

Python Institute.Premium.PCPP-32-101.45q - DEMO

Number: PCPP-32-101  
Passing Score: 800  
Time Limit: 120 min  
File Version: 1.7



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

**Exam Name:** PCPP1 - Certified Professional in Python Programming 1

**Website:** [www.VCEplus.io](http://www.VCEplus.io) - [www.VCEup.com](http://www.VCEup.com)

**VCEup**

## Exam A

### QUESTION 1

Select the true statement about composition

- A. Composition extends a class's capabilities by adding new components and modifying the existing ones.
- B. Composition allows a class to be projected as a container of different classes
- C. Composition is a concept that promotes code reusability while inheritance promotes encapsulation.
- D. Composition is based on the has a relation: so it cannot be used together with inheritance.

**Correct Answer: B**

**Section:**

**Explanation:**

Composition is an object-oriented design concept that models a has-a relationship. In composition, a class known as composite contains an object of another class known as component. In other words, a composite class has a component of another class1.

1. Composition allows a class to be projected as a container of different classes.

Composition is a concept in Python that allows for building complex objects out of simpler objects, by aggregating one or more objects of another class as attributes. The objects that are aggregated are generally considered to be parts of the whole object, and the containing object is often viewed as a container for the smaller objects.

In composition, objects are combined in a way that allows for greater flexibility and modifiability than what inheritance can offer. With composition, it is possible to create new objects by combining existing objects, by using a container object to host other objects. By contrast, with inheritance, new objects extend the behavior of their parent classes, and are limited by that inheritance hierarchy.

Reference:

Official Python documentation on

Composition: <https://docs.python.org/3/tutorial/classes.html#composition> article on Composition vs Inheritance: <https://www.geeksforgeeks.org/compositionvs-inheritance-python/Real> Python article on

Composition and Inheritance: <https://realpython.com/inheritancecomposition-python/>

### QUESTION 2

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.

**Correct Answer: D**

**Section:**

**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 3

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

```
class Sword:
    var1 = 'weapon'

    def __init__(self):
        self.name = 'Excalibur'
```

- A. self.name is the name of a class variable.
- B. var1 is the name of a global variable
- C. Excalibur is the value passed to an instance variable
- D. Weapon is the value passed to an instance variable

**Correct Answer: C**

**Section:**

**Explanation:**

The correct answer is C. Excalibur is the value passed to an instance variable. In the given code snippet, self.name is an instance variable of the Sword class. When an instance of the Sword class is created with `var1 = Sword('Excalibur')`, the value 'Excalibur' is passed as an argument to the `__init__` method and assigned to the name instance variable of the var1 object.

The code defines a class called Sword with an `__init__` method that takes one parameter name.

When a new instance of the Sword class is created with `var1 = Sword('Excalibur')`, the value of the 'Excalibur' string is passed as an argument to the `__init__` method, and assigned to the self.name instance variable of the var1 object.

Reference:

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

### QUESTION 4

The following snippet represents one of the OOP pillars Which one is that?

```
class A:
    def run(self):
        print("A is running")

class B:
    def fly(self):
        print("B is flying")

class C:
    def run(self):
        print("C is running")

for element in A(), B(), C():
    element.run()
```

- A. Serialization
- B. Inheritance
- C. Encapsulation
- D. Polymorphism

**Correct Answer: C**

**Section:**

**Explanation:**

The given code snippet demonstrates the concept of encapsulation in object-oriented programming.

Encapsulation refers to the practice of keeping the internal state and behavior of an object hidden from the outside world and providing a public interface for interacting with the object. In the given code snippet, the `__init__` and `get_balance` methods provide a public interface for interacting with instances of the `BankAccount` class, while the `__balance` attribute is kept hidden from the outside world by using a double underscore prefix.

#### QUESTION 5

Analyze the following function and choose the statement that best describes it.

```
def my_decorator(coating):
    def level1_wrapper(my_function):
        def level2_wrapper(*args):
            our_function(*args)

        return level2_wrapper

    return level1_wrapper
```

- A. It is an example of a decorator that accepts its own arguments.
- B. It is an example of decorator stacking.
- C. It is an example of a decorator that can trigger an infinite recursion.
- D. The function is erroneous.

**Correct Answer: A**

**Section:**

**Explanation:**

In the given code snippet, the repeat function is a decorator that takes an argument num\_times specifying the number of times the decorated function should be called.

The repeat function returns an inner function wrapper\_repeat that takes a function func as an argument and returns another inner function wrapper that calls func num\_times times.

The provided code snippet represents an example of a decorator that accepts its own arguments.

The @decorator\_function syntax is used to apply the decorator\_function to the some\_function function. The decorator\_function takes an argument arg1 and defines an inner function wrapper\_function that takes the original function func as its argument.

The wrapper\_function then returns the result of calling func, along with the arg1 argument passed to the decorator\_function.

Here is an example of how to use this decorator with some\_function:

@decorator\_function("argument 1") def some\_function(): return "Hello world" When some\_function is called, it will first be passed as an argument to the decorator\_function.

The decorator\_function then adds the string "argument 1" to the result of calling some\_function() and returns the resulting string. In this case, the final output would be "Hello world argument 1".

Reference:

Official Python documentation on Decorators: <https://docs.python.org/3/glossary.html#termdecorator>

#### QUESTION 6

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.

**Correct Answer: B**

**Section:**

**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:

1. 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 7

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.

**Correct Answer: B**

**Section:**

**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.

1. 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.

numberofcrosswords = 0

```
def __init__(self, author, title):
```

```
self.author = author
```

```
self.title = title
```

```
Crossword.numberofcrosswords += 1
```

```
@classmethod
```

```
def getNumberOfCrosswords(cls):
```

```
    return cls.numberofcrosswords
```

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>

Reference:

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

#### QUESTION 8

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 =

**Correct Answer: D**

**Section:**

**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 9

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 isinstance() function
- C. The id() function

D. The is operator

**Correct Answer: D**

**Section:**

**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#isThe> 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 10

Which sentence about the @property decorator is false?

- A. The @property decorator should be defined after the method that is responsible for setting an encapsulated attribute.
- B. The @property decorator designates a method which is responsible for returning an attribute value
- C. The @property decorator marks the method whose name will be used as the name of the instance attribute
- D. The @property decorator should be defined before the methods that are responsible for setting and deleting an encapsulated attribute

**Correct Answer: A**

**Section:**

**Explanation:**

The @property decorator should be defined after the method that is responsible for setting an encapsulated attribute is a false sentence. In fact, the @property decorator should be defined before the method that is used to set the attribute value. The @property decorator and the setter and deleter methods work together to create an encapsulated attribute, which is used to provide control over the attribute's value.

Reference:

Official Python documentation on

Property: <https://docs.python.org/3/library/functions.html#property>

The @property decorator is used to designate a method as a getter for an instance attribute. The method decorated with @property should be defined before any setter or deleter methods for the same attribute.

#### QUESTION 11

Select the true statement about the \_\_name\_\_ attribute.

- A. \_\_name\_\_ is a special attribute, which is inherent for both classes and instances, and it contains information about the class to which a class instance belongs.
- B. \_\_name\_\_ is a special attribute, which is inherent for both classes and instances, and it contains a dictionary of object attributes.
- C. \_\_name\_\_ is a special attribute, which is inherent for classes and it contains information about the class to which a class instance belongs.
- D. \_\_name\_\_ is a special attribute, which is inherent for classes, and it contains the name of a class.

**Correct Answer: D**

**Section:**

**Explanation:**

The true statement about the \_\_name\_\_ attribute is D. name is a special attribute, which is inherent for classes, and it contains the name of a class. The \_\_name\_\_ attribute is a special attribute of classes that contains the name of the class as a string.

The \_\_name\_\_ attribute is a special attribute in Python that is available for all classes, and it contains the name of the class as a string. The \_\_name\_\_ attribute can be accessed from both the class and its instances using the dot notation.



Reference:

Official Python documentation on Classes: <https://docs.python.org/3/tutorial/classes.html#class-objects>

#### QUESTION 12

What is a static method?

- A. A method that works on the class itself
- B. A method decorated with the @method trait
- C. A method that requires no parameters referring to the class itself
- D. A method that works on class objects that are instantiated

**Correct Answer: C**

**Section:**

**Explanation:**

A static method is a method that belongs to a class rather than an instance of the class. It is defined using the @staticmethod decorator and does not take a self or cls parameter. Static methods are often used to define utility functions that do not depend on the state of an instance or the class itself.

#### QUESTION 13

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

**Correct Answer: C**

**Section:**

**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 14

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

**Correct Answer: A, A, A, A, A, A, A, A, B, C, C, D, D, D, E, E, E, E, E, E, E, E, F, G, H, H, H, I, I, I, I, I, I, I, L, L, N, N, N, N, O, O, O, O, P, R, R, R, R, S, S, S, S, S, S, T, T, T, T, T, T, V, Z, Z**

**Section:**

**Explanation:**

Answer: A. Deserialization is the process of converting data that has been serialized

Explanation: 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> Python Tutorial on Serialization and Deserialization in Python: <https://realpython.com/pythonserialization/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 15

What is a \_\_\_ traceback \_\_\_?

(Select two answers )

- A. An attribute owned by every exception object
- B. A special method delivered by the traceback module to retrieve a full list of strings describing the traceback
- C. An attribute that is added to every object when the traceback module is imported
- D. An attribute that holds interesting information that is particularly useful when the programmer wants to store exception details in other objects

**Correct Answer: A, D**

**Section:**

**Explanation:**

The correct answers are A. An attribute owned by every exception object and D. An attribute that holds interesting information that is particularly useful when the programmer wants to store exception details in other objects. A traceback is an attribute of an exception object that contains a stack trace representing the call stack at the point where the exception was raised. The traceback attribute holds information about the sequence of function calls that led to the exception, which can be useful for debugging and error reporting.

#### QUESTION 16

Which of the following examples using line breaks and different indentation methods are compliant with PEP 8 recommendations? (Select two answers.)

A)

```
spam = my_function(arg_one, arg_two,
                   arg_three, arg_four)
```

B)

```
eggs = (1, 2, 3,
        4, 5, 6)
```

C)

```
my_list = [
    1, 2, 3,
    4, 5, 6,
]
```

D)

```
foo = my_function(
    arg_one, arg_two,
    arg_three, arg_four
)
```

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

**Correct Answer: B, D**

**Section:**

**Explanation:**

The correct answers are B. Option B and D. Option D. Both options B and D are compliant with PEP 8 recommendations for line breaks and indentation. PEP 8 recommends using 4 spaces per indentation level and breaking lines before binary operators. In option B, the arguments to the print function are aligned with the opening delimiter, which is another acceptable way to format long lines according to PEP 8. In option D, the

second line is indented by 4 spaces to distinguish it from the next logical line.

#### QUESTION 17

Look at the following examples of comments and docstrings in Python Select the ones that are useful and compliant with PEP 8 recommendations (Select the two best answers.)

A)

```
def area_price(area, price=1.25):  
    """Calculate the area in square meters.  
    Keyword arguments:  
    area -- the land area of the slot  
    price -- price per sq/m (default 1.25)"""  
    ...
```

B)

```
def area_price(area, price=2.25):  
    """Calculate the area in square meters.  
  
    Keyword arguments:  
    area -- the land area of the slot  
    price -- price per sq/m (default 2.25)  
    """  
    ...
```

C)

```
# Example that illustrates creating  
# a two-element list, and printing  
# the list contents to the screen.  
  
my_list = [a, b]  
print(my_list)
```

D)

```
price = price + 1 # Decrement price by one to compensate for loss.
```

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

**Correct Answer:** B, D

**Section:**

**Explanation:**

According to PEP 8 recommendations, the two best options are Option B and Option D.

Option B follows PEP 8's suggestion that all lines should be limited to 79 characters and for longer blocks of text like docstrings or comments, the length should be limited to 72 characters<sup>1</sup>. Option D follows PEP 8's conventions for writing good documentation strings (a.k.a. "docstrings") which are immortalized in PEP 257. It suggests writing docstrings for all public modules, functions, classes, and methods<sup>2</sup>.

**QUESTION 18**

Select the true statement related to PEP 257.

- A. String literals that occur immediately after another docstring are called attribute docstrings.
- B. Attribute docstrings and Additional docstrings are two types of extra docstrings that can be extracted by software tools.
- C. String literals that occur in places other than the first statement in a module, function, or class definition can act as documentation. They are recognized by the Python bytecode compiler and are accessible as runtime object attributes.
- D. String literals that occur immediately after a simple assignment at the top level of a module are called complementary docstrings.

**Correct Answer: B**

**Section:**

**Explanation:**

The true statement related to PEP 257 is Option B. According to PEP 257, string literals occurring elsewhere in Python code may also act as documentation. They are not recognized by the Python bytecode compiler and are not accessible as runtime object attributes (i.e. not assigned to doc), but two types of extra docstrings may be extracted by software tools: String literals occurring immediately after a simple assignment at the top level of a module, class, or init method are called "attribute docstrings". String literals occurring immediately after another docstring are called "additional docstrings"<sup>1</sup>.

**QUESTION 19**

Select the true statements related to PEP 8 programming recommendations for code writing. (Select two answers:)

- A. You should use the not ... is operator (e.g. if not spam is None:), rather than the is not operator (e.g. if spam is not None:), to increase readability.
- B. You should make object type comparisons using the isinstance() method (e.g. if isinstance(obj, int) :) instead of comparing types directly (eg if type(obj) is type(i)).
- C. You should write code in a way that favors the CPython implementation over PyPy, Cython, and Jython.
- D. You should not write string literals that rely on significant trailing whitespaces as they may be visually indistinguishable, and certain editors may trim them.

**Correct Answer: B, D**

**Section:**

**Explanation:**

The two true statements related to PEP 8 programming recommendations for code writing are Option B and Option D.

Option B is true because PEP 8 recommends making object type comparisons using the isinstance() method instead of comparing types directly<sup>1</sup>.

Option D is true because PEP 8 recommends not writing string literals that rely on significant trailing whitespaces as they may be visually indistinguishable, and certain editors may trim them<sup>1</sup>.

**QUESTION 20**

Select the true statements related to PEP 8 naming conventions. (Select two answers.)

- A. Function and variable names should be lower-case with words separated by underscores.
- B. You should always use self as the first argument to instance methods, and cls as the first argument to class methods.
- C. Modules should have short names written in CamelCase.
- D. Constants should be written in all lower-case letters with words separated by underscores.

**Correct Answer: A, D**

**Section:**

**Explanation:**

Option A is true because PEP 8 recommends that function and variable names should be lowercase, with words separated by underscores.

Option D is true because PEP 8 recommends that constants should be written in all capital letters with words separated by underscores.

PEP 8 is the official style guide for Python code. It provides guidelines for how to write readable code that follows consistent naming conventions. The aim of PEP 8 is to improve the readability of Python code and make it easier to understand and maintain.

According to PEP 8, variable and function names should be written in all lower-case letters with words separated by underscores, as stated in A. Constants, which are variables whose value is expected to remain constant throughout the code, should be written in all upper-case letters with words separated by underscores, as stated in D.

Reference:

PEP 8 -- Style Guide for Python Code: <https://www.python.org/dev/peps/pep-0008/Python> Documentation: <https://docs.python.org/3/tutorial/classes.html#classmethods-andstaticmethods>

**QUESTION 21**

Select the true statement about PEP 8 recommendations related to line breaks and binary operators.



- A. It is recommended that you use line breaks before binary operators to improve code readability.
- B. It is permissible to use line breaks before or after a binary operator as long as the convention is consistent locally However, for new code it is recommended that break lines should be used only after binary operators.
- C. It is recommended that you use line breaks after binary operators to improve code readability.
- D. There is no specific PEP 8 recommendation related to using line breaks with binary operators.

**Correct Answer: A**

**Section:**

**Explanation:**

According to PEP 8, Python's official style guide, line breaks before binary operators produce more readable code, especially in code blocks with long expressions. This is stated in several sources (1,2,6,8) and is a widely accepted convention.

Reference:

<https://www.python.org/dev/peps/pep-0008/#should-a-line-break-before-or-after-a-binaryoperator>

<https://stackoverflow.com/questions/30614124/are-long-lines-broken-up-before-or-after-binaryoperators-in-python>

<https://www.quora.com/What-is-PEP-8-Python>

<https://www.techbeamers.com/python-tutorial-pep-8/>

<https://www.section.io/engineering-education/python-coding-conventions-guidelines-for-pythonprogramming/>

<https://towardsdatascience.com/a-step-in-pep8-style-guide-improving-the-readability-of-the-code-8114fd4ccefa>

<https://www.codementor.io/@rishikeshdhokare/python-coding-style-best-practices-that-everypython-programmer-must-know-xybbcub8>

<https://www.dataschool.io/python-pep8-tips-and-tricks/>

## QUESTION 22

Look at the following code snippets and decide which ones follow PEP 8 recommendations for whitespaces in expressions and statements (Select two answers.)

A)

```
# Correct:
foo(5)
# Wrong:
foo (5)
```

No whitespace immediately before the opening parenthesis that starts the list of arguments of a function call:

B)

A whitespace immediately before a comma, semicolon, and colon:

```
# Correct:
if x == 2 : print x , y ; x , y = y , x
# Wrong:
if x == 2: print x, y; x, y = y, x
```

C)

No whitespace between a trailing comma and a following closing parenthesis:

```
# Correct:
spam = (1,)
# Wrong:
spam = (1, )
```

D)

A whitespace immediately after the opening parenthesis that starts indexing or slicing:

```
# Correct:
my_dict ['key'] = my_list [index]
# Wrong:
my_dict['key'] = my_list[index]
```

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

**Correct Answer: A, C**

**Section:**

**Explanation:**

Option A is true because PEP 8 recommends avoiding extraneous whitespace immediately inside parentheses, brackets or braces 1.

Option C is true because PEP 8 recommends avoiding extraneous whitespace between a trailing comma and a following close parenthesis 1.

#### QUESTION 23

Which of the following values can be returned by the messagebox. askquestion () method?

- A. "accept:" and "cancel"
- B. l and o
- C. "yes" and "no"
- D. True and False

**Correct Answer: C**

**Section:**

**Explanation:**

The messagebox.askquestion() method in Python's tkinter library displays a message box with a specified question and two response buttons labeled "Yes" and "No". It returns a string indicating which button was selected - either "yes" or "no".

This function is used to ask questions to the user that have only two options: YES or NO. It can be used to ask the user if they want to continue or if they want to submit something 1.

#### QUESTION 24

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.

**Correct Answer: A**

**Section:**

**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.

Reference:

<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 25

What is true about the `unbind()` method? (Select two answers.)

- A. It is invoked from within the events object
- B. It is invoked from within a widget's object
- C. It needs a widget's object as an argument
- D. It needs the event name as an argument

**Correct Answer: B, D**

**Section:**

**Explanation:**

Option B is true because the `unbind()` method is invoked from within a widget's object 1.

Option D is true because the `unbind()` method needs the event name as an argument 1.

The `unbind()` method in Tkinter is used to remove a binding between an event and a function. It can be invoked from within a widget's object when a binding is no longer needed. The method requires the event name as an argument to remove the binding for that specific event. For example: `button = tk.Button(root, text="Click me")` `button.bind("<Button-1>", callback_function)` # bind left mouse click event to `callback_function`  
`button.unbind("<Button-1>")` # remove the binding for the left mouse click event

#### QUESTION 26

If purple can be obtained from mixing red and blue, which color codes represent the two ingredients? Select two answers)

- A. #FFFFFF
- B. #0000FF
- C. #FF0000
- D. #000000

**Correct Answer: B, C**

**Section:**

**Explanation:**

If purple can be obtained from mixing red and blue, which color codes represent the two ingredients? Select two answers)

- 1. #FFFFFF
- 2. #0000FF
- 3. #FF0000
- 4. #000000

#### QUESTION 27

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.

**Correct Answer: A**

**Section:**

**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 28

Which of the following will set the button text's font to 12 point italics Arial? (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

**Correct Answer: B, C**

**Section:**

**Explanation:**

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

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

#### QUESTION 29

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

**Correct Answer: A, C**

**Section:**

**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.

Reference:

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)

#### QUESTION 30

If `w` is a correctly created main application window, which method would you use to find both of the main window's dimensions?

- A. `w. fixshape()`
- B. `w. fixdim()`
- C. `w. resizable()`



D. w.makewindow ()

**Correct Answer: C**

**Section:**

**Explanation:**

1. w.resizable()

The resizable() method takes two Boolean arguments, width and height, that specify whether the main window can be resized in the corresponding directions. Passing False to both arguments makes the main window non-resizable, whereas passing True to both arguments (or omitting them) makes the window resizable.

Here is an example that sets the dimensions of the main window to 500x400 pixels and makes it nonresizable: `import tkinter as tk root = tk.Tk() root.geometry("500x400") root.resizable(False, False) root.mainloop()`

Reference:

Tkinter documentation: <https://docs.python.org/3/library/tk.html>

Tkinter tutorial: [https://www.python-course.eu/python\\_tkinter.php](https://www.python-course.eu/python_tkinter.php)

The resizable () method of a tkinter window object allows you to specify whether the window can be resized by the user in the horizontal and vertical directions. You can pass two boolean arguments to this method, such as w.resizable (False, False), to prevent both dimensions from being changed. Alternatively, you can pass 0 or 1 as arguments, such as w.resizable (0, 0), to achieve the same effect<sup>1</sup>.

Reference:

1. w.resizable()

The resizable() method takes two Boolean arguments, width and height, that specify whether the main window can be resized in the corresponding directions. Passing False to both arguments makes the main window non-resizable, whereas passing True to both arguments (or omitting them) makes the window resizable.

Here is an example that sets the dimensions of the main window to 500x400 pixels and makes it nonresizable:

```
import tkinter as tk
```

```
root = tk.Tk()
```

```
root.geometry("500x400")
```

```
root.resizable(False, False)
```

```
root.mainloop()
```

Reference:

Tkinter documentation: <https://docs.python.org/3/library/tk.html>

Tkinter tutorial: [https://www.python-course.eu/python\\_tkinter.php](https://www.python-course.eu/python_tkinter.php)

The resizable () method of a tkinter window object allows you to specify whether the window can be resized by the user in the horizontal and vertical directions. You can pass two boolean arguments to this method, such as w.resizable (False, False), to prevent both dimensions from being

changed. Alternatively, you can pass 0 or 1 as arguments, such as w.resizable (0, 0), to achieve the same effect<sup>1</sup>.

Reference:

1: <https://stackoverflow.com/questions/36575890/how-to-set-a-tkinter-window-to-a-constant-size>

Other methods that can be used to control the window size are:

w.geometry () : This method allows you to set the initial size and position of the window by passing a string argument in the format "widthxheight+x+y", such as w.geometry ("500x500+100+100")<sup>12</sup>. w.minsize () and w.maxsize ():

These methods allow you to set the minimum and maximum size of the window in pixels, such as w.minsize (500, 500) and w.maxsize (1000, 1000)<sup>12</sup>. w.pack\_propagate () and w.grid\_propagate (): These methods allow you to enable or disable the propagation of the size of the widgets inside the window to the window itself. By default, these methods are set to True, which means that the window will adjust its size according to the widgets it contains. You can set

these methods to False or 0 to prevent this behavior, such as `w.pack_propagate(0)` or `w.grid_propagate(0)`.

`w.place()`: This method allows you to place the window at a specific position and size relative to its parent window or screen. You can use keyword arguments such as `x`, `y`, `width`, `height`, `relx`, `rely`, `relwidth`, and `relheight` to specify the coordinates and dimensions of the window in absolute or relative terms, such as `w.place(x=0, y=0, relwidth=1, relheight=1)`.

Reference:

2: <https://stackoverflow.com/questions/25690423/set-window-dimensions-in-tkinter-python-3> : <https://stackoverflow.com/questions/36575890/how-to-set-a-tkinter-window-to-a-constantsize/36576068#36576068> : <https://www.skotechlearn.com/2020/06/tkinter-window-position-sizecenter-screen-in-python.html>

### QUESTION 31

Select the true statements about the connection-oriented and connectionless types of communication. (Select two answers.)

- A. In the context of TCP/IP networks, the communication side that initiates a connection is called the client, whereas the side that answers the client is called the server
- B. Connectionless communications are usually built on top of TCP
- C. Using walkie-talkies is an example of a connection-oriented communication
- D. A phone call is an example of a connection-oriented communication

**Correct Answer: A, D**

**Section:**

**Explanation:**

A. In the context of TCP/IP networks, the communication side that initiates a connection is called the client, whereas the side that answers the client is called the server.

This statement is true because TCP/IP networks use a client-server model to establish connection-oriented communications. The client is the device or application that requests a service or resource from another device or application, which is called the server. The server responds to the client's request and provides the service or resource. For example, when you browse a website using a web browser, the browser acts as a client and sends a request to the web server that hosts the website. The web server acts as a server and sends back the requested web page to the browser<sup>1</sup>.

B. Connectionless communications are usually built on top of TCP.

This statement is false because TCP (Transmission Control Protocol) is a connection-oriented protocol that requires establishing and terminating a connection before and after sending data.

Connectionless communications are usually built on top of UDP (User Datagram Protocol), which is a connectionless protocol that does not require any connection setup or teardown. UDP simply sends data packets to the destination without checking if they are received or not<sup>2</sup>.

C. Using walkie-talkies is an example of a connection-oriented communication.

This statement is false because using walkie-talkies is an example of a connectionless communication. Walkie-talkies do not establish a dedicated channel or connection between the sender and receiver before transmitting data. They simply broadcast data over a shared frequency without ensuring that the receiver is ready or available to receive it. The sender does not know if the receiver has received the data or not<sup>3</sup>.

D. A phone call is an example of a connection-oriented communication.

This statement is true because a phone call is an example of a connection-oriented communication.

A phone call requires setting up a circuit or connection between the caller and callee before exchanging voice data. The caller and callee can hear each other's voice and know if they are connected or not. The phone call also requires terminating the connection when the conversation is over<sup>4</sup>.

Reference:

1: <https://www.techtarget.com/searchnetworking/definition/client-server> 2:

<https://www.javatpoint.com/connection-oriented-vs-connectionless-service> 3:

<https://en.wikipedia.org/wiki/Walkie-talkie> 4: [https://en.wikipedia.org/wiki/Telephone\\_call](https://en.wikipedia.org/wiki/Telephone_call)

A is true because in the context of TCP/IP networks, the communication side that initiates a connection is called the client, and the side that answers the client is called the server. This is the basis for establishing a connection-oriented communication.

D is true because a phone call is an example of a connection-oriented communication. Like TCP/IP, a phone call establishes a connection between two devices (in this case, two phones) before communication can occur.

A is true because in the context of TCP/IP networks, the communication side that initiates a connection is called the client, and the side that answers the client is called the server. This is the basis for establishing a connection-oriented communication.

D is true because a phone call is an example of a connection-oriented communication. Like TCP/IP, a phone call establishes a connection between two devices (in this case, two phones) before communication can occur.

B is false because connectionless communications are usually built on top of UDP, not TCP. UDP is a connectionless protocol that does not establish a connection before sending data.

C is false because using walkie-talkies is an example of a connectionless communication. Walkietalkies do not establish a connection before communication begins, and messages are simply broadcasted to all devices within range.

Here is a sample code in Python using the socket module to create a TCP server and client to demonstrate the connection-oriented communication:  
Server-side code:

```
import socket

HOST = '127.0.0.1'

PORT = 8080

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

Client-side code:

```
import socket

HOST = '127.0.0.1'

PORT = 8080

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world')
    data = s.recv(1024)
    print('Received', repr(data))
```

The server listens for incoming connections on port 8080, and when a connection is established, it prints the address of the client that has connected. The server then continuously receives data from the client and sends it back to the client until the connection is closed.

The client establishes a connection with the server and sends the message "Hello, world" encoded as bytes. It then waits for a response from the server and prints the data it receives.

**QUESTION 32**

Select the true statements about sockets. (Select two answers)

- A. A socket is a connection point that enables a two-way communication between programs running in a network.
- B. A socket is always the secure means by which computers on a network can safely communicate, without the risk of exposure to an attack
- C. A socket is a connection point that enables a one-way communication only between remote processes
- D. A socket can be used to establish a communication endpoint for processes running on the same or different machines.

**Correct Answer: A, D**

**Section:**

**Explanation:**

A. A socket is a connection point that enables a two-way communication between programs running in a network.

This statement is true because a socket is a software structure that serves as an endpoint for sending and receiving data across a network. A socket is defined by an application programming interface (API) for the networking architecture, such as TCP/IP. A socket can be used to establish a communication channel between two programs running on the same or different network nodes<sup>12</sup>.

B. A socket is always the secure means by which computers on a network can safely communicate, without the risk of exposure to an attack.

This statement is false because a socket by itself does not provide any security or encryption for the data transmitted over the network. A socket can be vulnerable to various types of attacks, such as eavesdropping, spoofing, hijacking, or denial-of-service. To ensure secure communication, a socket can use additional protocols or mechanisms, such as SSL/TLS, SSH, VPN, or firewall<sup>13</sup>.

C. A socket is a connection point that enables a one-way communication only between remote processes.

This statement is false because a socket can enable both one-way and two-way communication between processes running on the same or different network nodes. A socket can be used for connection-oriented or connectionless communication, depending on the type of protocol used. For example, TCP is a connection-oriented protocol that provides reliable and bidirectional data transfer, while UDP is a connectionless protocol that provides unreliable and unidirectional data transfer<sup>12</sup>.

D. A socket can be used to establish a communication endpoint for processes running on the same or different machines.

This statement is true because a socket can be used for inter-process communication (IPC) within a single machine or across different machines on a network. A socket can use different types of addresses to identify the processes involved in the communication, such as IP address and port number for network sockets, or file name or path for Unix domain sockets<sup>12</sup>.

Reference:

1: [https://en.wikipedia.org/wiki/Network\\_socket](https://en.wikipedia.org/wiki/Network_socket) 2: <https://www.geeksforgeeks.org/socket-incomputer-network/> 3: <https://www.tutorialspoint.com/what-is-a-network-socket-computernetworks>

**QUESTION 33**

Select the true statement about the socket. gaierror exception.

- A. It is raised when a timeout occurs on a socket.
- B. It is raised when a system function returns a system-related error.
- C. It is raised when an address-related error caused by the repr () function occurs.
- D. It is raised when an address-related error caused by the getaddrinfo () and getnameinfo () functions occurs.

**Correct Answer: D**

**Section:**

**Explanation:**

The socket.gaierror exception is raised when an address-related error caused by the getaddrinfo() and getnameinfo() functions occurs. These functions are used to translate hostnames to IP addresses and vice versa, and the gaierror exception is raised if they fail to perform this translation.

Reference:

Official Python documentation on socket.gaierror: <https://docs.python.org/3/library/socket.html#socket.gaierror>

**QUESTION 34**

Select the true statements about the json.-dumps () function. (Select two answers.)

- A. It returns a JSON string.
- B. It returns a Python entity.
- C. It takes a JSON string as its argument
- D. It takes Python data as its argument.



**Correct Answer: A, D**

**Section:**

**Explanation:**

The `json.dumps()` function is used to convert a Python object into a JSON string. It takes Python data as its argument, such as a dictionary or a list, and returns a JSON string.

Reference: Official Python documentation on `json.dumps()`: <https://docs.python.org/3/library/json.html#json.dumps>

A. It returns a JSON string.

This statement is true because the `json.dumps()` function takes a Python object as its argument and returns a JSON-formatted string that represents the object. For example, `json.dumps([1, 2, 3])` returns `'[1, 2, 3]'`.

D. It takes Python data as its argument.

This statement is true because the `json.dumps()` function accepts any Python object that can be serialized into JSON, such as lists, dictionaries, strings, numbers, booleans, or `None`. For example, `json.dumps({"name": "Alice", "age": 25})` returns `'{"name": "Alice", "age": 25}'`.

### QUESTION 35

What is ElementTree?

A. A Python built-in module that contains functions used for creating HTML files.

B. A Python library that contains an API used for parsing and manipulating JSON files.

C. A Python library that contains functions and tools used for manipulating text files in GUI Programming.

D. A Python built-in module that contains functions used for parsing and creating XML data.

**Correct Answer: D**

**Section:**

**Explanation:**

ElementTree is a Python built-in module that provides a simple and efficient API for parsing and creating XML data. It allows you to access and manipulate XML data in a very straightforward way, making it easy to write XML processing applications.

Reference: Official Python documentation on ElementTree: <https://docs.python.org/3/library/xml.etree.elementtree.html> This statement is true because ElementTree is a module in the standard library of Python that provides an API for working with XML data. The module supports parsing XML from strings or files, creating XML trees from scratch or modifying existing ones, searching and iterating over XML elements, and writing XML data to strings or files.

### QUESTION 36

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

**Correct Answer: A**

**Section:**

**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#jsonserialization>

### QUESTION 37

Select the true statements about the following invocation:

```
r = requests.get('http://localhost:3000')
```

(Select two answers.)

A. It addresses a service deployed at localhost (the host where the code is run).

B. It addresses a service whose timeout is set to 3000 ms.

- C. It addresses a service located at the following address local.host.com.
- D. It addresses a service listening at port 3000.

**Correct Answer: A, D**

**Section:**

**Explanation:**

A. It addresses a service deployed at localhost (the host where the code is run).

This statement is true because localhost is a special hostname that refers to the local machine or the current host where the code is run. It is equivalent to using the IP address 127.0.0.1, which is the loopback address of the network interface. By using localhost as the hostname, the invocation addresses a service that is deployed on the same machine as the client.

D. It addresses a service listening at port 3000.

This statement is true because port 3000 is the part of the URL that follows the colon after the hostname. It specifies the port number where the service is listening for incoming requests. A port number is a 16-bit integer that identifies a specific process or application on a host. By using port 3000, the invocation addresses a service that is listening at that port.

B. It addresses a service whose timeout is set to 3000 ms.

This statement is false because timeout is not a part of the URL, but a parameter that can be passed to the requests.get () function in Python. Timeout specifies how long to wait for the server to send data before giving up. It is measured in seconds, not milliseconds. By using timeout=3, the invocation sets the timeout to 3 seconds, not 3000 ms.

C. It addresses a service located at the following address local.host.com.

This statement is false because local.host.com is not the same as localhost. Local.host.com is a fully qualified domain name (FQDN) that consists of three parts: local, host, and com. It requires DNS resolution to map it to an IP address. Localhost, on the other hand, is a special hostname that does not require DNS resolution and always maps to 127.0.0.1. By using localhost as the hostname, the invocation does not address a service located at local.host.com.

Reference:

: <https://docs.python.org/3/library/requests.html> : <https://en.wikipedia.org/wiki/Localhost> :

[https://en.wikipedia.org/wiki/Port\\_\(computer\\_networking\)](https://en.wikipedia.org/wiki/Port_(computer_networking)) :

[https://en.wikipedia.org/wiki/Fully\\_qualified\\_domain\\_name](https://en.wikipedia.org/wiki/Fully_qualified_domain_name)

#### QUESTION 38

A socket object is usually created by which one of the following invocations?

- A. socket = socket (socket\_domain, socket\_type)
- B. socket = socket. socket (socket\_number)
- C. socket = socket. socket (socket\_domain, socket\_type, server\_address)
- D. socket = socket.socket(server address)

**Correct Answer: A**

**Section:**

**Explanation:**

A socket object is usually created using the socket() constructor provided by the socket module in Python. The correct invocation is socket.socket(socket\_domain, socket\_type). This creates a new socket object with the specified socket domain and type.

Reference: Official Python documentation on socket programming: <https://docs.python.org/3/library/socket.html>

#### QUESTION 39

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

**Correct Answer: C, D**

**Section:**

**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 40

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. `svQUOTE_ALL`
- D. `csv.QUOTE_NONNUMERIC`

**Correct Answer: A**

**Section:**

**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>

#### QUESTION 41

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`

**Correct Answer: B**

**Section:**

**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 42

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

- A. The sqlite3 module provides an interface compliant with the DB-API 2.0.
- B. The special name 'memory' is used to create a database in RAM.
- C. The sqlite3 module does not support transactions.
- D. The fetchall method returns an empty list when no rows are available

**Correct Answer: A, B**

**Section:**

**Explanation:**

A. The sqlite3 module in python provides an interface compliant to the DB-API 2.0. Thus, it follows a standard performance metric that allows for consistency in database programming with python.

B. The special name 'memory' is used to create a database in RAM using the sqlite3 module. Thus, when you use it as the name of the database file while opening a connection, it creates a temporary database that exists only in memory.

Reference: Official Python documentation on sqlite3: <https://docs.python.org/3/library/sqlite3.html>

#### QUESTION 43

www.VCEplus.io

What is the result of the following code?

```
import logging

logger = logging.getLogger()

logger.info('Debugging mode has been enabled')
logger.debug('Loading data...')
```

What is the result of the following code?

- A. Nothing will be displayed
- B. Loading data...
- C. Debugging mode has been enabled
- D. Debugging mode has been enabled Loading data...

**Correct Answer: B**

**Section:**

**Explanation:**

This statement is true because the code uses the logging module to create a logger object and set its level to logging.INFO. The logging module provides a way of reporting events that occur during the execution of a program. The logging level determines which events are reported and which are ignored. The logging module defines five levels of severity: DEBUG, INFO, WARNING, ERROR, and CRITICAL. The lower the level, the more events are reported.

The code then uses the logger object to log two messages: one with the level logging.DEBUG and one with the level logging.INFO. The logger object only reports the messages that have a level equal or higher than its own level. Therefore, the message with the level logging.DEBUG is ignored, while the message with the level logging.INFO is reported. The default format for reporting messages is "level name: message". Therefore, the output of the code is:

INFO: Loading data...

www.VCEplus.io

#### QUESTION 44

What will be the content of the colors.csv file when you run the following code?

```
import csv

with open('colors.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile, delimiter=',', quotechar='"', quoting=csv.QUOTE_NONE)

    writer.writerow(['Name'])
    writer.writerow(['red', 'green', 'blue'])
    writer.writerow(['yellow'])
```

A)

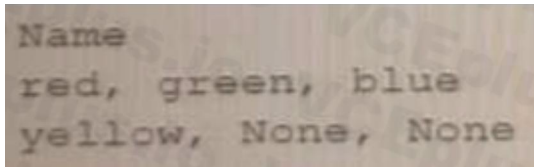
```
Name
red, green, blue
yellow
```

B)

```
Name
"red, green, blue"
yellow
```

C)





```
Name  
red, green, blue  
yellow, None, None
```

D)

An exception will be raised.

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

**Correct Answer: B**

**Section:**

**Explanation:**

#### QUESTION 45

Which of the following methods allow you to load a configuration using ConfigParser? (Select two answers.)

- A. read
- B. read\_dict
- C. read\_conf
- D. read\_str

**Correct Answer: A, D**

**Section:**

**Explanation:**

ConfigParser is a built-in library in Python that allows you to read and write configuration files. The read method is used to read the configuration file which can be in any of the supported file formats, such as INI, YAML, and JSON. The read\_dict method is used to read the configuration from a Python dictionary. The read\_conf and read\_str options are not valid methods in the ConfigParser module.

Therefore, the correct options to load a configuration using ConfigParser are A. read and D. read\_string.

Reference: Official Python documentation on

ConfigParser: <https://docs.python.org/3/library/configparser.html>