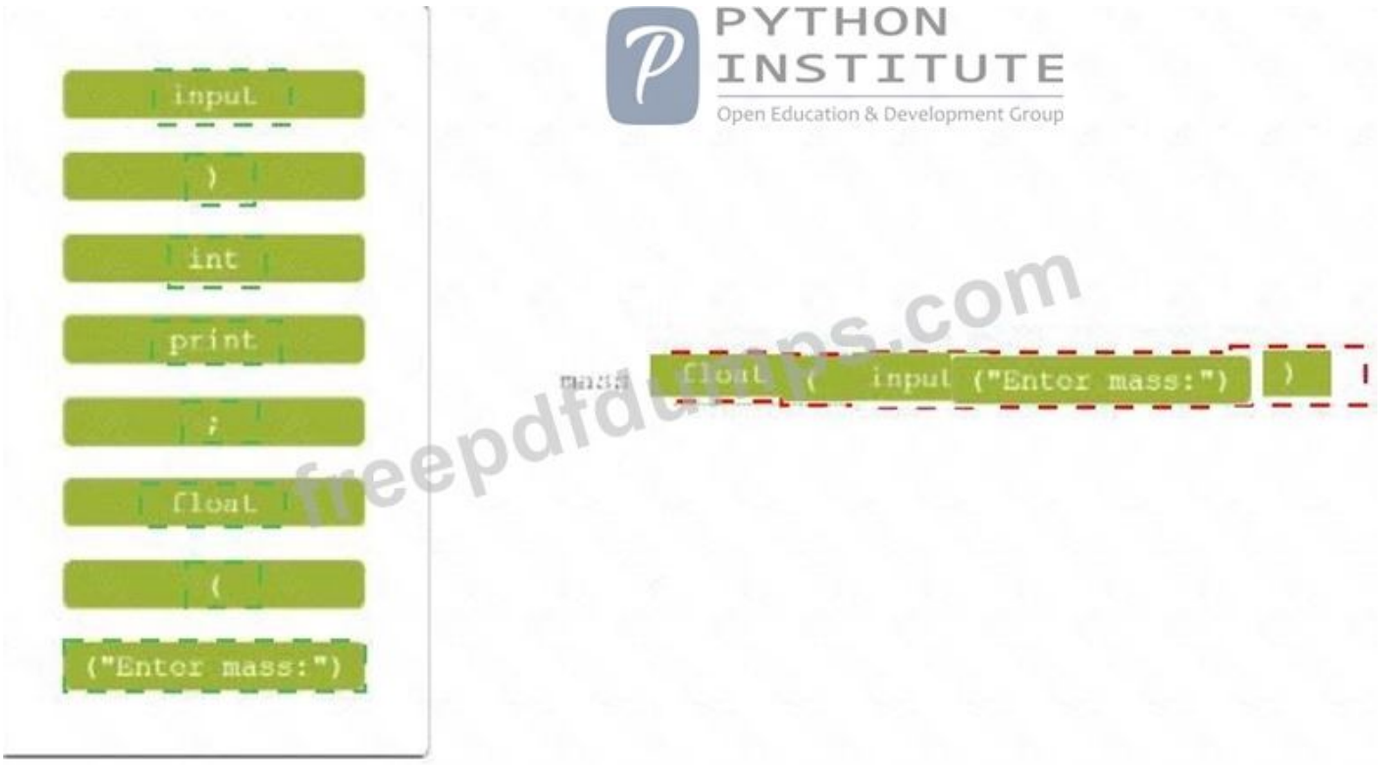
PythonInstitute.PCEP-30-02.v2023-12-20.q12

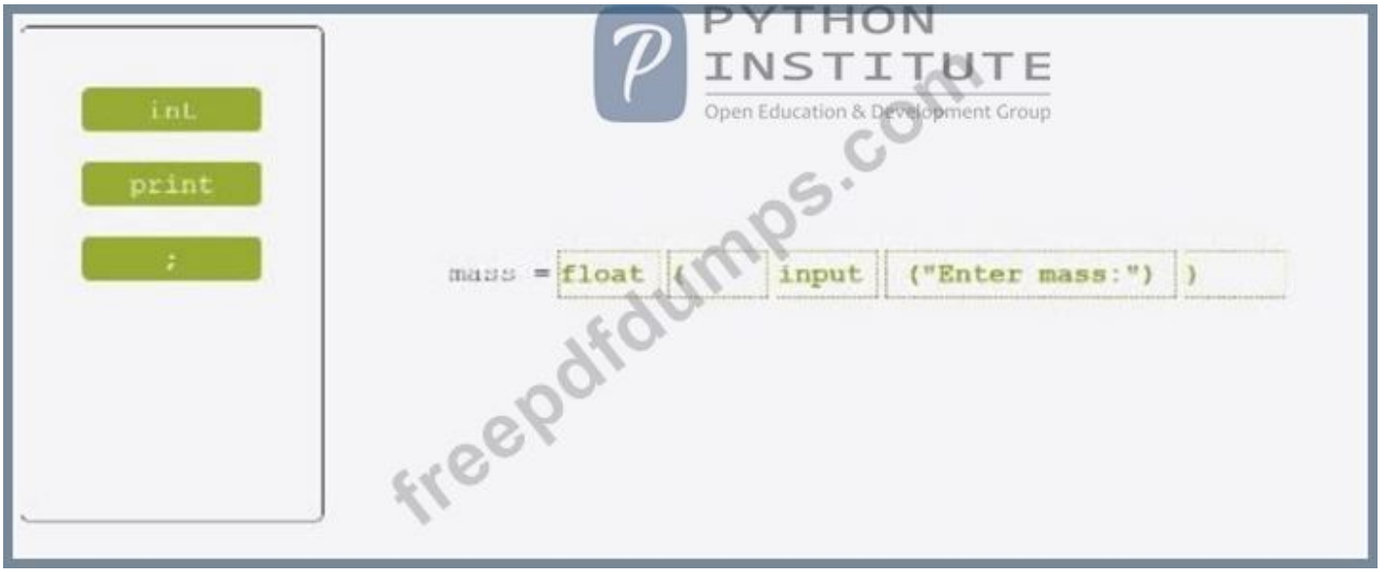
Exam Code:	PCEP-30-02
Exam Name:	PCEP - Certified Entry-Level Python Programmer
Certification Provider:	Python Institute
Free Question Number:	12
Version:	v2023-12-20
# of views:	2099
# of Questions views:	120
https://www.freepdfdumps.com/PythonInstitute.PCEP-30-02.v2023-12-20.q12.html	

NEW QUESTION: 1
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.)

Answer:



Explanation



One possible way to insert the code boxes in the correct positions in order to build a line of code that asks the user for a float value and assigns it to the mass variable is:

mass = float(input("Enter the mass:

This line of code uses the input function to prompt the user for a string value, and then uses the float function to convert that string value into a floating-point number. The result is then assigned to the variable mass.

You can find more information about the input and float functions in Python in the following references:

[Python input() Function]

[Python float() Function]

NEW QUESTION: 2

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 prints 1 .

B. The code prints 2

C. The code raises an unhandled exception.

D. The code prints 0

Answer: (SHOW ANSWER)

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.

NEW QUESTION: 3

Which of the following functions can be invoked with two arguments?

A.
```
def mu(None):
    pass
```
B.
```
def iota(level, size = 0):
    pass
```
C.
```
def kappa(level):
    pass
```
D.
```
def lambda():
    pass
```
Answer: B (LEAVE A REPLY)

Explanation

The code snippets that you have sent are defining four different functions in Python. A function is a block of code that performs a specific task and can be reused in the program. A function can take zero or more arguments, which are values that are passed to the function when it is called. A function can also return a value or None, which is the default return value in Python.

To define a function in Python, you use the def keyword, followed by the name of the function and parentheses. Inside the parentheses, you can specify the names of the parameters that the function will accept.

After the parentheses, you use a colon and then indent the code block that contains the statements of the function. For example:

def function_name(parameter1, parameter2): # statements of the function return value To call a function in Python, you use the name of the function followed by parentheses. Inside the parentheses, you can pass the values for the arguments that the function expects. The number and order of the arguments must match the number and order of the parameters in the function definition, unless you use keyword arguments or default values. For example:

function_name(argument1, argument2)

The code snippets that you have sent are as follows:

A) def my_function(): print("Hello")

B) def my_function(a, b): return a + b

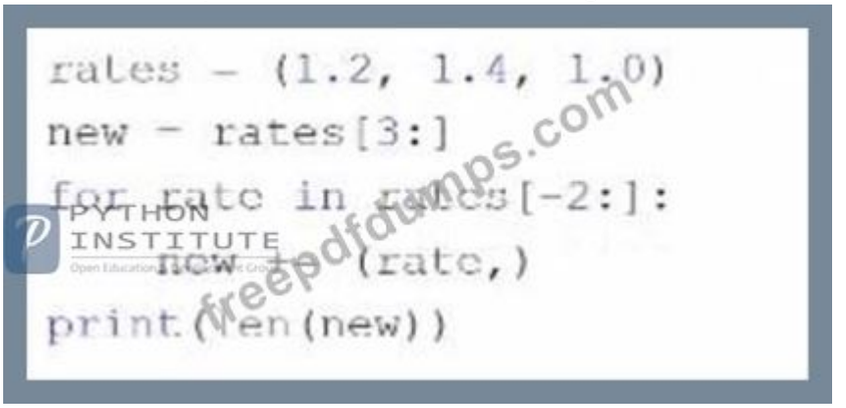
C) def my_function(a, b, c): return a * b * c

D) def my_function(a, b=0): return a - b

The question is asking which of these functions can be invoked with two arguments. This means that the function must have two parameters in its definition, or one parameter with a default value and one without.

The default value is a value that is assigned to a parameter if no argument is given for it when the function is called. For example, in option D, the parameter b has a default value of 0, so the function can be called with one or two arguments.
The only option that meets this criterion is option B. The function in option B has two parameters, a and b, and returns the sum of them. This function can be invoked with two arguments, such as my_function(2, 3), which will return 5.
The other options cannot be invoked with two arguments. Option A has no parameters, so it can only be called with no arguments, such as my_function(), which will print "Hello". Option C has three parameters, a, b, and c, and returns the product of them. This function can only be called with three arguments, such as my_function(2, 3, 4), which will return 24. Option D has one parameter with a default value, b, and one without, a, and returns the difference of them. This function can be called with one or two arguments, such as my_function(2) or my_function(2, 3), which will return 2 or -1, respectively.
Therefore, the correct answer is B. Option B.

NEW QUESTION: 4
What is the expected result of the following code?



A. 5
B. 2
C. 1
D. The code will cause an unhandled
Answer: D (LEAVE A REPLY)
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.

NEW QUESTION: 5
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.
Answer: D (LEAVE A REPLY)
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.

NEW QUESTION: 6
What is the expected result of the following code?

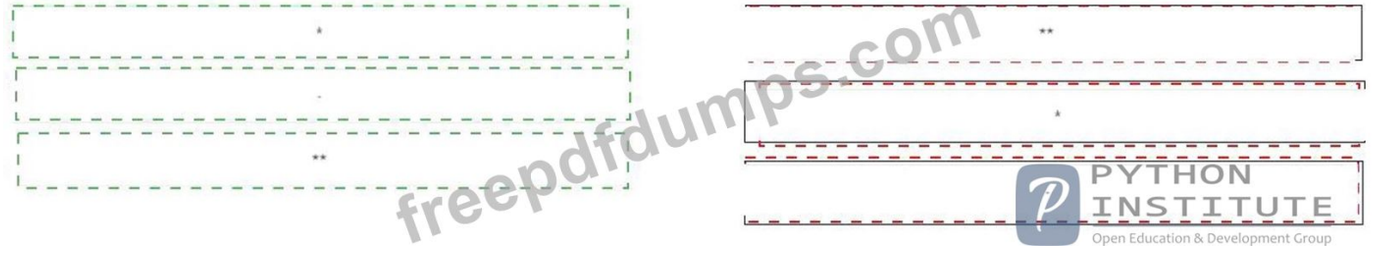
```
def velocity(x=10):
    return speed + x


speed = 10
new_speed = velocity()
new_speed = velocity(new_speed)
print(new_speed)
```


A. The code is erroneous and cannot be run.
B. 20
C. 10
D. 30
Answer: (SHOW ANSWER)
Explanation
The code snippet that you have sent is trying to use the global keyword to access and modify a global variable inside a function. The code is as follows:
speed = 10 def velocity(): global speed speed = speed + 10 return speed print(velocity()) The code starts with creating a global variable called "speed" and assigning it the value 10. A global variable is a variable that is defined outside any function and can be accessed by any part of the code. Then, the code defines a function called "velocity" that takes no parameters and returns the value of "speed" after adding 10 to it. Inside the function, the code uses the global keyword to declare that it wants to use the global variable

"speed", not a local one. A local variable is a variable that is defined inside a function and can only be accessed by that function. The global keyword allows the function to modify the global variable, not just read it. Then, the code adds 10 to the value of "speed" and returns it. Finally, the code calls the function "velocity" and prints the result.
However, the code has a problem. The problem is that the code uses the global keyword inside the function, but not outside. The global keyword is only needed when you want to modify a global variable inside a function, not when you want to create or access it outside a function. If you use the global keyword outside a function, you will get a SyntaxError exception, which is an error that occurs when the code does not follow the rules of the Python language. 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 uses the global keyword incorrectly. Therefore, the correct answer is A. The code is erroneous and cannot be run.

NEW QUESTION: 7
Arrange the binary numeric operators in the order which reflects their priorities, where the top-most position has the highest priority and the bottom-most position has the lowest priority.

Answer:

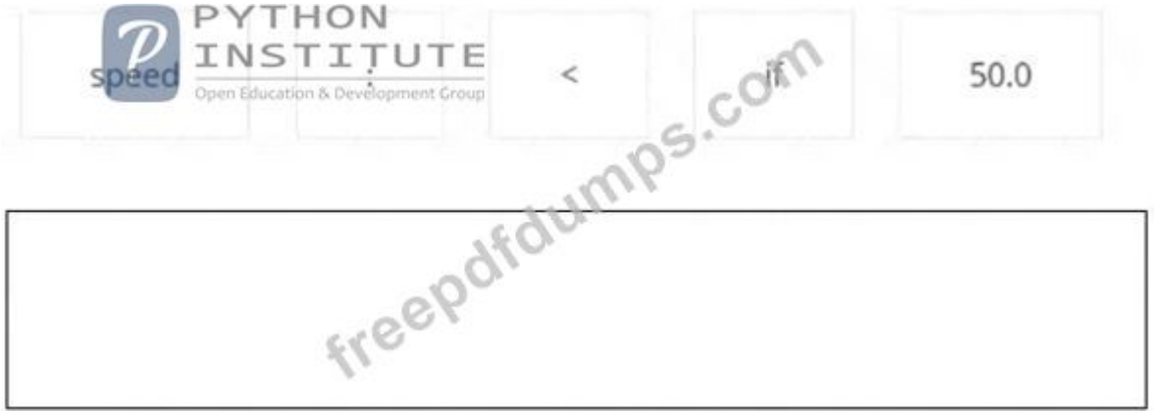


Explanation

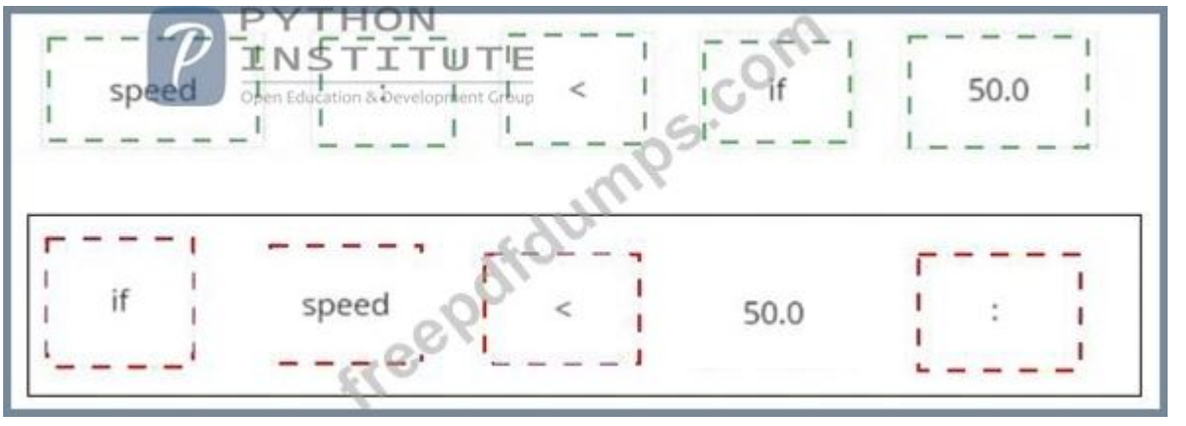
The correct order of the binary numeric operators in Python according to their priorities is:
Exponentiation (**)

Multiplication (*) and Division (
Addition (+) and Subtraction (
This order follows the standard mathematical convention of operator precedence, which can be remembered by the acronym PEMDAS (Parentheses, Exponents, Multiplication/Division, Addition/Subtraction). Operators with higher precedence are evaluated before those with lower precedence, but operators with the same precedence are evaluated from left to right.
Parentheses can be used to change the order of evaluation by grouping expressions.
For example, in the expression 2 + 3 * 4 ** 2, the exponentiation operator (**) has the highest priority, so it is evaluated first, resulting in 2 + 3 * 16. Then, the multiplication operator (*) has the next highest priority, so it is evaluated next, resulting in 2 + 48. Finally, the addition operator (+) has the lowest priority, so it is evaluated last, resulting in 50.
You can find more information about the operator precedence in Python in the following references:
6. Expressions - Python 3.11.5 documentation
Precedence and Associativity of Operators in Python - Programiz
Python Operator Priority or Precedence Examples Tutorial

NEW QUESTION: 8
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:



Explanation

One possible way to 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 is:

if speed < 50.0:

print("The speed is low.")

This code uses the if keyword to create a conditional statement that checks the value of the variable speed. If the value is less than 50.0, then the code will print "The speed is low." to the screen. The print function is used to display the output. The code is indented to show the block of code that belongs to the if condition.

You can find more information about the if statement and the print function in Python in the following references:

Python If ... Else

Python Print Function

NEW QUESTION: 9

What is the expected output of the following code?

A. 1

B. The code raises an unhandled exception.

C. False

D. ('Fermi ', '2021', 'False')

Answer: D (LEAVE A REPLY)

Explanation

The code snippet that you have sent is defining and calling a function in Python. The code is as follows:

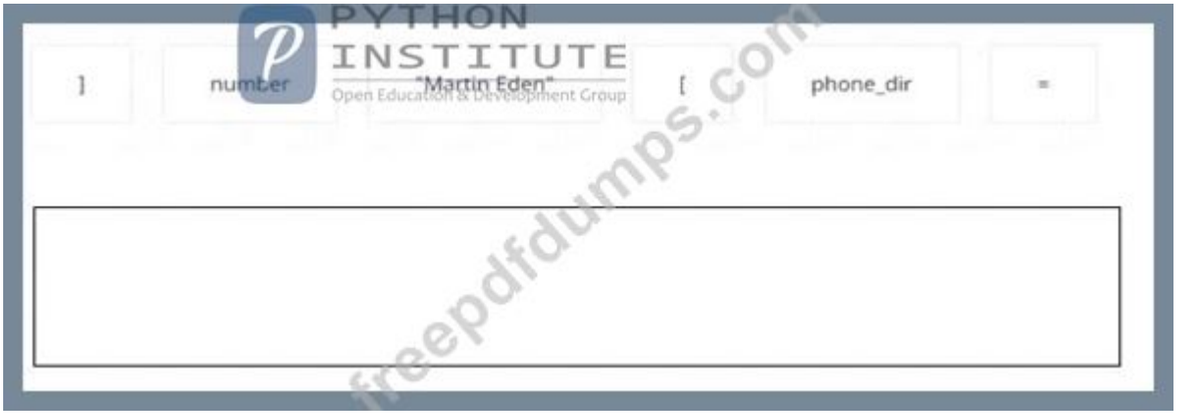
def runner(brand, model, year): return (brand, model, year)

print(runner("Fermi"))

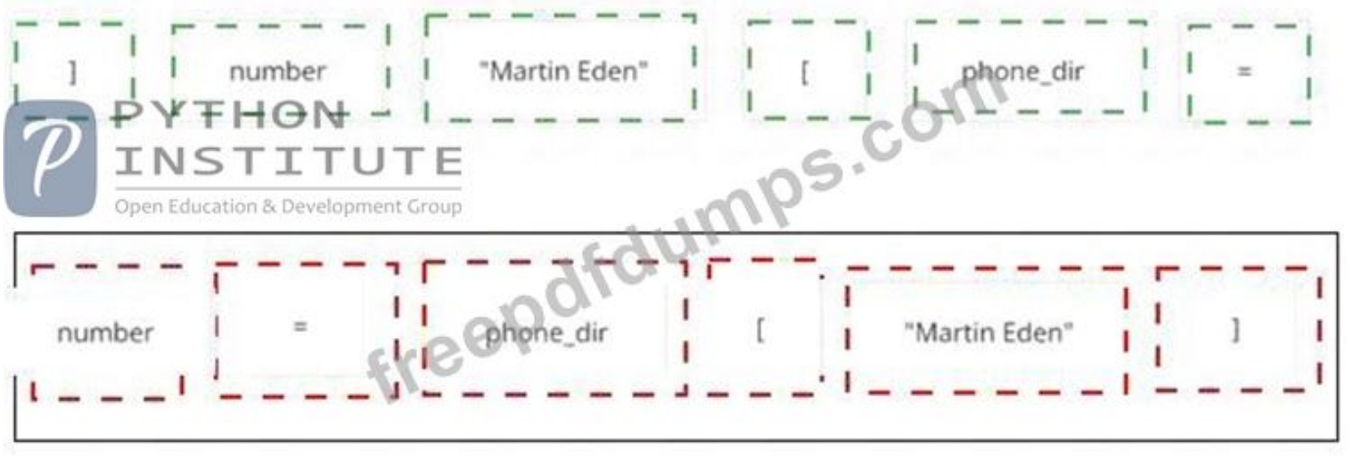
The code starts with defining a function called "runner" with three parameters: "brand", "model", and "year".

The function returns a tuple with the values of the parameters. A tuple is a data type in Python that can store multiple values in an ordered and immutable way. A tuple is created by using parentheses and separating the values with commas. For example, (1, 2, 3) is a tuple with three values.
Then, the code calls the function "runner" with the value "Fermi" for the "brand" parameter and prints the result. However, the function expects three arguments, but only one is given. This will cause a TypeError exception, which is an error that occurs when a function or operation receives an argument that has the wrong type or number. The code does not handle the exception, and therefore it will terminate with an error message.
However, if the code had handled the exception, or if the function had used default values for the missing parameters, the expected output of the code would be ('Fermi ', '2021', 'False'). This is because the function returns a tuple with the values of the parameters, and the print function displays the tuple to the screen.
Therefore, the correct answer is D. ('Fermi ', '2021', 'False').

NEW QUESTION: 10
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:



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: 11

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

Answer: (SHOW ANSWER)

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: 12

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

Answer: C,D (LEAVE A REPLY)

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.