



# QUESTION & ANSWER

HIGHER QUALITY, BETTER SERVICE

**Provide One Year Free Update!**

<https://www.passquestion.com>

**Exam** : **PCPP-32-101**

**Title** : PCPP1-Certified  
Professional in Python  
Programming 1

**Version** : DEMO

1.Which of the following constants will be used if you do not define the quoting argument in the writer method provided by the csv module?

- A. csv.QUOTE\_MINIMAL
- B. csv.QUOTE\_NONE
- C. csv.QUOTE\_ALL
- D. csv.QUOTE\_NONNUMERIC

**Answer:** A

**Explanation:**

If you do not define the quoting argument in the writer method provided by the csv module, the default quoting behavior is set to QUOTE\_MINIMAL. This means that fields containing special characters such as the delimiter or newline character will be quoted, while fields that do not contain special characters will not be quoted.

Reference: Official Python documentation on the csv module: <https://docs.python.org/3/library/csv.html>

2.What is true about the unbind\_all () method?

(Select two answers.)

- A. It can be invoked from any widget
- B. It can be invoked from the main window widget only
- C. It is parameterless
- D. It causes all the widgets to disappear

**Answer:** A,C

**Explanation:**

The unbind\_all() method in Tkinter is used to remove all event bindings from a widget. It is a method of the widget object and can be called on any widget in the Tkinter application. Therefore, option A is the correct answer.

Option B is incorrect because the method can be called on any widget, not just the main window widget.

Option C is correct as unbind\_all() does not take any parameters.

Option D is incorrect because the method only removes event bindings and does not cause the widgets to disappear.

So, the correct answers are A and C.

References:

⇒ Tkinter documentation: <https://docs.python.org/3/library/tkinter.html#event-bindings>

⇒ Tkinter tutorial: [https://www.python-course.eu/tkinter\\_events\\_binds.php](https://www.python-course.eu/tkinter_events_binds.php)

3.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.  $i \neq j$ . 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.

4. 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.

5. 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>