Free Questions for PCEP-30-02 by actualtestdumps

Shared by Moran on 24-05-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

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])
```

Options:

A- The code prints 1 .

B- The code prints 2

C- The code raises an unhandled exception.

D- The code prints 0

Answer:

C

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.

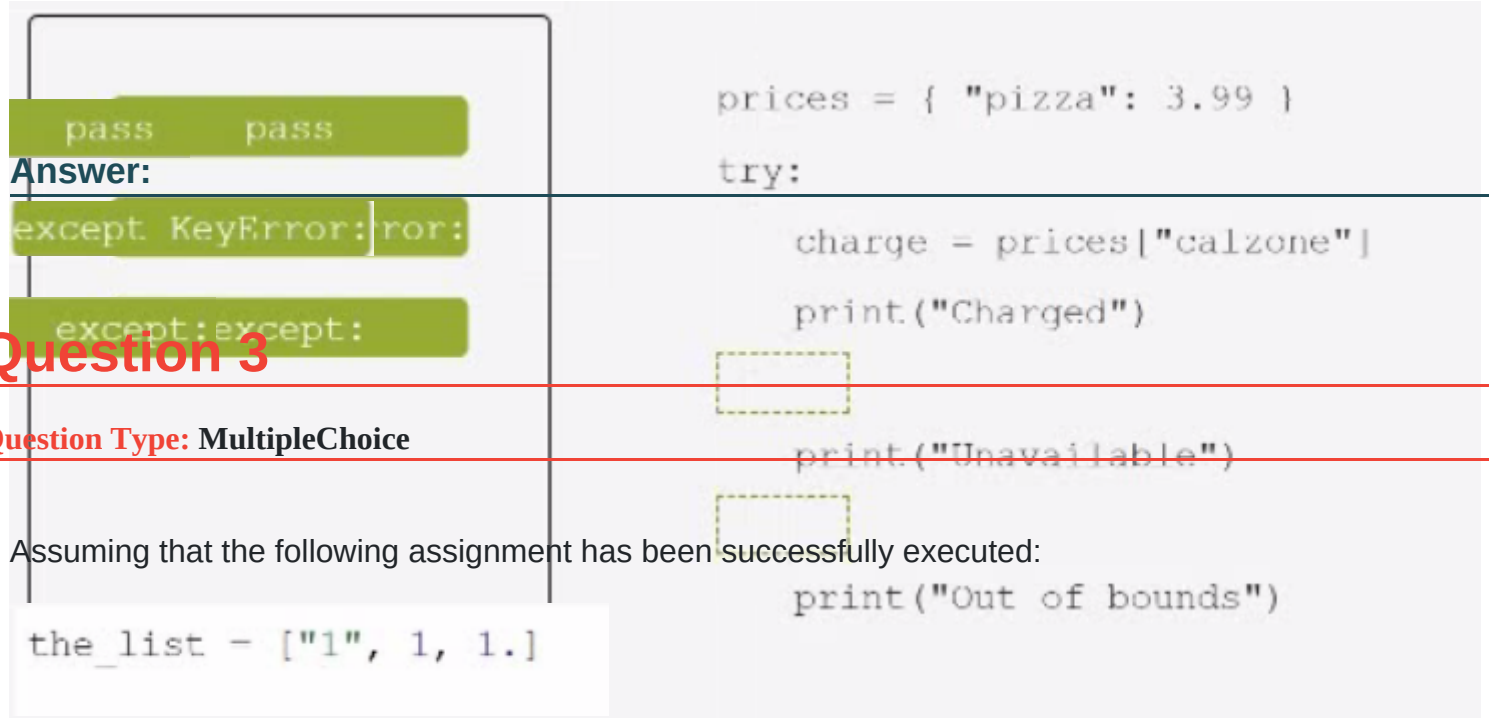
The expected result of the code is an unhandled exception, because the code tries to find a value that does not exist in the list. Therefore, the correct answer is C. The code raises an unhandled exception.

Question 2

Question Type: DragDrop

Drag and drop the code boxes in order to build a program which prints Unavailable to the screen.

(Note: one code box will not be used.)



Answer:

Question 3

Question Type: MultipleChoice

Assuming that the following assignment has been successfully executed:

Which of the following expressions evaluate to True? (Select two expressions.)

Options:

A- the_List.index {'1'} in the_list

B- 1.1 in the_list |1:3 |

C- len (the list [0:2]} <3

D- the_list. index {'1'} -- 0

Answer:

C, D

Explanation:

The code snippet that you have sent is assigning a list of four values to a variable called "the_list". The code is as follows:

the_list = ['1', 1, 1, 1]

The code creates a list object that contains the values '1', 1, 1, and 1, and assigns it to the variable "the_list". The list can be accessed by using the variable name or by using the index of the values. The index starts from 0 for the first value and goes up to the length of the list minus one for the last value. The index can also be negative, in which case it counts from the end of the list. For example, the_list[0] returns '1', and the_list[-1] returns 1.

The expressions that you have given are trying to evaluate some conditions on the list and return a boolean value, either True or False. 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) the_List.index {"1"} in the_list: This expression is trying to check if the index of the value '1' in the list is also a value in the list. However, this expression is invalid, because it uses curly brackets instead of parentheses to call the index method. 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_list.index {"1"} will raise a SyntaxError exception and output nothing.

B) 1.1 in the_list |1:3 |: This expression is trying to check if the value 1.1 is present in a sublist of the list. However, this expression is invalid, because it uses a vertical bar instead of a colon to specify the start and end index of the sublist. The sublist is obtained by using the slicing operation, which uses square brackets and a colon to get a part of the list. For example, the_list[1:3] returns [1, 1], which is the sublist of the list from the index 1 to the index 3, excluding the end index. However, the_list |1:3 | will raise a SyntaxError exception and output nothing.

C) len (the list [0:2]} <3: This expression is trying to check if the length of a sublist of the list is less than 3. This expression is valid, because it uses the len function and the slicing operation correctly. The len function is used to return the number of values in a list or a sublist. For example, len(the_list) returns 4, because the list has four values. The slicing operation is used to get a part of the list by using square brackets and a colon. For example, the_list[0:2] returns ['1', 1], which is the sublist of the list from the index 0 to the index 2, excluding the end index. The expression len (the list [0:2]} <3 returns True, because the length of the sublist ['1', 1] is 2, which is less than 3.

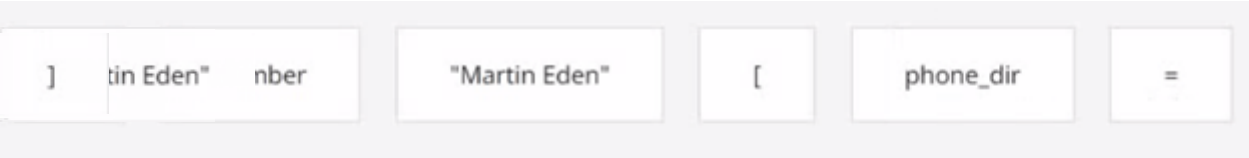
D) the_list. index {'1'} -- 0: This expression is trying to check if the index of the value '1' in the list is equal to 0. This expression is valid, because it uses the index method and the equality operator correctly. The index method is used to return the first occurrence of a value in a list. For example, the_list.index('1') returns 0, because '1' is the first value in the list. The equality operator is used to compare two values and return True if they are equal, or False if they are not. For example, 0 == 0 returns True, and 0 == 1 returns False. The expression the_list. index {'1'} -- 0 returns True, because the index of '1' in the list is 0, and 0 is equal to 0.

Therefore, the correct answers are C. len (the list [0:2]} <3 and D. the_list. index {'1'} -- 0.

Question 4

Question Type: DragDrop

Assuming that the phonc_dir dictionary contains namemumber 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.



Answer:

Question 5

Question Type: MultipleChoice

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

Options:

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:

A, D

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 tuplet = ('a', 'b', 'c'), thent[0]returns'a', andt[-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 tuplet = ('a', 'b', 'c', 'd', 'e'), thent[2]returns'c', andt[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 tuplet = (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 tuplet = ('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 tuplet = '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 tuplet = ('a', 'b', 'c'), thenlen(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 tuplet = ()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 itemt = ('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.

Question 6

Question Type: MultipleChoice

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.)

Options:

A- my_list[-10]

B- my_list|my_Li1st | 3| I

C- my list [6]

D- my_List- [0:1]

Answer:

B, D

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| I: 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| I: 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.

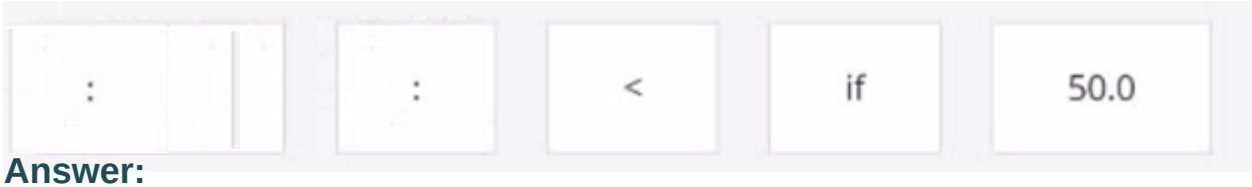
D) my_List- [0:1]: This expression is a valid Python code that uses the slicing operation to get a sublist of the list. The slicing operation does not raise any exception, even if the start or end index is out of range. It will just return an empty list or the closest possible sublist. For example, my_list[0:10] 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| I and D. my_List- [0:1].

Question 7

Question Type: DragDrop

Arrange the code boxes in the correct positions to form a conditional instruction which guarantees that a certain statement is executed when the speed variable is less than 50.0.

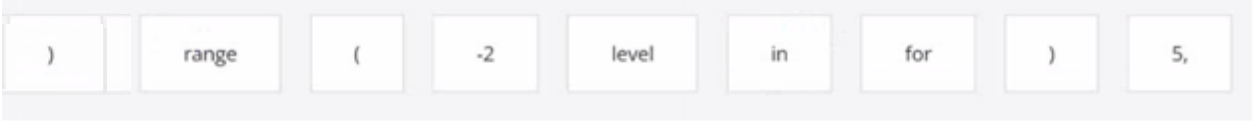


Answer:

Question 8

Question Type: DragDrop

Arrange the code boxes in the correct positions in order to obtain a loop which executes its body with the level variable going through values 5, 1, and 1 (in the same order).



Answer:

Question 9

Question Type: MultipleChoice

What happens when the user runs the following code?

```
total = 0
for i in range(4):
    if 2 * i < 4:
        total += 1
else:
    total += 1
print(total)
```

Options:

A- The code outputs 3.

B- The code outputs 2.

C- The code enters an infinite loop.

D- The code outputs 1.

Answer:

B

Explanation:

The code snippet that you have sent is calculating the value of a variable "total" based on the values in the range of 0 to 3. The code is as follows:

total = 0 for i in range(0, 3): if i % 2 == 0: total = total + 1 else: total = total + 2 print(total)

The code starts with assigning the value 0 to the variable "total". Then, it enters a for loop that iterates over the values 0, 1, and 2 (the range function excludes the upper bound). Inside the loop, the code checks if the current value of "i" is even or odd using the modulo operator (%). If "i" is even, the code adds 1 to the value of "total". If "i" is odd, the code adds 2 to the value of "total". The loop ends when "i" reaches 3, and the code prints the final value of "total" to the screen.

The code outputs 2 to the screen, because the value of "total" changes as follows:

When i = 0, total = 0 + 1 = 1

When i = 1, total = 1 + 2 = 3

When i = 2, total = 3 + 1 = 4

When i = 3, the loop ends and total = 4 is printed

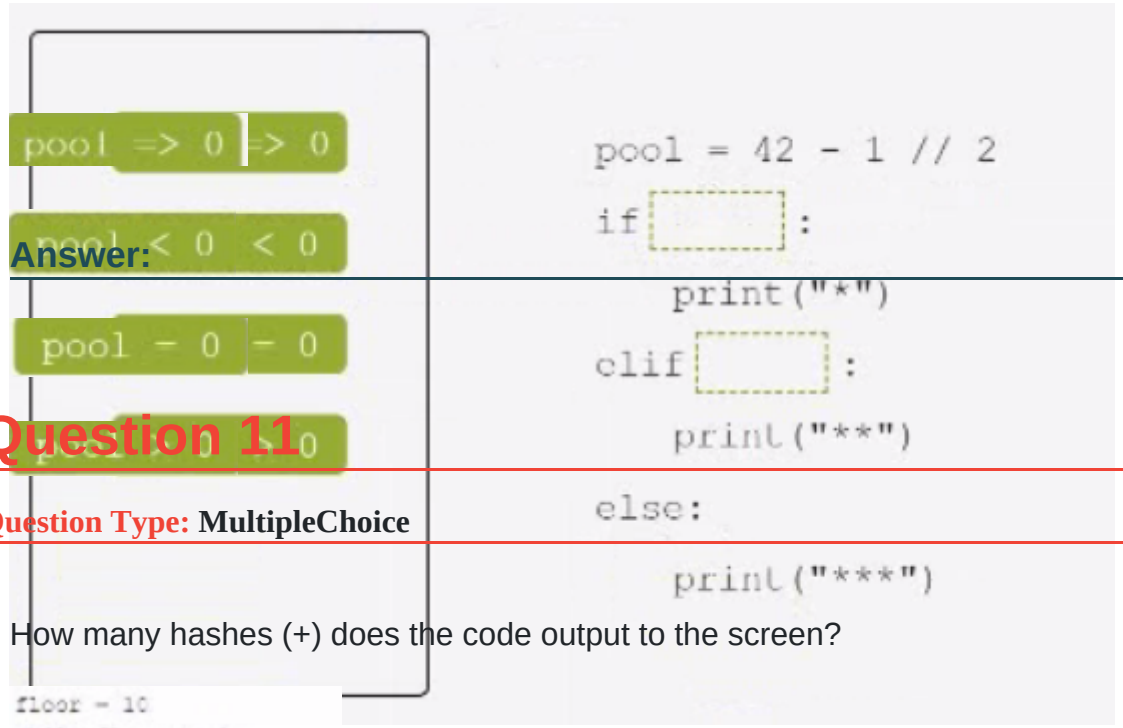
Therefore, the correct answer is B. The code outputs 2.

Question 10

Question Type: **DragDrop**

Drag and drop the conditional expressions to obtain a code which outputs * to the screen.

(Note: some code boxes will not be used.)



Answer:

Question 11

Question Type: MultipleChoice

How many hashes (+) does the code output to the screen?

```
floor = 10
while floor != 0:
    floor //= 4
    print("#", end="")
else:
    print("#")
```

Options:

A- one

B- zero (the code outputs nothing)

C- five

D- three

Answer:

C

Explanation:

The code snippet that you have sent is a loop that checks if a variable "floor" is less than or equal to 0 and prints a string accordingly. The code is as follows:

floor = 5 while floor > 0: print("+") floor = floor - 1

The code starts with assigning the value 5 to the variable "floor". Then, it enters a while loop that repeats as long as the condition "floor > 0" is true. Inside the loop, the code prints a "+" symbol to the screen, and then subtracts 1 from the value of "floor". The loop ends when "floor" becomes 0 or negative, and the code exits.

The code outputs five "+" symbols to the screen, one for each iteration of the loop. Therefore, the correct answer is C. five.