Number: PCEP-30-02
Passing Score: 800.0
Time Limit: 120.0
File Version: 7.0

Exam Code: PCEP-30-02

Exam Name: PCEP – Certified Entry-Level Python Programmer

Exam A

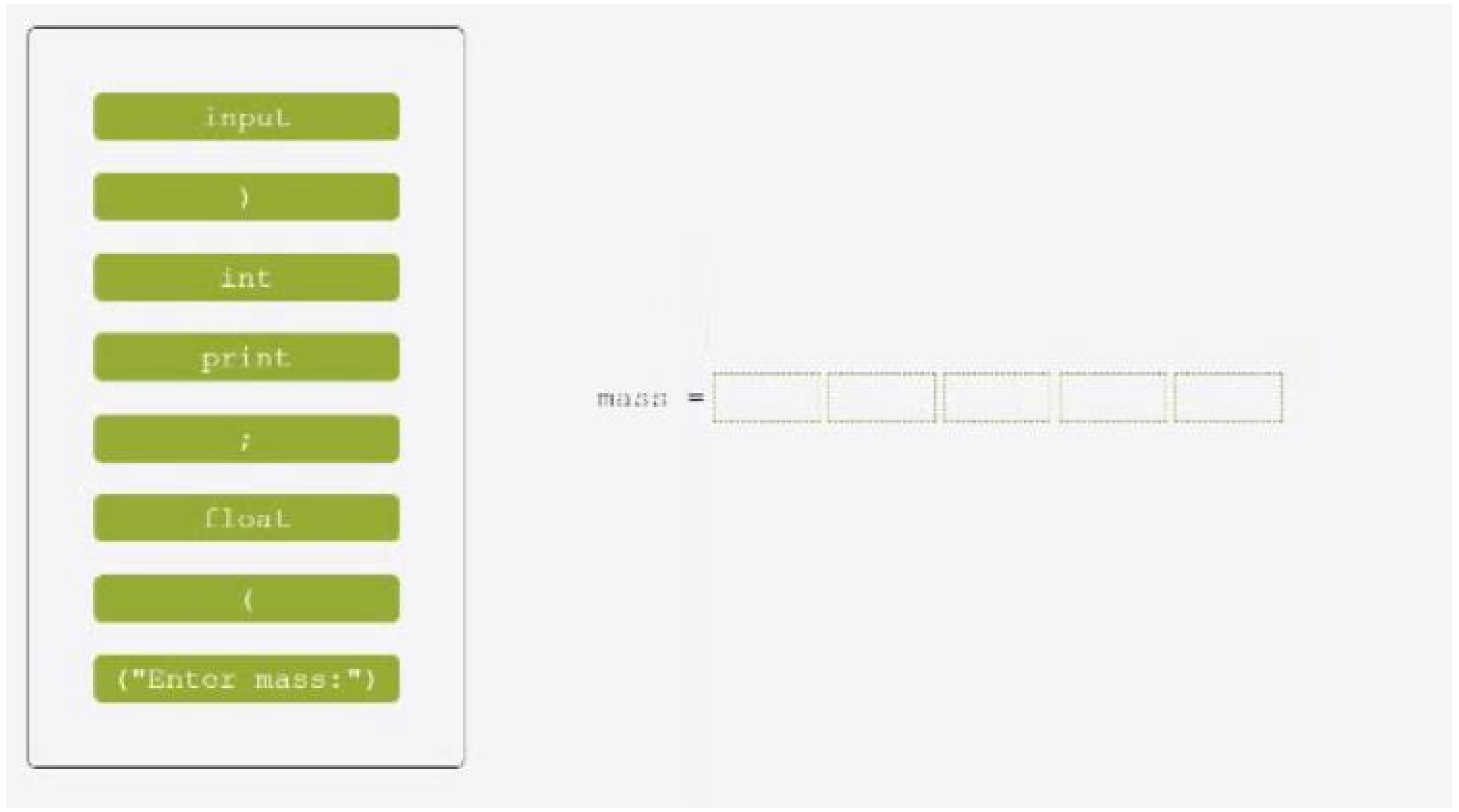
QUESTION 1
Which of the following expressions evaluate to a non-zero result? (Select two answers.)

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

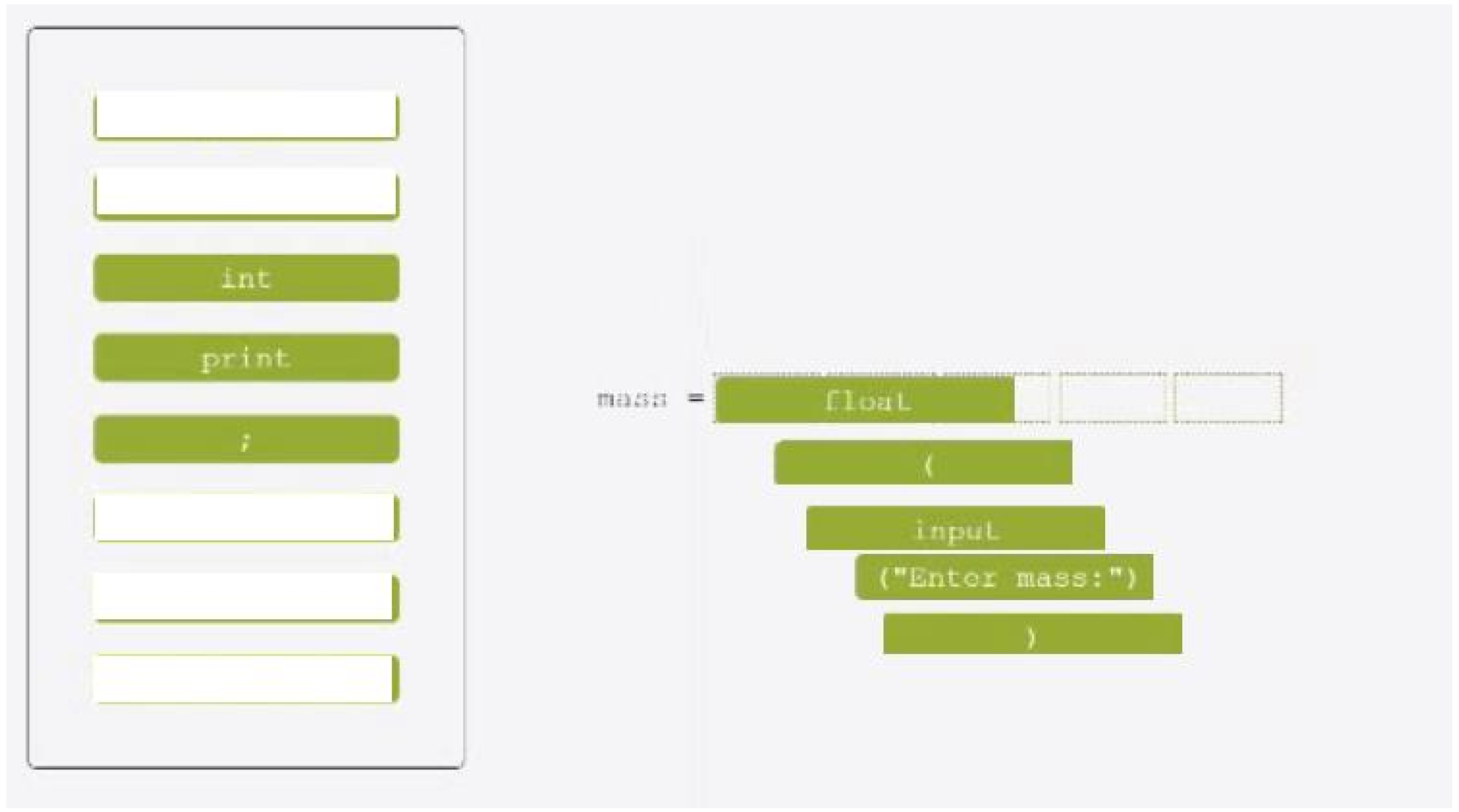
Correct Answer: A, B
Section:
Explanation:
In Python, the ** operator is used for exponentiation, the / operator is used for floating-point division, and the // operator is used for integer division. The order of operations is parentheses, exponentiation, multiplication/division, and addition/subtraction. 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.

QUESTION 2
DRAG DROP
Insert the code boxes in the correct positions in order to build a line of code which asks the user for a float value and assigns it to the mass variable.
(Note: some code boxes will not be used.)

Select and Place:



Correct Answer:



Section:
Explanation:

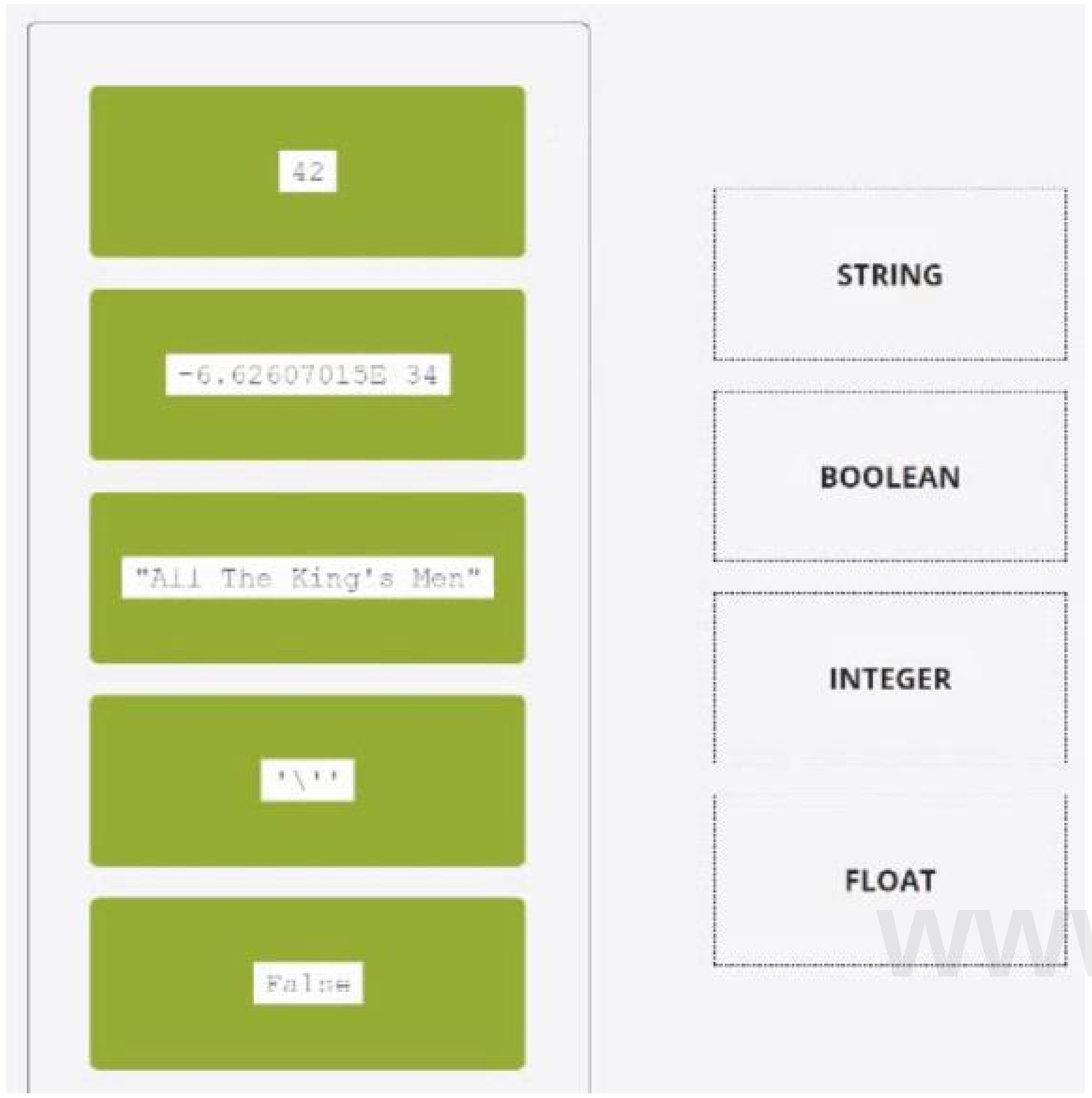
QUESTION 3
Python Is an example of which programming language category?

A. interpreted
B. assembly
C. compiled
D. machine

Correct Answer: A
Section:
Explanation:
Python is an interpreted programming language, which means that the source code is translated into executable code by an interpreter at runtime, rather than by a compiler beforehand. Interpreted languages are more flexible and portable than compiled languages, but they are also slower and less efficient. Assembly and machine languages are low-level languages that are directly executed by the hardware, while compiled languages are high-level languages that are translated into machine code by a compiler before execution.

QUESTION 4
DRAG DROP
Drag and drop the literals to match their data type names.

Select and Place:



```
42
```

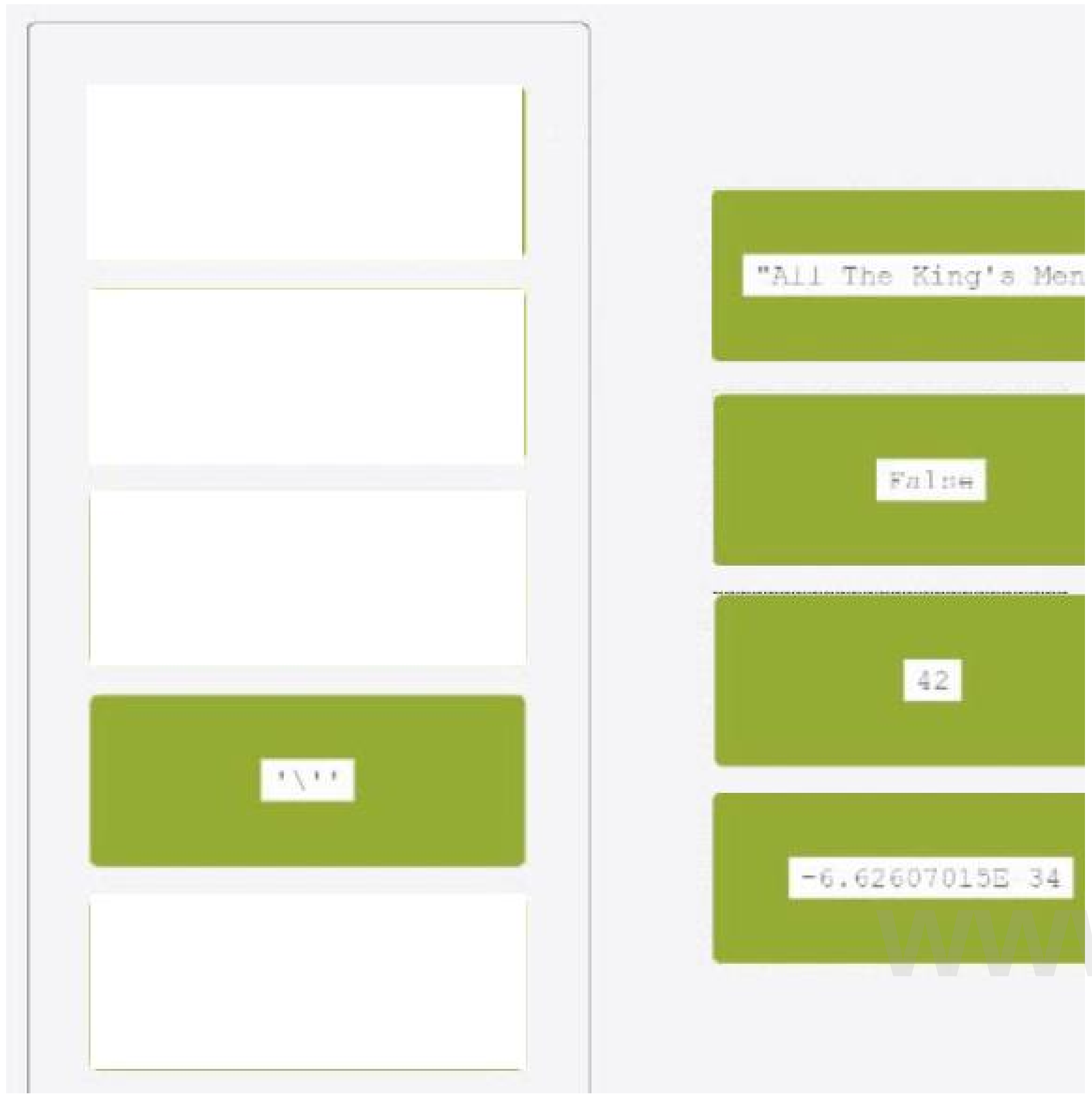
```
-6.62607015E 34
```

```
"All The King's Men"
```

```
'\''
```

```
False
```

Correct Answer:



Section:
Explanation:

QUESTION 5
How many hashes (+) does the code output to the screen?

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

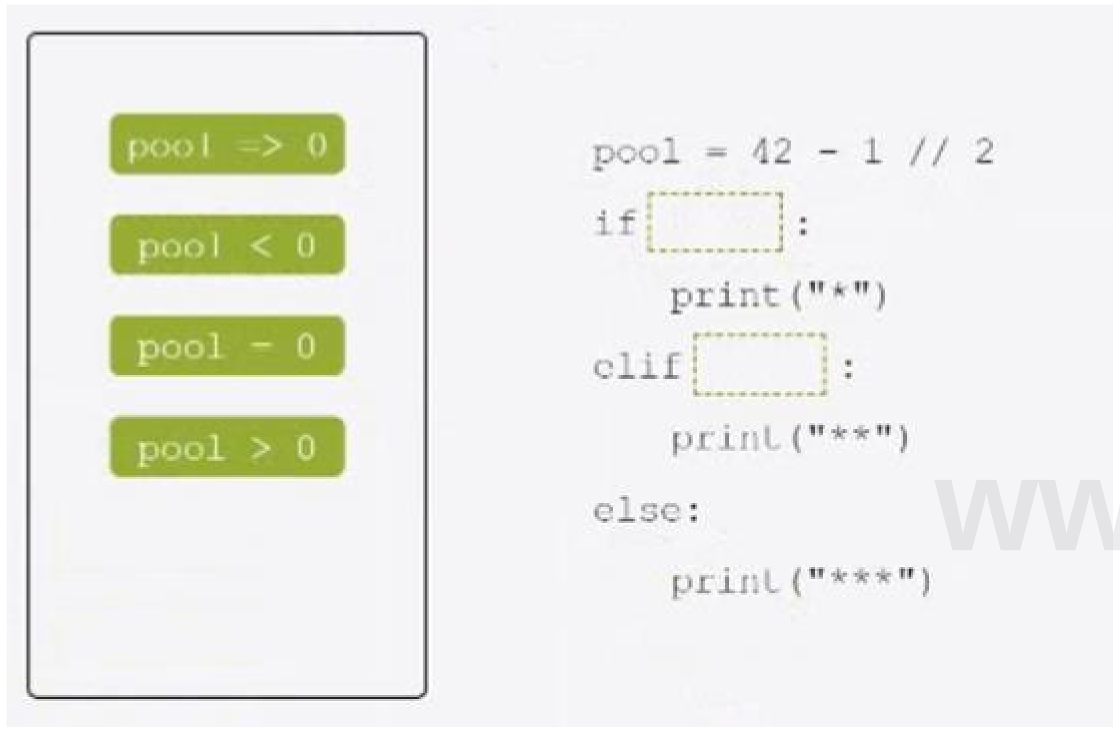
A. one
B. zero (the code outputs nothing)
C. five
D. three

Correct Answer: C
Section:
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.

QUESTION 6
DRAG DROP
Drag and drop the conditional expressions to obtain a code which outputs * to the screen.
(Note: some code boxes will not be used.)

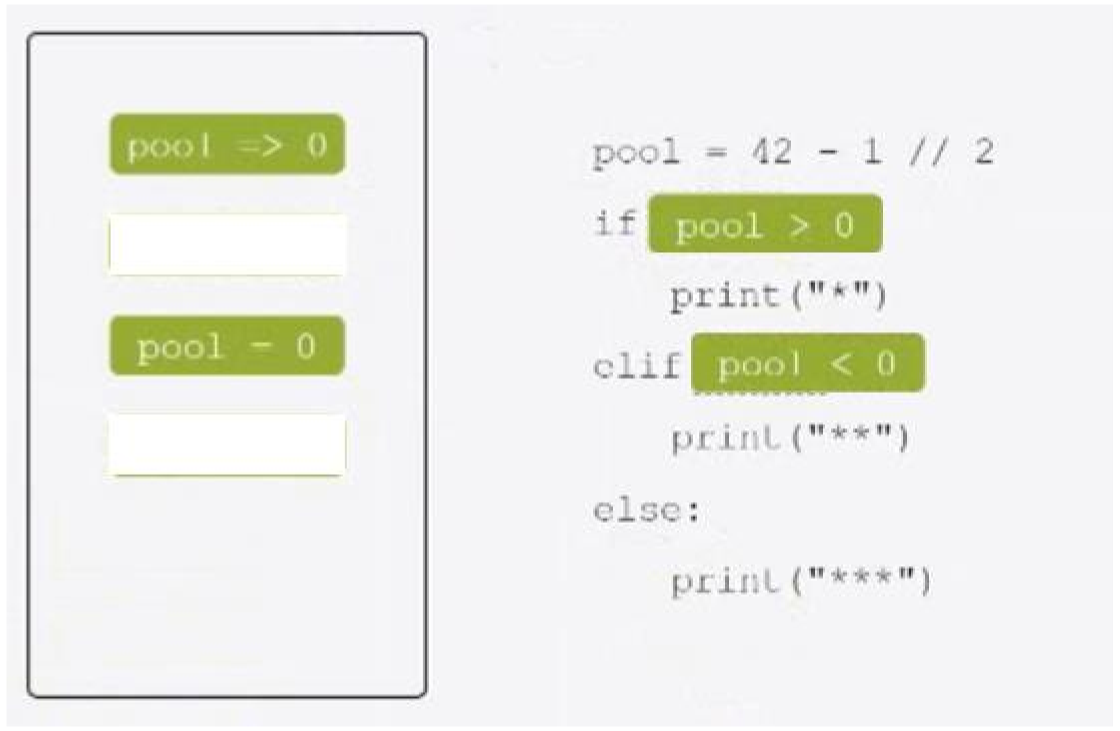
Select and Place:



```
pool => 0
pool < 0
pool == 0
pool > 0
```

```
pool = 42 - 1 // 2
if [      ]:
    print("*")
elif [      ]:
    print("**")
else:
    print("***")
```

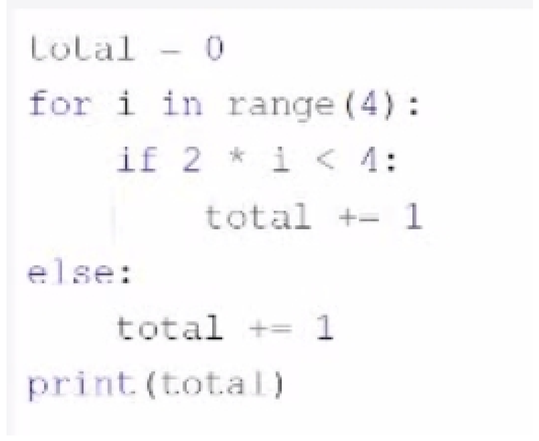
Correct Answer:



```
pool = 42 - 1 // 2
if pool > 0
    print("*")
elif pool < 0
    print("**")
else:
    print("***")
```

Section:
Explanation:

QUESTION 7
What happens when the user runs the following code?



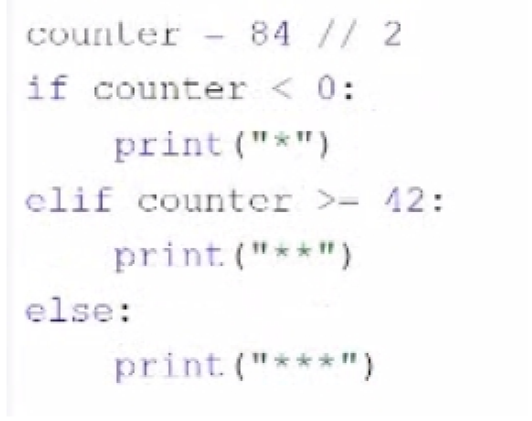
```
total - 0
for i in range(4):
    if 2 * i < 4:
        total += 1
else:
    total += 1
print(total)
```

A. The code outputs 3.
B. The code outputs 2.
C. The code enters an infinite loop.
D. The code outputs 1.

Correct Answer: B
Section:
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 8
What is the expected output of the following code?



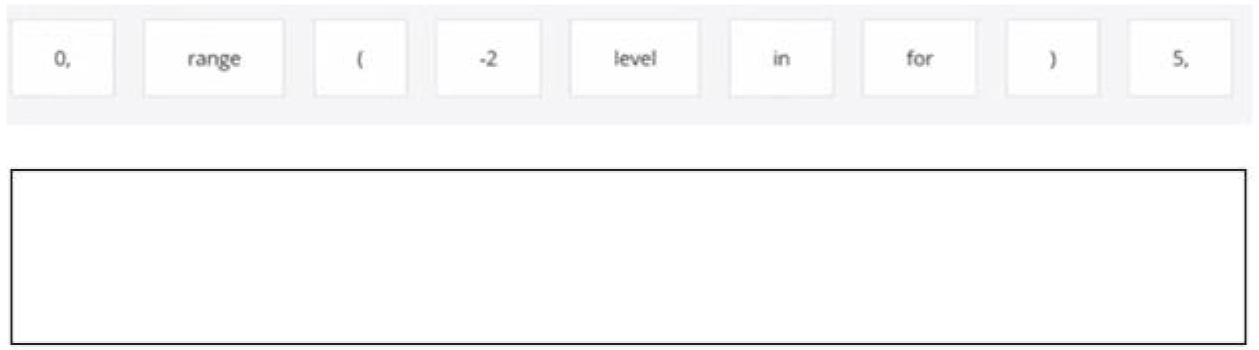
```
counter = 84 // 2
if counter < 0:
    print("*")
elif counter >= 42:
    print("**")
else:
    print("***")
```

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

Correct Answer: C
Section:
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. * *

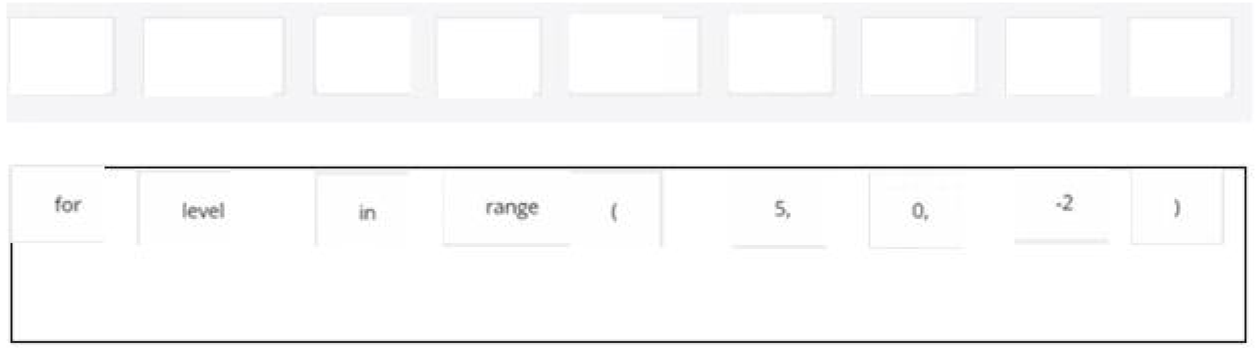
QUESTION 9
DRAG DROP
Arrange the code boxes in the correct positions in order to obtain a loop which executes its body with the level variable going through values 5, 1, and 1 (in the same order).

Select and Place:



0, range (-2 level in for) 5,

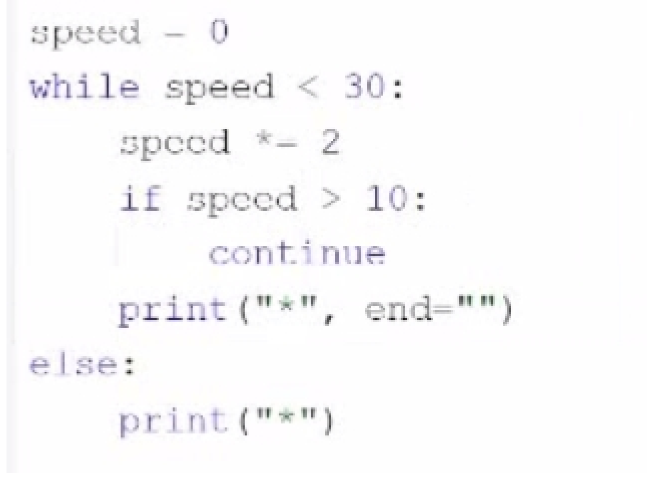
Correct Answer:



for level in range (5, 0, -2)

Section:
Explanation:

QUESTION 10
What happens when the user runs the following code?



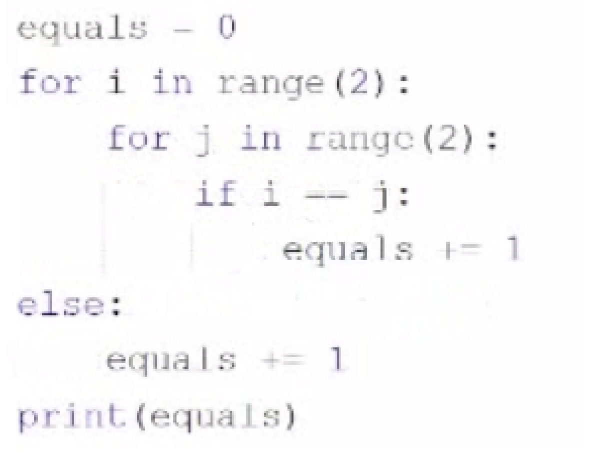
```
speed = 0
while speed < 30:
    speed *= 2
    if speed > 10:
        continue
    print("*", end="")
else:
    print("*")
```

A. The program outputs three asterisks (***)to the screen.
B. The program outputs one asterisk (*) to the screen.
C. The program outputs five asterisks (*****) to the screen.
D. The program enters an infinite loop.

Correct Answer: D
Section:
Explanation:
The code snippet that you have sent is a while loop with an if statement and a print statement inside it. The code is as follows:
while True: if counter < 0: print("") else: print("**")

The code starts with entering a while loop that repeats indefinitely, because the condition ''True'' is always true. Inside the loop, the code checks if the value of ''counter'' is less than 0. If yes, it prints a single asterisk () to the screen. If no, it prints three asterisks (**) to the screen. However, the code does not change the value of ''counter'' inside the loop, so the same condition is checked over and over again. The loop never ends, and the code enters an infinite loop.
The program outputs either one asterisk () or three asterisks (**) to the screen repeatedly, depending on the initial value of ''counter''. Therefore, the correct answer is D. The program enters an infinite loop.

QUESTION 11
What is the expected output of the following code?



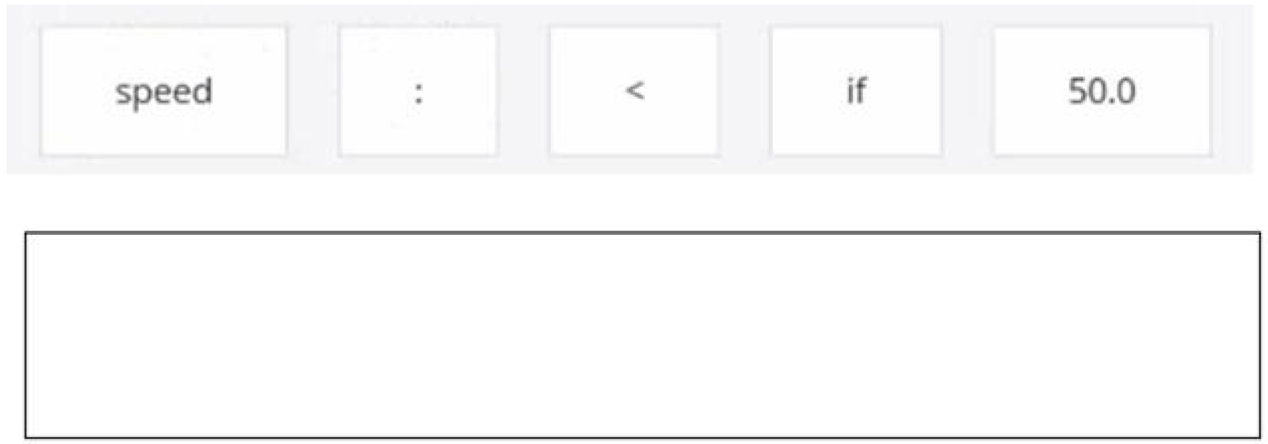
```
equals = 0
for i in range(2):
    for j in range(2):
        if i == j:
            equals += 1
else:
    equals += 1
print(equals)
```

A. The code outputs nothing.
B. 3
C. 1
D. 4

Correct Answer: C
Section:
Explanation:
The code snippet that you have sent is checking if two numbers are equal and printing the result. The code is as follows:
num1 = 1 num2 = 2 if num1 == num2: print(4) else: print(1)
The code starts with assigning the values 1 and 2 to the variables ''num1'' and ''num2'' respectively. Then, it enters an if statement that compares the values of ''num1'' and ''num2'' using the equality operator (==). If the values are equal, the code prints 4 to the screen. If the values are not equal, the code prints 1 to the screen.
The expected output of the code is 1, because the values of ''num1'' and ''num2'' are not equal. Therefore, the correct answer is C. 1.

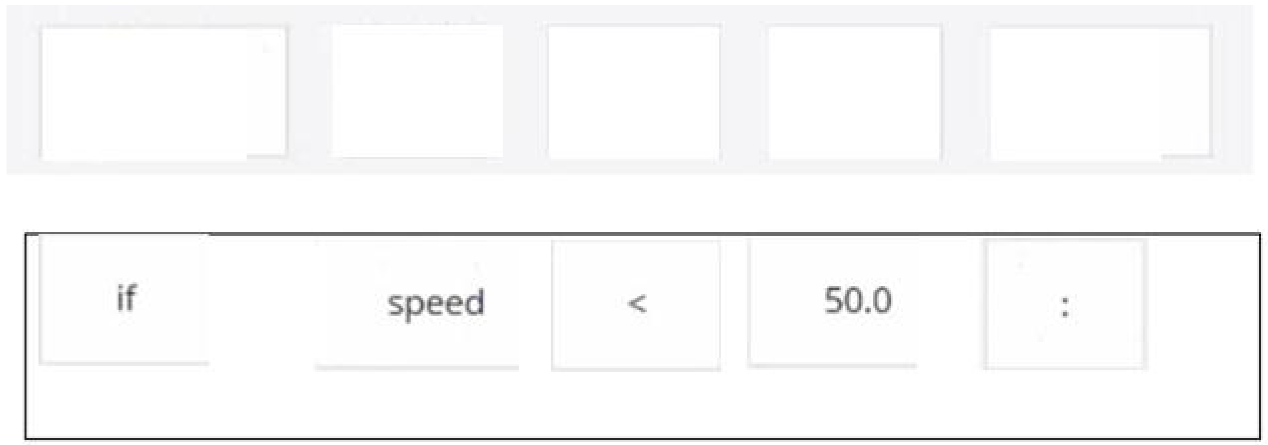
QUESTION 12
DRAG DROP
Arrange the code boxes in the correct positions to form a conditional instruction which guarantees that a certain statement is executed when the speed variable is less than 50.0.

Select and Place:



speed | : | < | if | 50.0

Correct Answer:



if speed < 50.0 :

Section:
Explanation:

QUESTION 13
What is the expected output of the following code?

```
collection = []
collection.append(1)
collection.insert(0, 2)
duplicate = collection
duplicate.append(3)
print(len(collection) + len(duplicate))
```

A. 5
B. 4
C. 6
D. The code raises an exception and outputs nothing.

Correct Answer: D
Section:
Explanation:
The code snippet that you have sent is trying to print the combined length of two lists, "collection" and "duplicate". The code is as follows:
collection = [] collection.append(1) collection.insert(0, 2) duplicate = collection duplicate.append(3) print(len(collection) + len(duplicate))

The code starts with creating an empty list called ''collection'' and appending the number 1 to it. The list now contains [1]. Then, the code inserts the number 2 at the beginning of the list. The list now contains [2, 1]. Then, the code creates a new list called ''duplicate'' and assigns it the value of ''collection''. However, this does not create a copy of the list, but rather a reference to the same list object. Therefore, any changes made to ''duplicate'' will also affect ''collection'', and vice versa. Then, the code appends the number 3 to ''duplicate''. The list now contains [2, 1, 3], and so does ''collection''. Finally, the code tries to print the sum of the lengths of ''collection'' and ''duplicate''. However, this causes an exception, because the len function expects a single argument, not two. The code does not handle the exception, and therefore outputs nothing.
The expected output of the code is nothing, because the code raises an exception and terminates. Therefore, the correct answer is D. The code raises an exception and outputs nothing.

QUESTION 14
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| I
C. my list [6]
D. my_List- [0:1]

Correct Answer: B, D
Section:
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].

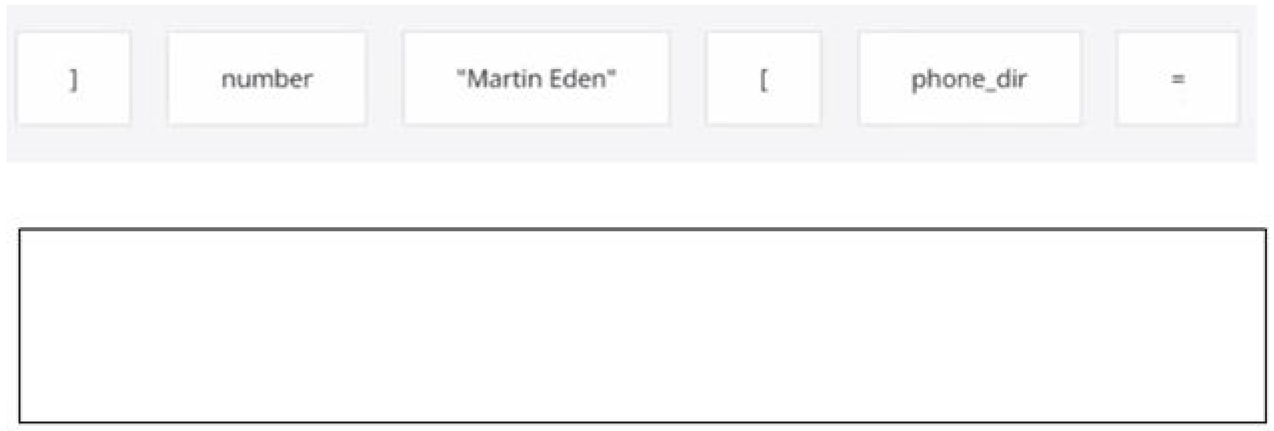
QUESTION 15
What is true about tuples? (Select two answers.)

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.

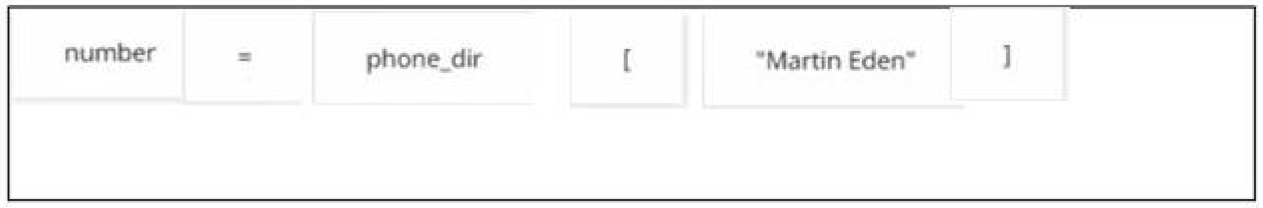
Correct Answer: A, D
Section:
Explanation:
Tuples are one of the built-in data types in Python that are used to store collections of data. Tuples have some characteristics that distinguish them from other data types, such as lists, sets, and dictionaries. Some of these characteristics are:
Tuples are 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 16
DRAG DROP
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.

Select and Place:



Correct Answer:



Section:
Explanation:

QUESTION 17
Assuming that the following assignment has been successfully executed:

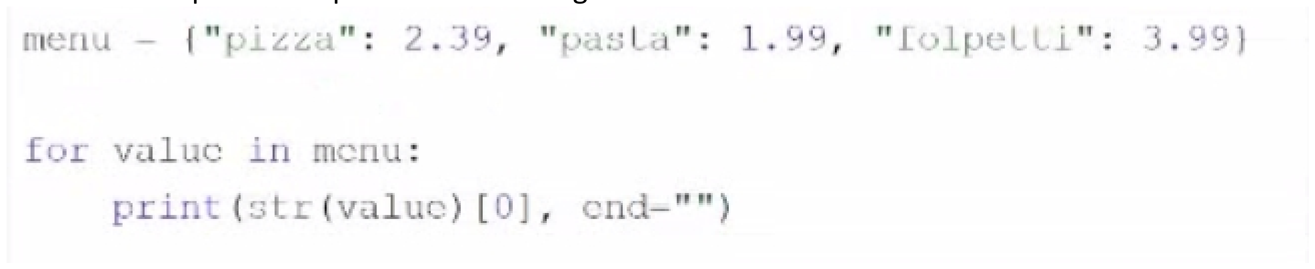
```
the_list - ["1", 1, 1.]
```

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

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

Correct Answer: C, D
Section:
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 18
What is the expected output of the following code?

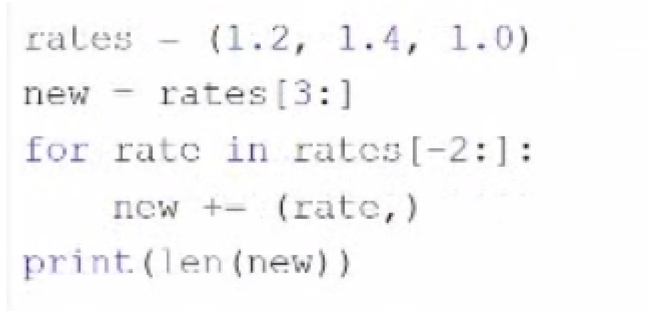

```
menu = {"pizza": 2.39, "pasta": 1.99, "folpetti": 3.99}

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

A. The code is erroneous and cannot be run.
B. ppt
C. 213
D. pizzapastafolpetti

Correct Answer: B
Section:
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.

QUESTION 19
What is the expected result of the following code?


```
rates = (1.2, 1.4, 1.0)
new = rates[3:]
for rate in rates[-2:]:
    new += (rate,)
print(len(new))
```

A. 5
B. 2
C. 1
D. The code will cause an unhandled

Correct Answer: D
Section:
Explanation:
The code snippet that you have sent is trying to use a list comprehension to create a new list from an existing list. The code is as follows:
my_list = [1, 2, 3, 4, 5] new_list = [x for x in my_list if x > 5]
The code starts with creating a list called "my_list" that contains the numbers 1, 2, 3, 4, and 5. Then, it tries to create a new list called "new_list" by using a list comprehension. A list comprehension is a concise way of creating a new list from an existing list by applying some expression or condition to each element. The syntax of a list comprehension is:
new_list = [expression for element in old_list if condition]

The expression is the value that will be added to the new list, which can be the same as the element or a modified version of it. The element is the variable that takes each value from the old list. The condition is an optional filter that determines which elements will be included in the new list. For example, the following list comprehension creates a new list that contains the squares of the even numbers from the old list:
old_list = [1, 2, 3, 4, 5, 6] new_list = [x ** 2 for x in old_list if x % 2 == 0]
new_list = [4, 16, 36]
The code that you have sent is trying to create a new list that contains the elements from the old list that are greater than 5. However, there is a problem with this code. The problem is that none of the elements in the old list are greater than 5, so the condition is always false. This means that the new list will be empty, and the expression will never be evaluated. However, the expression is not valid, because it uses the variable x without defining it. This will cause a NameError exception, which is an error that occurs when a variable name is not found in the current scope. 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 use an undefined variable in an expression that is never executed. Therefore, the correct answer is D. The code will cause an unhandled exception.