Certified Associate in Python Programming (PCAP) Practice Exam (Sample)

Study Guide

Everything you need from our exam experts!

Copyright © 2025 by Examzify - A Kaluba Technologies Inc. product.

ALL RIGHTS RESERVED.

No part of this book may be reproduced or transferred in any form or by any means, graphic, electronic, or mechanical, including photocopying, recording, web distribution, taping, or by any information storage retrieval system, without the written permission of the author.

Notice: Examzify makes every reasonable effort to obtain from reliable sources accurate, complete, and timely information about this product.

SAMPLE

Questions

SAMPLE

1. What will occur if you replace the print statement using an alias with the original function name?

A. The program will run without errors.

B. The program will cause a runtime exception.

C. It will print the value assigned to the alias.

D. It will print the value of the original function.

2. How can you iterate over a dictionary in Python?

A. By using a for loop with the .items() method

B. By using the list() function on the dictionary

C. By directly accessing keys using an index

D. By importing the itertools module

3. What does a child class do in the context of inheritance?

A. It overrides all methods of the parent class

B. It can extend or modify behaviors of the parent class

C. It cannot access parent class attributes

D. It can only inherit properties, not methods

4. Which method would you use to find the number of items in a list?

A. count()

B. len()

C. size()

D. length()

5. What is the primary function of 'a' mode when opening a file?

A. To create a new file for writing

B. To read data from the file

C. To write data at the end of the file without truncating

D. To open the file for reading and writing

SAMPLE

6. When importing with an alias, what will happen if you use the alias to access the original function?

- **A. The original function will be accessible through the alias only.**
- **B. The program will cause a NameError.**
- **C. The original function can no longer be accessed by its name.**
- **D. The alias can still refer to the original function without issues.**

7. What is the purpose of the Python `self` keyword in class methods?

- **A. It refers to the instance of the class**
- **B. It defines the class itself**
- **C. It initializes class attributes**
- **D. It is used to call class methods**

8. What is the difference between a shallow copy and a deep copy?

- **A. A shallow copy creates a new object with references to the original elements**
- **B. A deep copy copies the object without any references**
- **C. A shallow copy duplicates nested objects**
- **D. A deep copy only copies basic data types**

9. Which of the following is not a method for creating strings in Python?

- **A. Using single quotes**
- **B. Using double quotes**
- **C. Using the `join()` method**
- **D. Using the `array()` method**

10. What does 'w+' mode allow when opening a file?

- **A. It only allows reading**
- **B. It allows writing without creating a new file**
- **C. It opens for reading and writing, truncating the file**
- **D. It appends data to the end of the file**

SAMPLE

Answers

SAMPLE

1. B
2. A
3. B
4. B
5. C
6. C
7. A
8. A
9. D
10. C

SAMPLE

Explanations

SAMPLE

1. What will occur if you replace the print statement using an alias with the original function name?

A. The program will run without errors.

B. The program will cause a runtime exception.

C. It will print the value assigned to the alias.

D. It will print the value of the original function.

If you replace the print statement that uses an alias with the original function name, the outcome will depend on the definition and context of how the alias was created. If the alias has been appropriately assigned to refer to the `print` function, using the original function name may lead to confusion, particularly if the original function name is no longer bound to the built-in function it was referring to. In a scenario where the alias is created, for example, via an assignment such as `print_alias = print`, and then you replace `print_alias("Hello")` with `print("Hello")`, if there is no reassignment or if `print` has been accidentally redefined as something else in the course of the program, trying to use `print` might cause a runtime exception if `print` is perceived as something other than the expected print function. Thus, this scenario highlights the importance of understanding namespaces and function bindings in Python. If `print` has been overridden or not correctly referenced after altering the alias, a runtime exception will occur due to the function name not pointing to the correct function definition.

2. How can you iterate over a dictionary in Python?

A. By using a for loop with the .items() method

B. By using the list() function on the dictionary

C. By directly accessing keys using an index

D. By importing the itertools module

Iterating over a dictionary in Python can be effectively accomplished using a for loop combined with the .items() method. This method allows you to access both keys and values simultaneously in each iteration. When you call `.items()` on a dictionary, it returns a view object that displays a list of a dictionary's key-value tuple pairs. This is particularly useful when you need to work with both pieces of data at the same time, enhancing readability and efficiency in your code. For example, if you have a dictionary `my_dict = {'a': 1, 'b': 2, 'c': 3}`, using a loop like `for key, value in my_dict.items():` gives you direct access to each key and its corresponding value during each iteration. Other approaches mentioned in the options do not effectively provide a complete or straightforward way to iterate over a dictionary. Using the list() function on the dictionary will only give you a list of keys, which does not allow for access to the values. Directly accessing keys using an index is not possible with dictionaries, as they are not ordered collections like lists. Finally, while the itertools module can provide additional tools for iteration, it is not a required step for simply iterating over a dictionary.

SAMPLE

3. What does a child class do in the context of inheritance?

A. It overrides all methods of the parent class

B. It can extend or modify behaviors of the parent class

C. It cannot access parent class attributes

D. It can only inherit properties, not methods

In the context of inheritance, a child class is designed to build upon the functionality of a parent class. When a child class extends or modifies the behaviors of the parent class, it leverages the concepts of inheritance to both inherit attributes and methods and to introduce or customize functionality that is specific to the child class. This means it can use existing methods from the parent while also modifying them or adding new methods and attributes, thus enhancing or altering the behavior observed in the parent class. This capability allows for the creation of a more specialized class that retains the characteristics of its predecessor while offering additional features or adjustments. This is a fundamental aspect of object-oriented programming, enabling code reuse and promoting a hierarchical structure that is both organized and efficient. In contrast, the other options inaccurately describe the relationship between child and parent classes rather than capturing the essence of how inheritance is intended to work.

4. Which method would you use to find the number of items in a list?

A. count()

B. len()

C. size()

D. length()

To find the number of items in a list in Python, the `len()` function is the appropriate method to use. This built-in function takes an iterable, such as a list, and returns the total count of its elements. For example, if you have a list defined as `my_list = [1, 2, 3, 4]`, calling `len(my_list)` would return `4`, indicating there are four items in the list. Other options like `count()`, `size()`, and `length()` are not valid methods for finding the total number of items in a list. While `count()` can be used to count occurrences of a specific element within the list (for example, `my_list.count(1)` would return `1` if `1` is present in `my_list`), it does not provide the overall count of items. The methods `size()` and `length()` do not exist as built-in functions or methods for the list type in Python, making them invalid choices for this task. Thus, `len()` is the correct and most efficient way to determine the number of items in a list.

SAMPLE

5. What is the primary function of 'a' mode when opening a file?

A. To create a new file for writing

B. To read data from the file

C. To write data at the end of the file without truncating

D. To open the file for reading and writing

The 'a' mode, when opening a file in Python, is specifically designed for appending data. This means that when a file is opened in 'a' mode, any data written to the file is added to the end of the existing content, rather than replacing it. This is beneficial when you want to preserve the current data within the file and simply add more information to it, such as new entries in a log or additional records in a dataset. Using 'a' mode ensures that the original content of the file remains intact and that your new data is seamlessly added. This makes it a straightforward option for situations where data needs to be accumulated over time without the risk of loss from previous entries or data truncation.

6. When importing with an alias, what will happen if you use the alias to access the original function?

A. The original function will be accessible through the alias only.

B. The program will cause a NameError.

C. The original function can no longer be accessed by its name.

D. The alias can still refer to the original function without issues.

When you import a module using an alias, you create a new name for that module, which allows you to refer to it using the alias. However, the original function can still be accessed by its name if it is properly imported. Using an alias does not change the accessibility of the original name; it simply provides a shorthand way to reference the module or its components. Therefore, even after defining an alias, you can still use the original function's name unless you have overwritten that name in your current namespace with a new assignment. In this case, since the original function retains its name in the namespace and can be accessed without issues, the assertion that the original function can no longer be accessed by its name is incorrect.

SAMPLE

7. What is the purpose of the Python `self` keyword in class methods?

A. It refers to the instance of the class

B. It defines the class itself

C. It initializes class attributes

D. It is used to call class methods

The `self` keyword in Python is crucial because it refers to the instance of the class in which a method is being called. This allows methods to access attributes and other methods associated with the specific instance of the class. When you define a method within a class, the first parameter traditionally named `self` acts as a reference to the current object - that is, the instance that invokes the method. For example, if you have a class `Car` and an instance of that class called `my_car`, when you call a method like `my_car.start_engine()`, within that method, `self` allows you to access properties like `self.color` or methods like `self.turn()`, providing a way to manipulate or retrieve instance-specific data. In summary, `self` is essential for instance-level access to attributes and methods, ensuring that the behavior of the methods operates on the data specific to the object instance creating a clear distinction between class-level behavior and instance-level behavior.

8. What is the difference between a shallow copy and a deep copy?

A. A shallow copy creates a new object with references to the original elements

B. A deep copy copies the object without any references

C. A shallow copy duplicates nested objects

D. A deep copy only copies basic data types

A shallow copy creates a new object, but instead of copying the elements themselves, it only copies references to those elements from the original object. This means that if the object contains nested structures like lists or dictionaries, the shallow copy reflects changes made to those nested structures since both the original and the shallow copy point to the same objects in memory. In contrast, a deep copy duplicates both the object and all objects nested within it, creating entirely new instances of the nested objects. This ensures that modifications to the copied object do not affect the original object. The statement about a shallow copy refers specifically to the copying mechanism—making it essential for understanding how changes to the original and copied objects can influence each other. The other choices incorrectly describe the behaviors of shallow and deep copies. For instance, deep copies do copy nested objects, and they do so in a manner that they are independent of the original objects. Furthermore, the notion that deep copies only copy basic data types misrepresents what a deep copy does, as it applies to all nested objects regardless of their type.

SAMPLE

9. Which of the following is not a method for creating strings in Python?

A. Using single quotes

B. Using double quotes

C. Using the `join()` method

D. Using the `array()` method

Creating strings in Python can be accomplished in several ways, each suited for different needs. The options that involve using both single quotes and double quotes are standard methods for defining string literals. Whether you wrap characters in single or double quotes, Python treats them equivalently, allowing flexibility for string creation. The `join()` method is a string method that combines elements from an iterable (like a list or a tuple) into a single string, using the string it's called on as a separator. For example, if you use `",".join(['a', 'b', 'c'])`, it produces the string `"a,b,c"`. This method is indeed a valid approach to create strings from lists of characters or strings. In contrast, the `array()` method is not a built-in method for creating strings in Python. The `array()` function typically relates to creating an array of numeric data, which is part of the `array` module in Python, but it does not create strings. Therefore, this option is not regarded as a method for string creation, making it the correct choice for this question.

10. What does 'w+' mode allow when opening a file?

A. It only allows reading

B. It allows writing without creating a new file

C. It opens for reading and writing, truncating the file

D. It appends data to the end of the file

The 'w+' mode when opening a file allows for both reading and writing, and importantly, it truncates the file upon opening. This means that if the file already exists, its contents will be deleted immediately, and the file will be treated as an empty file for subsequent read or write operations. This mode is particularly useful when you want to create a file and start fresh with writing new content while also having the ability to read from it during the same session. The truncation aspect ensures that any previous data is lost, allowing for the new content to take precedence. In contrast, other modes like 'r+' allow reading and writing without truncating the existing content, while 'a' or 'a+' would simply append new data without modifying existing content. Therefore, 'w+' distinctly provides the feature of truncation alongside read and write capabilities in a single operation.

SAMPLE