Providing You Marvelous PCEP-30-02 Study Material with 100% Passing Guarantee

BTW, DOWNLOAD part of ExamcollectionPass PCEP-30-02 dumps from Cloud Storage: https://drive.google.com/open?id=1yma4dcXyUW_sJUpAkc817pigNJxn9kCY

One of the best ways to prepare for the Python Institute PCEP-30-02 exam is to study the PCEP - Certified Entry-Level Python Programmer (PCEP-30-02) exam questions. Familiarizing yourself with the PCEP-30-02 certification using practice test on real-world data sets can help you build your confidence and prepare you for the exam. Additionally, taking PCEP-30-02 Exam Questions and quizzes can help you identify areas where you need to improve and gauge your understanding of the material.

Python Institute PCEP-30-02 Exam Syllabus Topics:

Topic	Details
Topic 1	• Control Flow: This section covers conditional statements such as if, if-else, if-elif, if-elif-else
Topic 2	• Data Collections: In this section, the focus is on list construction, indexing, slicing, methods, and comprehensions; it covers Tuples, Dictionaries, and Strings.
Topic 3	• Functions and Exceptions: This part of the exam covers the definition of function and invocation
Topic 4	• Loops: while, for, range(), loops control, and nesting of loops.
Topic 5	• parameters, arguments, and scopes. It also covers Recursion, Exception hierarchy, Exception handling, etc.

>> PCEP-30-02 Study Material <<

Reliable Python Institute PCEP-30-02 Exam Prep | PCEP-30-02 Exam PDF

With limited time for your preparation, many exam candidates can speed up your pace of making progress. Our PCEP-30-02 study materials will remedy your faults of knowledge understanding. As we know, some people failed the exam before, and lost confidence in this agonizing exam before purchasing our PCEP-30-02 training guide. Also it is good for releasing pressure. Many customers get manifest improvement and lighten their load with our PCEP-30-02 exam braindumps. So just come and have a try!

Python Institute PCEP - Certified Entry-Level Python Programmer Sample Questions (Q18-Q23):

NEW QUESTION # 18
What is the expected output of the following code?

```
menu = {"pizza": 2.39, "pasta": 1.99, "folpetti": 3.99}

for value in menu:
    print(str(value)[0], end="")
```

- A. pizzapastafolpetti
- B. 0
- C. ppt
- D. The code is erroneous and cannot be run.

Answer: C

Explanation:
Explanation
The code snippet that you have sent is using the slicing operation to get parts of a string and concatenate them together. The code is as follows:
pizza = "pizza" pasta = "pasta" folpetti = "folpetti" print(pizza[0] + pasta[0] + folpetti[0]) The code starts with assigning the strings "pizza", "pasta", and "folpetti" to the variables pizza, pasta, and folpetti respectively. Then, it uses the print function to display the result of concatenating the first characters of each string. The first character of a string can be accessed by using the index 0 inside square brackets. For example, pizza[0] returns "p". The concatenation operation is used to join two or more strings together by using the + operator. For example, "a" + "b" returns "ab". The code prints the result of pizza[0] + pasta[0] + folpetti[0], which is "p" + "p" + "f", which is "ppt".
The expected output of the code is ppt, because the code prints the first characters of each string. Therefore, the correct answer is B. ppt.

NEW QUESTION # 19
Assuming that the phone_dir dictionary contains name:number pairs, arrange the code boxes to create a valid line of code which adds Oliver Twist's phone number (5551122333) to the directory.

Answer:

Explanation:
phone_dir["Oliver Twist"] = ["5551122333"]
Explanation:

To correctly add Oliver Twist's phone number to the phone_dir dictionary, the code must follow this phone_dir["Oliver Twist"] = ["5551122333"] Now, let's match that with your code boxes and arrange them:
* phone_dir
* [
* "Oliver Twist"
*]
* =
* [
* "5551122333"
*]
Final Order:phone_dir # [# "Oliver Twist" #] # = # [# "5551122333" #]

NEW QUESTION # 20
What is the expected result of running the following code?

```
def do_the_mess(parameter):
    parameter[0] += variable
    return parameter[0]


the_list = [x for x in range(2, 3)]
variable = -1
do_the_mess(the_list)
print(the list[0])
```


- A. The code raises an unhandled exception.
- B. The code prints 0
- C. The code prints 2
- D. The code prints 1 .

Answer: A

Explanation:
The code snippet that you have sent is trying to use the index method to find the position of a value in a list.
The code is as follows:
the_list = [1, 2, 3, 4, 5] print(the_list.index(6))
The code starts with creating a list called "the_list" that contains the numbers 1, 2, 3, 4, and 5. Then, it tries to print the result of calling the index method on the list with the argument 6. The index method is used to return the first occurrence of a value in a list. For example, the_list.index(1) returns 0, because 1 is the first value in the list.
However, the code has a problem. The problem is that the value 6 is not present in the list, so the index method cannot find it. This will cause a ValueError exception, which is an error that occurs when a function or operation receives an argument that has the right type but an inappropriate value. The code does not handle the exception, and therefore it will terminate with an error message.
The expected result of the code is an unhandled exception, because the code tries to find a value that does not exist in the list.
Therefore, the correct answer is C. The code raises an unhandled exception.
Reference: Python List index() Method - W3SchoolsPython Exceptions: An Introduction - Real Python

NEW QUESTION # 21
What is true about tuples? (Select two answers.)

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

Answer: C,D

Explanation:
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.

NEW QUESTION # 22
What is true about exceptions and debugging? (Select two answers.)

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

Answer: B,D

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 except: # 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 resort5 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 - GeeksforGeeksHow 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 thrownPython Try Except [Error Handling and Debugging - Programming with Python for Engineers]

NEW QUESTION # 23
.....

If you want to learn the PCEP-30-02 practice guide anytime, anywhere, then we can tell you that you can use our products on a variety of devices. As you can see on our website, we have three different versions of the PCEP-30-02 exam questions: the PDF, Software and APP online. Though the content of them are the same. But the displays are totally different. And you can use them to study on different time and conditions. If you want to know them clearly, you can just free download the demos of the PCEP-30-02 Training Materials!