to different ranges.
randint(a, b)	Returns a random integer N such that N is greater than or equal to a and less than or equal to b. Both end-points are included, which makes it convenient for tasks like dice rolls or selecting random indexes from a list.	Creating random IDs, simulating dice, generating random passwords, games, and more.
uniform(a, b)	Generates a floating point number that is uniformly distributed between a and b, suitable for simulating real-world measurements or any scenario needing a continuous range.	Random temperatures, speed, physics variables, simulations, and custom ranges.
choice(seq)	Picks and returns a single random item from a non-empty sequence (like a list or tuple). Useful for randomly selecting from a group of options or entities.	Selecting a random winner, random enemy in games, picking a task, option selection.
shuffle(seq)	Randomly rearranges the items of a mutable sequence (such as a list) in place, producing a new order each time it’s called. The original sequence gets modified.	Shuffling decks of cards, randomizing quiz questions, mixing teams or players.
sample(seq, k)	Returns a new list containing k unique elements selected randomly from the given sequence. Items are picked without replacement, ensuring no repeats.	Drawing lottery numbers, selecting participants, randomized study samples or control groups.
choices(seq, weights=None, k=...)	Returns a list of k elements randomly selected from the population, with replacement. Optional weights can influence selection probability.	Weighted sampling, loot drops, quiz answers with bias, any scenario allowing duplicates.
randrange(start, stop[, step])	Returns a randomly selected element from a specified range built with start, stop, and optional step, similar to the built-in range() function.	Picking random even or odd numbers, selecting values in patterns or custom intervals.
seed(a=None)	Initializes the random number generator to a deterministic state based on the provided seed. This allows for reproducible random sequences, crucial for debugging and testing.	Reproducible experiments, consistent test results, research needing repeatable randomness.
gauss(mu, sigma)	Produces a random number using a Gaussian (normal) distribution, with mu as the mean and sigma as the standard deviation. Results are distributed in a bell curve centered on the mean.	Simulating measurement error, human response times, behavioral modeling.
normalvariate(mu, sigma)	Like gauss(), this draws a value from a normal distribution using the given mean and standard deviation, but with a different underlying algorithm.	Statistical simulations, financial analysis, scientific studies.
lognormvariate(mu, sigma)	Generates a random number from a log-normal distribution, where the logarithm of the result is normally distributed. Defined by a mean and standard deviation for the natural log values.	Stock price evolution models, scientific phenomena with multiplicative factors.
triangular(low, high, mode)	Returns a random floating-point number based on the triangular distribution, bounded by given lower and upper limits, with a peak (mode) determining the most likely value.	Modeling game loot, simulated traffic, rewards distribution with a likely outcome.
expovariate(lambd)	Returns a value from an exponential distribution with a given rate (lambd). This is often used for modeling the time until the next event in a sequence of independent events.	Modeling time between arrivals/events, server hit simulation, decay processes.
betavariate(alpha, beta)	Draws a number from the beta distribution within [0.0, 1.0], shaped by two parameters. Commonly used for modeling and simulating probabilities themselves.	A/B testing simulations, representing likelihoods, stochastic modeling.
gammavariate(alpha, beta)	Returns a value from a gamma distribution (with shape alpha and scale beta), good for modeling scenarios involving waiting times for multiple events.	Advanced modeling, biology, reliability engineering, machine learning.
weibullvariate(alpha, beta)	Returns a number from a Weibull distribution, often used in reliability analyses and survival modeling. The scale (alpha) and shape (beta) parameters control its properties.	Survival analysis, reliability, time-to-failure prediction.
paretovariate(alpha)	Generates a number from a Pareto distribution, characterized by its “long tail”–often used to represent wealth distributions or phenomena following power law.	Wealth simulation, financial modeling, business analytics.
vonmisesvariate(mu, kappa)	Produces a random value from a von Mises (circular) distribution, useful for simulations where data points represent directions or angles, such as compass headings.	Modeling directional data, wind bearings, GPS signal directions, circular statistics.