



Up-to-date Questions and Answers from authentic resources to improve knowledge and pass the exam at very first attempt. ----- Guaranteed.



PCPP-32-101 MCQs
PCPP-32-101 TestPrep
PCPP-32-101 Study Guide
PCPP-32-101 Practice Test
PCPP-32-101 Exam Questions



Python

PCPP-32-101

PCPP1-Certified Professional in Python Programming
1



Question: 175

Analyze the code and choose the best statement that describes it.

```
class Item:
    def __init__(self, initial_value):
        self.value = initial_value

    def __ne__(self, other):
        ...
```

- A. `__ne__()` is not a built-in special method
- B. The code is erroneous
- C. The code is responsible for the support of the negation operator e.g. `a = - a`.
- D. The code is responsible for the support of the inequality operator i.e. `i =`

Answer: D

Explanation:

The correct answer is

D. The code is responsible for the support of the inequality operator i.e. `i != j`. In the given code snippet, the `__ne__` method is a special method that overrides the behavior of the inequality operator `!=` for instances of the `MyClass` class. When the inequality operator is used to compare two instances of `MyClass`, the `__ne__` method is called to determine whether the two instances are unequal.

Question: 176

Analyze the following snippet and select the statement that best describes it.

```
class OwnMath:
    pass

def calculate_value(numerator, denominator):
    try:
        value = numerator / denominator
    except ZeroDivisionError as e:
        raise OwnMath from e
    return value

calculate_value(4, 0)
```

- A. The code is an example of implicitly chained exceptions.
- B. The code is erroneous as the OwnMath class does not inherit from any Exception type class
- C. The code is fine and the script execution is not interrupted by any exception.
- D. The code is an example of explicitly chained exceptions.

Answer: D

Explanation:

In the given code snippet, an instance of OwnMath exception is raised with an explicitly specified `__cause__` attribute that refers to the original exception (ZeroDivisionError). This is an example of explicitly chaining exceptions in Python.

Question: 177

Analyze the following snippet and decide whether the code is correct and/or which method should be distinguished as a class method.

```

class Crossword:
    number_of_Crosswords = 0

    def __init__(self, height, width):
        self.height = height
        self.width = width
        self.progress = 0

    @staticmethod
    def isElementCorrect(word):
        if self.isSolved():
            print('The crossword is already solved')
            return True
        result = True
        for char in word:
            if char.isdigit():
                result = False
                break
        return result

    def isSolved(self):
        if self.progress == 100:
            return True
        return False

    def getNumberOfCrosswords(cls):
        return cls.number_of_Crosswords

```

- A. There is only one initializer, so there is no need for a class method.
- B. The getNumberOfCrosswords () method should be decorated With @classmethod.
- C. The code is erroneous.
- D. The gexNumberOfcrosswords () and issrived methods should be decorated with @classzoechod.

Answer: B

Explanation:

The correct answer is B. The getNumberOfCrosswords() method should be decorated with @classmethod. In the given code snippet, the getNumberOfCrosswords method is intended to be a class method that returns the value of the numberofcrosswords class variable. However, the method is not decorated with the @classmethod decorator and does not take a cls parameter representing the class itself. To make getNumberOfCrosswords a proper class method, it should be decorated with @classmethod and take a cls parameter as its first argument.

B. The getNumberOfCrosswords() method should be decorated with @classmethod. This is because the getNumberOfCrosswords() method is intended to access the class-level variable numberofcrosswords, but it is defined as an instance method, which requires an instance of the class to be created before it can be called. To make it work as

a class-level method, you can define it as a class method by adding the `@classmethod` decorator to the function.

Here's an example of how to define `getNumberOfCrosswords()` as a class method:

```
classCrossword:

    numberofcrosswords =0

    def __init__(self, author, title):

        self.author = author

        self.title = title

    Crossword.numberofcrosswords +=1

    @classmethod

    defgetNumberOfCrosswords(cls):

        returncls.numberofcrosswords
```

In this example, `getNumberOfCrosswords()` is defined as a class method using the `@classmethod` decorator, and the `cls` parameter is used to access the class-level variable `numberofcrosswords`.

Reference: Official Python documentation on Classes: <https://docs.python.org/3/tutorial/classes.html>

Question: 178

Select the true statements about the `sqlite3` module. (Select two answers.)

- A. The `fetchall` method returns `None` when no rows are available
- B. The `execute` method allows you to perform several queries at once
- C. The `execute` method is provided by the `Cursor` class
- D. The `fetchone` method returns `None` when no rows are available

Answer: A,C,D

Explanation:

- C. The `execute` method is provided by the `Cursor` class

This statement is true because the `execute` method is one of the methods of the `Cursor` class in the `sqlite3` module. The `Cursor` class represents an object that can execute SQL statements and fetch results from a database connection. The `execute` method takes an SQL query as an argument and executes it against the database. For example, `cur = conn.cursor (); cur.execute ("SELECT * FROM table")` creates and executes a cursor object that selects all rows from a table.

- D. The `fetchone` method returns `None` when no rows are available

This statement is true because the `fetchone` method is another method of the `Cursor` class in the `sqlite3` module. The `fetchone` method fetches the next row of a query result set and returns it as a single tuple or `None` if no more rows are available. For example, `row = cur.fetchone ()` fetches and returns one row from the cursor object or `None` if there are

no more rows.

Question: 179

What is true about the invocation of the `cget ()` method?

- A. It can be used to read widget attributes.
- B. It has the same effect as the `config ()` method.
- C. It can be used to set new values to widget attributes.
- D. It can be replaced with a dictionary-like access manner.

Answer: A

Explanation:

The `cget()` method in Python is used to read the configuration options of a widget in Tkinter. It retrieves the value of a specified configuration option for a Tkinter widget. Hence, option A is the correct answer.

Question: 180

In the JSON processing context, the term serialization:

- A. names a process in which Python data is turned into a JSON string.
- B. names a process in which a JSON string is turned into Python data.
- C. refers to nothing, because there is no such thing as JSON serialization.
- D. names a process in which a JSON string is remodeled and transformed into a new JSON string

Answer: A

Explanation:

In the JSON processing context, the term serialization:

- A. names a process in which Python data is turned into a JSON string.

Serialization refers to the process of converting a data object, such as a Python object, into a format that can be easily transferred over a network or stored in a file. In the case of JSON, serialization refers to converting Python data into a string representation using the JSON format. This string can be sent over a network or stored as a file, and later deserialized back into the original Python data object.

Reference: Official Python documentation on json: <https://docs.python.org/3/library/json.html#json-serialization>

Question: 181

What does the term deserialization mean? Select the best answer.

- A. It is a process of creating Python objects based on sequences of bytes.
- B. It is a process of assigning unique identifiers to every newly created Python object
- C. It is another name for the data transmission process
- D. It is a process of converting the structure of an object into a stream of bytes

Answer: A

Explanation:

A. Deserialization is the process of converting data that has been serialized or encoded in a specific format, back into its original form as an object or a data structure in memory. In Python, this typically involves creating Python objects based on sequences of bytes that have been serialized using a protocol such as JSON, Pickle, or YAML.

For example, if you have a Python object `my_obj` and you want to serialize it to a JSON string, you might do something like this:

```
import json
```

```
serialized_obj = json.dumps(my_obj)
```

To deserialize the JSON string back into a Python object, you would use the `json.loads()` method:

```
deserialized_obj = json.loads(serialized_obj)
```

This would convert the JSON string back into its original Python object form.

Reference:

Official Python Documentation on

Serialization: <https://docs.python.org/3/library/pickle.html#module-pickle>

Real Python Tutorial on Serialization and Deserialization in Python: <https://realpython.com/python-serialization/>

Deserialization is the process of converting a sequence of bytes, such as a file or a network message, into a Python object. This is the opposite of serialization, which is the process of converting a Python object into a sequence of bytes for storage or transmission.

Question: 182

Analyze the following snippet and select the statement that best describes it.

```
def f1(*arg, **args):  
    pass
```

- A. The code is syntactically correct despite the fact that the names of the function parameters do not follow the naming convention
- B. The `*arg` parameter holds a list of unnamed parameters
- C. The code is missing a placeholder for unnamed parameters.
- D. The code is syntactically incorrect - the function should be defined as `def f1 (*args, **kwargs) :`

Answer: B

Explanation:

The provided code snippet defines a function `f1` that accepts variable-length arguments using the `*args` and `**kwargs` syntax. The `*args` parameter allows for an arbitrary number of unnamed arguments to be passed to the function as a tuple, while the `**kwargs` parameter allows for an arbitrary number of named arguments to be passed to the function as a dictionary.

Therefore, the correct statement that best describes the code is:

- B. The `*args` parameter holds a list of unnamed parameters, while the `**kwargs` parameter holds a dictionary of named parameters.

Reference:

Official Python documentation on Function definitions: <https://docs.python.org/3/tutorial/controlflow.html#defining-functions>

The `arg` parameter holds a list of unnamed parameters. In the given code snippet, the `f1` function takes two arguments: `*arg` and `**kwarg`. The `*arg` syntax in the function signature is used to pass a variable number of non-keyword (positional) arguments to the function. Inside the function, `arg` is a tuple containing the positional arguments passed to the function. The `**kwarg` syntax in the function signature is used to pass a variable number of keyword arguments to the function. Inside the function, `kwarg` is a dictionary containing the keyword arguments passed to the function.

Question: 183

Which one of the following methods allows you to debug an XML tree in the `xml.etree.ElementTree` module?

- A. `debug`
- B. `dump`
- C. `log`
- D. `parse`

Answer: B

Explanation:

The `dump()` method in the `xml.etree.ElementTree` module allows you to output a debug representation of an XML tree to a file or standard output. This method is useful for analyzing the structure of the tree and tracking down errors.

Reference: Official Python documentation on the `ElementTree` module:
<https://docs.python.org/3/library/xml.etree.elementtree.html>

Question: 184

Which function or operator should you use to obtain the answer True or False to the question: "Do two variables refer to the same object?"

- A. The = operator
- B. The isinstanceO function
- C. The id () function
- D. The is operator

Answer: D

Explanation:

To test whether two variables refer to the same object in memory, you should use the is operator. The is operator returns True if the two variables point to the same object in memory, and False otherwise.

For example:

```
a = [1, 2, 3]
```

```
b = a
```

```
c = [1, 2, 3]
```

```
print(a is b) # True
```

```
print(a is c) # False
```

In this example, a and b refer to the same list object in memory, so a is b returns True. On the other hand, a and c refer to two separate list objects with the same values, so a is c returns False.

Reference:

Official Python documentation on

Comparisons: <https://docs.python.org/3/reference/expressions.html#not-in>

Official Python documentation on Identity

comparisons: <https://docs.python.org/3/reference/expressions.html#is>

The is operator is used to test whether two variables refer to the same object in memory. If two variables x and y refer to the same object, the expression x is y will evaluate to True. Otherwise, it will evaluate to False.

Question: 185

What is true about type in the object-oriented programming sense?

- A. It is the bottommost type that any object can inherit from.
- B. It is a built-in method that allows enumeration of composite objects
- C. It is the topmost type that any class can inherit from

D. It is an object used to instantiate a class

Answer: C

Explanation:

In Python, type is the built-in metaclass that serves as the base class for all new-style classes. All new-style classes in Python, including built-in types like int and str, are instances of the type metaclass and inherit from it.

Question: 186

What will happen if the main window is too small to fit all its widgets?

- A. Some widgets may be invisible
- B. The window will be expanded.
- C. An exception will be raised.
- D. The widgets will be scaled down to fit the window's size.

Answer: A

Explanation:

If the main window is too small to fit all its widgets, some widgets may be invisible. So, the correct answer is Option A.

When a window is not large enough to display all of its content, some widgets may be partially or completely hidden. The window will not automatically expand to fit all of its content, and no exception will be raised. The widgets will not be automatically scaled down to fit the window's size.

If the main window is too small to fit all its widgets, some of the widgets may not be visible or may be partially visible. This is because the main window has a fixed size, and if there are more widgets than can fit within that size, some of them will be outside the visible area of the window.

To avoid this issue, you can use layout managers such as grid, pack, or place to dynamically adjust the size and position of the widgets as the window changes size. This will ensure that all the widgets remain visible and properly arranged regardless of the size of the main window.

References:

• <https://www.tkdcs.com/tutorial/widgets.html#managers>

• <https://www.geeksforgeeks.org/python-tkinter-widgets/>

• <https://anzelg.github.io/rin2/book2/2405/docs/tkinter/introduction.html>

Question: 187

Which of the following will set the button text's font to 12 point italics? (Select two answers)

A)

```
button.ButtonFont('Arial', '12', 'italic')
```

B)

```
button.setFont(('Arial', '12', 'italic'))
```

C)

```
button=Button(wnd, font=('Arial', '12', 'italic'))
```

D)

```
button.config(font=('Arial', '12', 'italic'))
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

Answer: A,B,C

Explanation:

Option B is correct because it sets the font option of the button to a tuple containing the font family (âArialâ), size (12), and style (âitalicâ).

Option C is correct because it sets the font option of the button to a string containing the font family (âArialâ), size (12), and style (âitalicâ) separated by spaces.

Killexams.com is a leading online platform specializing in high-quality certification exam preparation. Offering a robust suite of tools, including MCQs, practice tests, and advanced test engines, Killexams.com empowers candidates to excel in their certification exams. Discover the key features that make Killexams.com the go-to choice for exam success.



Exam Questions:

Killexams.com provides exam questions that are experienced in test centers. These questions are updated regularly to ensure they are up-to-date and relevant to the latest exam syllabus. By studying these questions, candidates can familiarize themselves with the content and format of the real exam.

Exam MCQs:

Killexams.com offers exam MCQs in PDF format. These questions contain a comprehensive collection of questions and answers that cover the exam topics. By using these MCQs, candidate can enhance their knowledge and improve their chances of success in the certification exam.

Practice Test:

Killexams.com provides practice test through their desktop test engine and online test engine. These practice tests simulate the real exam environment and help candidates assess their readiness for the actual exam. The practice test cover a wide range of questions and enable candidates to identify their strengths and weaknesses.

Guaranteed Success:

Killexams.com offers a success guarantee with the exam MCQs. Killexams claim that by using this materials, candidates will pass their exams on the first attempt or they will get refund for the purchase price. This guarantee provides assurance and confidence to individuals preparing for certification exam.

Updated Contents:

Killexams.com regularly updates its question bank of MCQs to ensure that they are current and reflect the latest changes in the exam syllabus. This helps candidates stay up-to-date with the exam content and increases their chances of success.