PassQueen
a.k.a dumpsqueen.com

WGU Foundations of Computer Science

WGU Foundations-of-Computer-Science

Version Demo

Total Demo Questions: 7

Total Premium Questions: 70

Buy Premium PDF

https://passqueen.com

support@passqueen.com

QUESTION NO: 1

What is the built-in data structure that implements a hash table in Python?

A. Tuple

B. List

C. Array

D. Dictionary

ANSWER: D

Explanation:

Dictionary is correct because Python's built-in `dict` type is a mapping that stores associations between unique, hashable keys and corresponding values. Internally, dictionaries use a hash-table-based implementation: Python calculates a key's hash value to efficiently locate the position where its associated value is stored. This supports fast expected lookup, insertion, update, and deletion by key, commonly described as average-case `O(1)` performance. For example, a dictionary such as `{"course": "Foundations of Computer Science", "credits": 3}` permits direct retrieval through a key, as in `d["course"]`. Python requires dictionary keys to be hashable, meaning their hash value remains stable while they are used as keys; common key types include strings, integers, and tuples containing only hashable values. This behavior makes dictionaries appropriate for tasks such as counting occurrences, representing records, caching previously computed results, and creating adjacency mappings in graph algorithms. The Python documentation defines dictionaries as mappings indexed by keys and notes that keys must be immutable types or otherwise hashable. See the Python documentation for dictionary mapping types and the Python glossary definition of hashable objects.

QUESTION NO: 2

Which order is impossible when traversing a binary tree using depth first search?

A. Level-order traversal

B. Pre-order traversal

C. Post-order traversal

D. In-order traversal

ANSWER: A

Explanation:

Level-order traversal is the traversal order that cannot be produced by the standard depth-first-search traversal patterns for a binary tree. Depth-first search follows one branch downward as far as possible before returning, or backtracking, to visit another branch. The common binary-tree depth-first orders are pre-order, in-order, and post-order; each is defined by when the current node is processed relative to recursively visiting its left and right subtrees. Level-order traversal instead processes nodes by their distance from the root: first the root, then every node on the next level, then every node on the following level, and so on. That breadth-by-breadth processing model is breadth-first search behavior and is normally implemented with a queue that holds nodes awaiting processing. DFS is typically implemented recursively or with a stack, reflecting its branch-first behavior. Therefore, when the question uses the conventional definitions of binary-tree traversals, level-order traversal belongs to breadth-first traversal rather than depth-first traversal. See the NIST Dictionary of Algorithms and Data Structures definition of depth-first traversal and the corresponding breadth-first traversal definition.

QUESTION NO: 3

What is the time complexity of a quicksort algorithm?

A. O(n)

B. O(n log n)

C. O(log n)

D. O(1)

ANSWER: B

Explanation:

O(n log n) is the expected, or average-case, time complexity of quicksort and is the standard answer when a question asks generally for quicksort's running time without specifically requesting its worst case. Quicksort uses divide and conquer: it partitions a collection around a pivot, then recursively sorts the portions on either side. Partitioning requires work proportional to the number of elements currently being processed. With reasonably balanced partitions on average, the recursion has approximately log n levels, and each level performs a total of n partitioning work. Multiplying these factors gives O(n log n). This efficiency is why quicksort is commonly presented as a fast comparison-based sorting algorithm and why practical implementations use pivot-selection strategies intended to promote balanced partitions. The algorithm's performance analysis distinguishes expected behavior from its special worst-case arrangement, but the expected complexity requested by this broadly phrased coursework-style question is O(n log n). See the Princeton Algorithms quicksort reference and the NIST Dictionary of Algorithms and Data Structures entry for quicksort for descriptions of quicksort and its expected running-time behavior.

QUESTION NO: 4

Given the following code, what is the expected output?

A. [10, 20, 30, 40]

B. [1, 10]

C. [1, 2, 3, 4]

D. array([10, 20, 30, 40])

ANSWER: C

Explanation:

`[1, 2, 3, 4]` is the intended result because indexing a two-dimensional NumPy array with one integer selects the row at that index. NumPy, like Python generally, uses zero-based indexing, so index `0` identifies the first row. For an array whose rows contain `[1, 2, 3, 4]` and `[10, 20, 30, 40]`, the expression `np_2d[0]` therefore produces the first row's values. The resulting object is a one-dimensional NumPy array containing those four integers. When directly printed by Python, NumPy normally renders an integer array without commas, such as `[1 2 3 4]`; the answer's comma-separated notation still correctly represents the selected values conceptually. This behavior follows NumPy's standard basic-indexing rule: supplying a single index for a two-dimensional array indexes its first axis, which corresponds to rows. NumPy documents this indexing model and illustrates that a single integer index selects a subarray along the first dimension. See the NumPy indexing guide and the ndarray reference.

QUESTION NO: 5

What are Python functions that belong to specific Python objects?

A. Modules

B. Methods

C. Scripts

D. Libraries

ANSWER: B

Explanation:

Methods are functions associated with an object or a class. They define behavior that an object can perform and are normally invoked through dot notation on a particular object. For example, calling `"hello".upper()` invokes the `upper()` method associated with that string object, while `items.append(value)` invokes a behavior provided by a list object. In a user-defined Python class, an instance method is defined in the class and conventionally receives the instance as its first parameter, named `self`. When the method is called through an instance, Python supplies that instance automatically, allowing the method to access or update that object's state. This association between data and behavior is a core object-oriented programming concept: an object exposes operations through methods while maintaining its own data. Python's tutorial describes methods as functions that "belong to" objects and notes that different object types support different methods. See the Python Tutorial: Data Structures and the Python Tutorial: Classes for examples of built-in and user-defined methods.

QUESTION NO: 6

How is the NumPy package imported into a Python session?

A. import num_py

B. import numpy as np

C. using numpy

D. include numpy

ANSWER: B

Explanation:

`import numpy as np` is the conventional Python statement for loading the NumPy package and assigning it the short alias `np`. After this statement runs, NumPy functionality is available through that alias, such as `np.array([1, 2, 3])` to create an array or `np.mean(values)` to calculate an average. Python's import syntax permits an optional `as` clause to bind an imported module to a chosen local name. Although an alias is not strictly required—`import numpy` is also valid—the `np` alias is the established NumPy community convention and is used throughout NumPy documentation, tutorials, scientific Python projects, and many data-analysis codebases. Using this standard spelling improves readability because other Python users immediately recognize `np` as NumPy. The NumPy documentation's quickstart demonstrates importing the library in precisely this form before creating and operating on arrays. See the NumPy Quickstart and the Python documentation on the import statement.

QUESTION NO: 7

How does the data type of a variable get set in Python?

A. It is explicitly declared by the programmer.

B. It is chosen randomly.

C. It is always set to string by default.

D. It is determined by the value assigned to it.

ANSWER: D

Explanation:

It is determined by the value assigned to it. Python is dynamically typed: a name becomes associated with an object when an assignment statement executes, and the object has a type. For example, assigning `x = 42` binds `x` to an integer object, while a later assignment such as `x = "hello"` binds that same name to a string object. The programmer normally does not need to declare a variable's type before assigning a value. This behavior lets code work naturally with objects of different types while the Python runtime tracks the type of each object. Type annotations, such as `count: int = 42`, can document intended types and support static-analysis tools, but they do not change Python's fundamental runtime behavior of determining the object type from the assigned value. The official Python documentation describes assignment as binding names to objects and explains that every object has a type. See the Python assignment-statement reference and the documentation for type().