

Python-Institute

Exam Questions PCEP-30-02

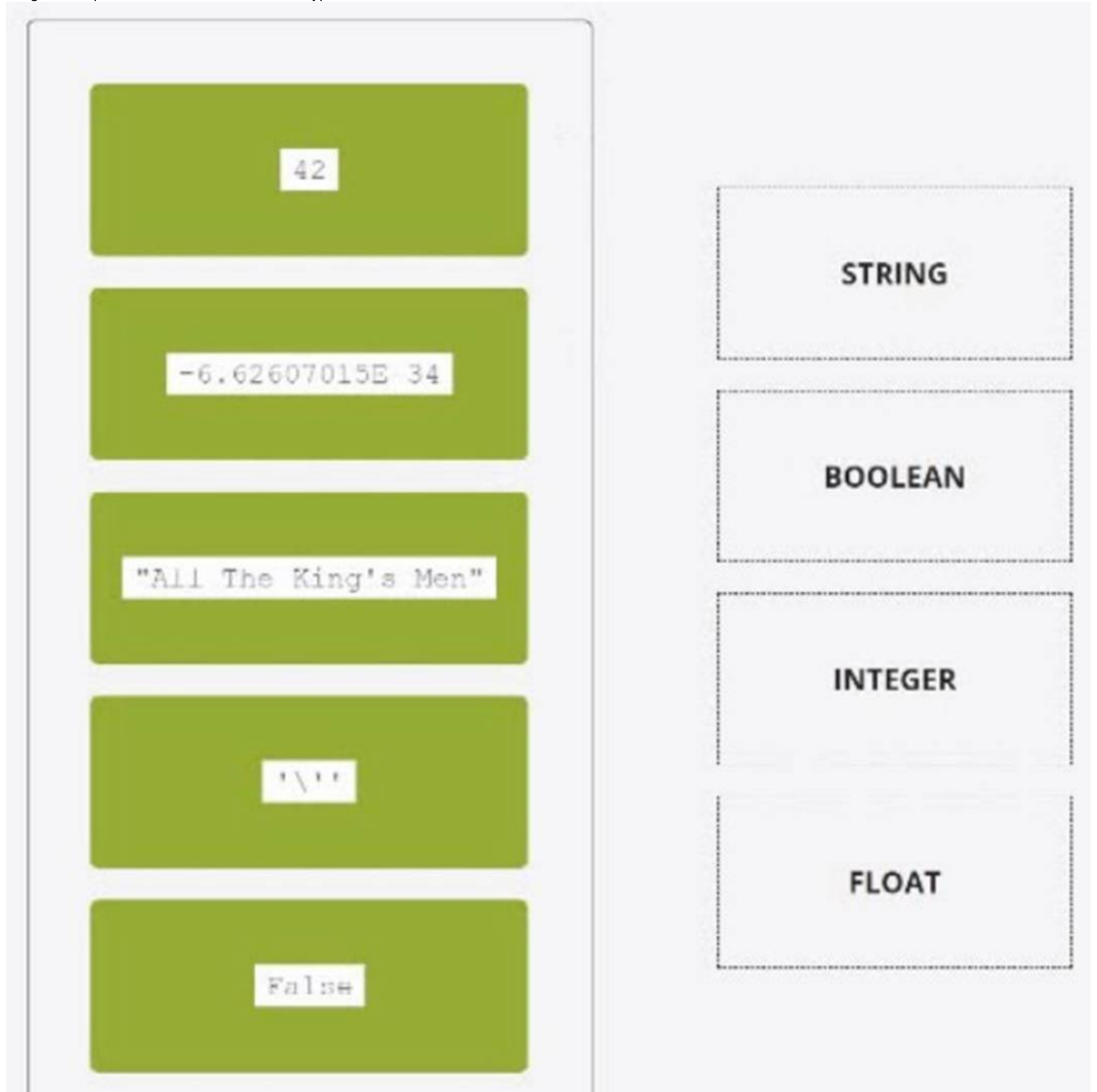
PCEP - Certified Entry-Level Python Programmer



NEW QUESTION 1

DRAG DROP

Drag and drop the literals to match their data type names.



- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

One possible way to drag and drop the literals to match their data type names is:

? STRING: ??All The King??s Men??

? BOOLEAN: False

? INTEGER: 42

? FLOAT: -6.62607015E-34

A literal is a value that is written exactly as it is meant to be interpreted by the Python interpreter. A data type is a category of values that share some common characteristics or operations. Python has four basic data types: string, boolean, integer, and float.

A string is a sequence of characters enclosed by either single or double quotes. A string can represent text, symbols, or any other information that can be displayed as text. For example, ??All The King??s Men?? is a string literal that represents the title of a novel.

A boolean is a logical value that can be either True or False. A boolean can represent the result of a comparison, a condition, or a logical operation. For example,

False is a boolean literal that represents the opposite of True.

An integer is a whole number that can be positive, negative, or zero. An integer can represent a count, an index, or any other quantity that does not require fractions or decimals. For example, 42 is an integer literal that represents the answer to life, the universe, and everything.

A float is a number that can have a fractional part after the decimal point. A float can represent a measurement, a ratio, or any other quantity that requires precision or

approximation. For example, -6.62607015E-34 is a float literal that represents the Planck constant in scientific notation.

You can find more information about the literals and data types in Python in the following references:

? [Python Data Types]

? [Python Literals]

? [Python Basic Syntax]

NEW QUESTION 2

What is the expected output of the following code?

```
def traverse(stop):
    if stop == 0:
        return 0
    else:
        return stop * traverse(stop - 1)

print(traverse(2))
```

- A. 2
- B. 3
- C. 1

Answer: D

Explanation:

The code snippet that you have sent is using the count method to count the number of occurrences of a value in a list. The code is as follows:

```
my_list = [1, 2, 3, 4, 5] print(my_list.count(1))
```

The code starts with creating a list called `my_list` that contains the numbers 1, 2, 3, 4, and 5. Then, it uses the print function to display the result of calling the count method on the list with the argument 1. The count method is used to return the number of times a value appears in a list. For example, `my_list.count(1)` returns 1, because 1 appears once in the list.

The expected output of the code is 1, because the code prints the number of occurrences of 1 in the list. Therefore, the correct answer is D. 1.

Reference: Python List count() Method - W3Schools

NEW QUESTION 3

What is true about exceptions and debugging? (Select two answers.)

- A. A tool that allows you to precisely trace program execution is called a debugger.
- B. If some Python code is executed without errors, this proves that there are no errors in it.
- C. One try-except block may contain more than one except branch.
- D. The default (anonymous) except branch cannot be the last branch in the try-except block.

Answer: AC

Explanation:

Exceptions and debugging are two important concepts in Python programming that are related to handling and preventing errors. Exceptions are errors that occur when the code cannot be executed properly, such as syntax errors, type errors, index errors, etc. Debugging is the process of finding and fixing errors in the code, using various tools and techniques. Some of the facts about exceptions and debugging are:

? A tool that allows you to precisely trace program execution is called a debugger. A debugger is a program that can run another program step by step, inspect the values of variables, set breakpoints, evaluate expressions, etc. A debugger can help you find the source and cause of an error, and test possible solutions. Python has a built-in debugger module called `pdb`, which can be used from the command line or within the code. There are also other third-party debuggers available for Python, such as PyCharm, Visual Studio Code, etc¹²

? If some Python code is executed without errors, this does not prove that there are no errors in it. It only means that the code did not encounter any exceptions that would stop the execution. However, the code may still have logical errors, which are errors that cause the code to produce incorrect or unexpected results. For example, if you write a function that is supposed to calculate the area of a circle, but you use the wrong formula, the code may run without errors, but it will give you the wrong answer. Logical errors are harder to detect and debug than syntax or runtime errors, because they do not generate any error messages. You have to test the code with different inputs and outputs, and compare them with the expected results³⁴

? One try-except block may contain more than one except branch. A try-except block is a way of handling exceptions in Python, by using the keywords `try` and `except`. The try block contains the code that may raise an exception, and the except block contains the code that will execute if an exception occurs. You can have multiple except blocks for different types of exceptions, or for different actions to take. For example, you can write a try-except block like this:

```
try: # some code that may raise an exception
except ValueError: # handle the ValueError exception
except ZeroDivisionError: # handle the ZeroDivisionError
exception: # handle any other exception
```

This way, you can customize the error handling for different situations, and provide more informative messages or alternative solutions⁵

? The default (anonymous) except branch can be the last branch in the try-except block. The default except branch is the one that does not specify any exception type, and it will catch any exception that is not handled by the previous except branches. The default except branch can be the last branch in the try-except block, but it cannot be the first or the only branch. For example, you can write a try- except block like this:

```
try: # some code that may raise an exception
except ValueError: # handle the ValueError exception
except: # handle any other exception
```

This is a valid try-except block, and the default except branch will be the last branch. However, you cannot write a try-except block like this:

```
try: # some code that may raise an exception
except: # handle any exception
```

This is an invalid try-except block, because the default except branch is the only branch, and it will catch all exceptions, even those that are not errors, such as

KeyboardInterrupt or SystemExit. This is considered a bad practice, because it may hide or ignore important exceptions that should be handled differently or propagated further. Therefore, you should always specify the exception types that you want to handle, and use the default except branch only as a last resort. Therefore, the correct answers are A. A tool that allows you to precisely trace program execution is called a debugger. and C. One try-except block may contain more than one except branch.

Reference: Python Debugger – Python pdb - GeeksforGeeks
 How can I see the details of an exception in Python's debugger? Python Debugging (fixing problems)
 Python - start interactive debugger when exception would be otherwise thrown
 Python Try Except [Error Handling and Debugging — Programming with Python for Engineers]

NEW QUESTION 4

DRAG DROP

Assuming that the phone_dir dictionary contains name: number pairs, arrange the code boxes to create a valid line of code which retrieves Martin Eden's phone number, and assigns it to the number variable.

]

number

"Martin Eden"

[

phone_dir

=

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

`number = phone_dir["Martin Eden"]`

This code uses the square brackets notation to access the value associated with the key "Martin Eden" in the phone_dir dictionary. The value is then assigned to the variable number. A dictionary is a data structure that stores key-value pairs, where each key is unique and can be used to retrieve its corresponding value.

You can find more information about dictionaries in Python in the following references:

- ? [Python Dictionaries - W3Schools]
- ? [Python Dictionary (With Examples) - Programiz]
- ? [5.5. Dictionaries — How to Think Like a Computer Scientist ??]

NEW QUESTION 5

Which of the following expressions evaluate to a non-zero result? (Select two answers.)

- A. $2 ** 3 / A - 2$
- B. $4 / 2 ** 3 - 2$
- C. $1 ** 3 / 4 - 1$
- D. $1 * 4 // 2 ** 3$

Answer: AB

Explanation:

In Python, the ** operator is used for exponentiation, the / operator is used for floating-point division, and the // operator is used for integer division. The order of operations is parentheses, exponentiation, multiplication/division, and addition/subtraction. Therefore, the expressions can be evaluated as follows:

* A. $2 ** 3 / A - 2 = 8 / A - 2$ (assuming A is a variable that is not zero or undefined) B. $4 / 2 ** 3 - 2 = 4 / 8 - 2 = 0.5 - 2 = -1.5$ C. $1 ** 3 / 4 - 1 = 1 / 4 - 1 = 0.25 - 1 = -0.75$ D. $1 * 4 // 2 ** 3 = 4 // 8 = 0$

Only expressions A and B evaluate to non-zero results.

Reference: [Python Institute - Entry-Level Python Programmer Certification]

NEW QUESTION 6

What is the expected output of the following code?

```
counter = 84 // 2
if counter < 0:
    print("*")
elif counter >= 42:
    print("***")
else:
    print("**")
```

- A. The code produces no output.
- B. * * *
- C. * *
- D. *

Answer: C

Explanation:

The code snippet that you have sent is a conditional statement that checks if a variable `counter` is less than 0, greater than or equal to 42, or neither. The code is as follows: `if counter < 0: print(???) elif counter >= 42: print(???) else: print(???)`
 The code starts with checking if the value of `counter` is less than 0. If yes, it prints a single asterisk (*) to the screen and exits the statement. If no, it checks if the value of `counter` is greater than or equal to 42. If yes, it prints three asterisks (***) to the screen and exits the statement. If no, it prints two asterisks (**) to the screen and exits the statement.
 The expected output of the code depends on the value of `counter`. If the value of `counter` is 10, as shown in the image, the code will print two asterisks (**) to the screen, because 10 is neither less than 0 nor greater than or equal to 42. Therefore, the correct answer is C.

Reference: [Python Institute - Entry-Level Python Programmer Certification]

NEW QUESTION 7

Assuming that the following assignment has been successfully executed: `My_list = [1, 1, 2, 3]`
 Select the expressions which will not raise any exception. (Select two expressions.)

- A. `my_list[-10]`
- B. `my_list|my_Li1st | 3| |`
- C. `my list [6]`
- D. `my_List- [0:1]`

Answer: BD

Explanation:

The code snippet that you have sent is assigning a list of four numbers to a variable called `my_list`. The code is as follows:
`my_list = [1, 1, 2, 3]`
 The code creates a list object that contains the elements 1, 1, 2, and 3, and assigns it to the variable `my_list`. The list can be accessed by using the variable name or by using the index of the elements. The index starts from 0 for the first element and goes up to the length of the list minus one for the last element. The index can also be negative, in which case it counts from the end of the list. For example, `my_list[0]` returns 1, and `my_list[-1]` returns 3.
 The code also allows some operations on the list, such as slicing, concatenation, repetition, and membership. Slicing is used to get a sublist of the original list by specifying the start and end index. For example, `my_list[1:3]` returns [1, 2]. Concatenation is used to join two lists together by using the + operator. For example, `my_list + [4, 5]` returns [1, 1, 2, 3, 4, 5]. Repetition is used to create a new list by repeating the original list a number of times by using the * operator. For example, `my_list * 2` returns [1, 1, 2, 3, 1, 1, 2, 3]. Membership is used to check if an element is present in the list by using the in operator. For example, `2 in my_list` returns True, and `4 in my_list` returns False.
 The expressions that you have given are trying to access or manipulate the list in different ways. Some of them are valid, and some of them are invalid and will raise an exception. An exception is an error that occurs when the code cannot be executed properly. The expressions are as follows:
 * A. `my_list[-10]`: This expression is trying to access the element at the index -10 of the list. However, the list only has four elements, so the index -10 is out of range. This will raise an `IndexError` exception and output nothing.
 * B. `my_list|my_Li1st | 3| |`: This expression is trying to perform a bitwise OR operation on the list and some other operands. The bitwise OR operation is used to compare the binary representation of two numbers and return a new number that has a 1 in each bit position where either number has a 1. For example, `3 | 1`

returns 3, because 3 in binary is 11 and 1 in binary is 01, and 11 | 01 is 11. However, the bitwise OR operation cannot be applied to a list, because a list is not a number. This will raise a `TypeError` exception and output nothing.

* C. `my_list[6]`: This expression is trying to access the element at the index 6 of the list. However, the list only has four elements, so the index 6 is out of range. This will raise an `IndexError` exception and output nothing.

* D. `my_List- [0:1]`: This expression is trying to perform a subtraction operation on the list and a sublist. The subtraction operation is used to subtract one number from another and return the difference. For example, `3 - 1` returns 2. However, the subtraction operation cannot be applied to a list, because a list is not a number. This will raise a `TypeError` exception and output nothing.

Only two expressions will not raise any exception. They are:

* B. `my_list|my_Li1st | 3| l`: This expression is not a valid Python code, but it is not an expression that tries to access or manipulate the list. It is just a string of characters that has no meaning. Therefore, it will not raise any exception, but it will also not output anything.

* D. `my_List- [0:1]`: This expression is a valid Python code that uses the slicing operation to get a sublist of the list. The slicing operation does not raise any exception, even if the start or end index is out of range. It will just return an empty list or the closest possible sublist. For example, `my_list[0:10]` returns `[1, 1, 2, 3]`, and `my_list[10:20]` returns `[]`. The expression `my_List- [0:1]` returns the sublist of the list from the index 0 to the index 1, excluding the end index. Therefore, it returns `[1]`. This expression will not raise any exception, and it will output `[1]`.

Therefore, the correct answers are B. `my_list|my_Li1st | 3| l` and D. `my_List- [0:1]`. Reference: [Python Institute - Entry-Level Python Programmer Certification]

NEW QUESTION 8

What is true about tuples? (Select two answers.)

- A. Tuples are immutable, which means that their contents cannot be changed during their lifetime.
- B. The `len { }` function cannot be applied to tuples.
- C. An empty tuple is written as `{ }`.
- D. Tuples can be indexed and sliced like lists.

Answer: AD

Explanation:

Tuples are one of the built-in data types in Python that are used to store collections of data. Tuples have some characteristics that distinguish them from other data types, such as lists, sets, and dictionaries. Some of these characteristics are:

? Tuples are immutable, which means that their contents cannot be changed during their lifetime. Once a tuple is created, it cannot be modified, added, or removed. This makes tuples more stable and reliable than mutable data types. However, this also means that tuples are less flexible and dynamic than mutable data types. For example, if you want to change an element in a tuple, you have to create a new tuple with the modified element and assign it to the same variable¹²

? Tuples are ordered, which means that the items in a tuple have a defined order and can be accessed by using their index. The index of a tuple starts from 0 for the first item and goes up to the length of the tuple minus one for the last item. The index can also be negative, in which case it counts from the end of the tuple. For example, if you have a tuple `t = ("a", "b", "c")`, then `t[0]` returns "a", and `t[- 1]` returns "c"¹²

? Tuples can be indexed and sliced like lists, which means that you can get a single item or a sublist of a tuple by using square brackets and specifying the start and end index. For example, if you have a tuple `t = ("a", "b", "c", "d", "e")`, then `t[2]` returns "c", and `t[1:4]` returns ("b", "c", "d"). Slicing does not raise any exception, even if the start or end index is out of range. It will just return an empty tuple or the closest possible sublist¹²

? Tuples can contain any data type, such as strings, numbers, booleans, lists, sets, dictionaries, or even other tuples. Tuples can also have duplicate values, which means that the same item can appear more than once in a tuple. For example, you can have a tuple `t = (1, 2, 3, 1, 2)`, which contains two 1s and two 2s¹²

? Tuples are written with round brackets, which means that you have to enclose the items in a tuple with parentheses. For example, you can create a tuple `t = ("a", "b", "c")` by using round brackets. However, you can also create a tuple without using round brackets, by just separating the items with commas. For example, you can create the same tuple `t = "a", "b", "c"` by using commas. This is called tuple packing, and it allows you to assign multiple values to a single variable¹²

? The `len()` function can be applied to tuples, which means that you can get the number of items in a tuple by using the `len()` function. For example, if you have a tuple `t = ("a", "b", "c")`, then `len(t)` returns 3¹²

? An empty tuple is written as `()`, which means that you have to use an empty pair of parentheses to create a tuple with no items. For example, you can create an empty tuple `t = ()` by using empty parentheses. However, if you want to create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple. For example, you can create a tuple with one item `t = ("a",)` by using a comma¹²

Therefore, the correct answers are A. Tuples are immutable, which means that their contents cannot be changed during their lifetime. and D. Tuples can be indexed and sliced like lists.

Reference: Python Tuples - W3SchoolsTuples in Python - GeeksforGeeks

NEW QUESTION 10

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

PCEP-30-02 Practice Exam Features:

- * PCEP-30-02 Questions and Answers Updated Frequently
- * PCEP-30-02 Practice Questions Verified by Expert Senior Certified Staff
- * PCEP-30-02 Most Realistic Questions that Guarantee you a Pass on Your First Try
- * PCEP-30-02 Practice Test Questions in Multiple Choice Formats and Updates for 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The PCEP-30-02 Practice Test Here](#)