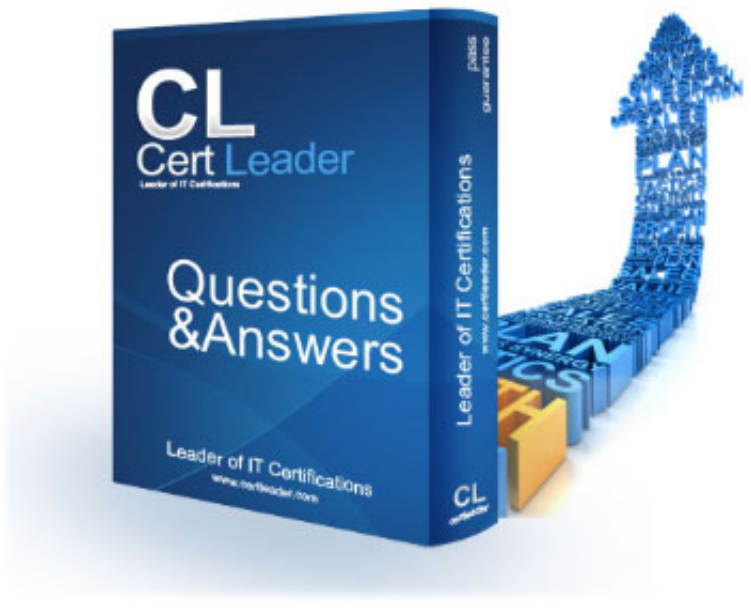
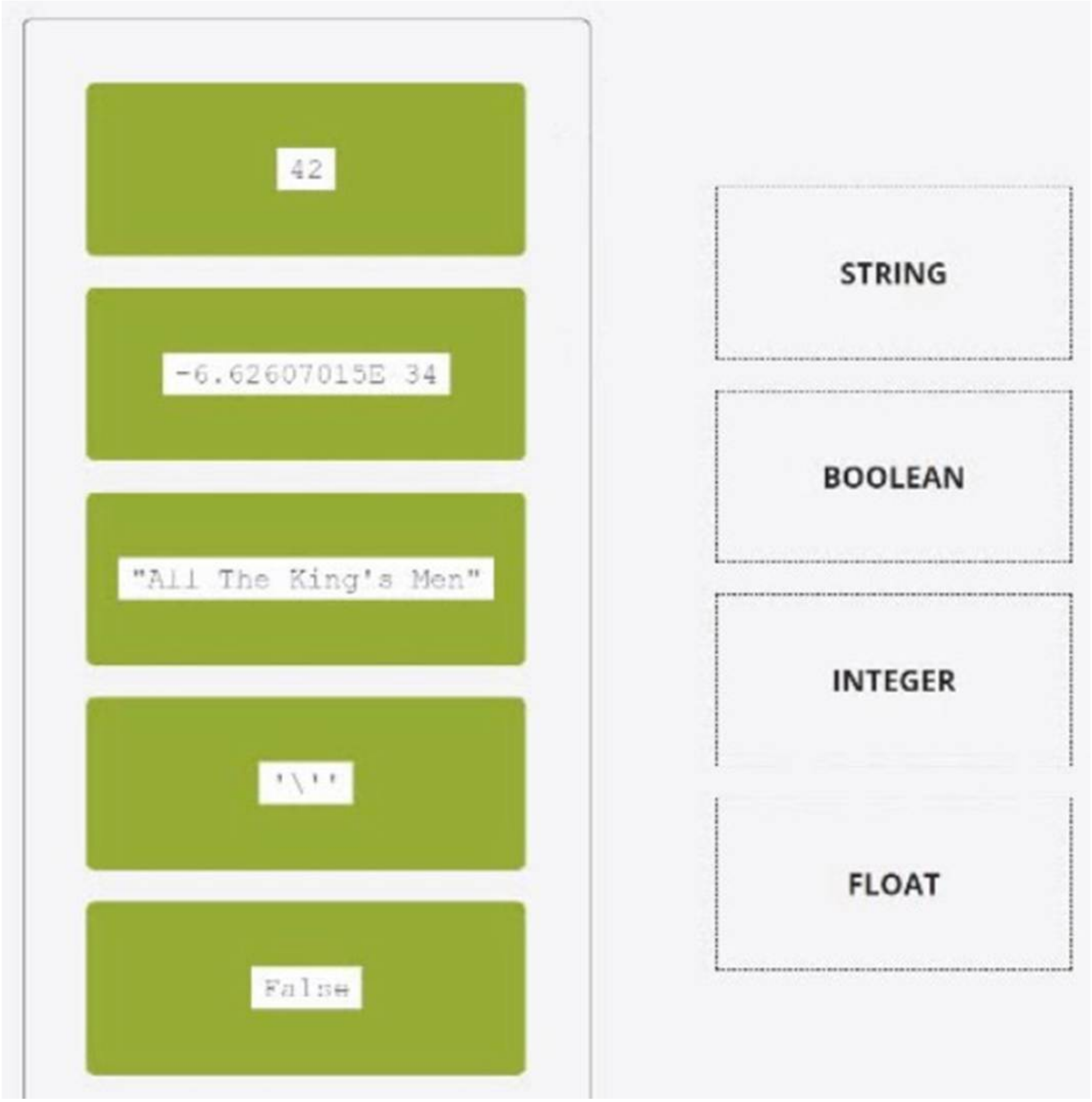
PCEP-30-02 Dumps

PCEP - Certified Entry-Level Python Programmer

https://www.certleader.com/PCEP-30-02-dumps.html



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 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

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 1. 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.
Reference: [Python Institute - Entry-Level Python Programmer Certification]

NEW QUESTION 3
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 4
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 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 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
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_lis
E. index {'1'} -- 0

Answer: CD

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. Reference: Python List Methods - W3Schools5. Data Structures — Python 3.11.5 documentationList methods in Python - GeeksforGeeks

NEW QUESTION 7
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: A

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.
Reference: Python Global Keyword - W3SchoolsPython Exceptions: An Introduction – Real
Python
The code is erroneous because it is trying to call the ??velocity?? function without passing any parameter, which will raise a TypeError exception. The ??velocity?? function requires one parameter ??x??, which is used to calculate the return value of ??speed?? multiplied by ??x??. If no parameter is passed, the function will not know what value to use for ??x??.
The code is also erroneous because it is trying to use the ??new_speed?? variable before it is defined. The ??new_speed?? variable is assigned the value of 20 after the first function call, but it is used as a parameter for the second function call, which will raise a NameError
exception. The variable should be defined before it is used in any expression or function call.
Therefore, the code will not run and will not produce any output. The correct way to write the code would be:
Define the speed variable speed = 10
Define the velocity function def velocity(x):
return speed * x
Define the new_speed variable new_speed = 20
Call the velocity function with new_speed as a parameter print(velocity(new_speed))
Copy
This code will print 200, which is the result of 10 multiplied by 20. References:
[Python Programmer Certification (PCPP) – Level 1] [Python Programmer Certification (PCPP) – Level 2] [Python Programmer Certification (PCPP) – Level 3]
[Python: Built-in Exceptions]
[Python: Defining Functions]
[Python: More on Variables and Printing]

NEW QUESTION 8
What is the expected output of the following code?

```
def runner(brand, model="", year=2021, convertible=False):
    return (brand, str(year), str(convertible))


print(runner("Fermi")[2][2])
```

A. 1
B. The code raises an unhandled exception.
C. False
D. ('Fermi ', '2021', 'False')

Answer: D

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??).
Reference: Python Functions - W3SchoolsPython Tuples - W3SchoolsPython Exceptions: An Introduction – Real Python

NEW QUESTION 9
Which of the following functions can be invoked with two arguments?
A)

```
def mu(None):
    pass
```

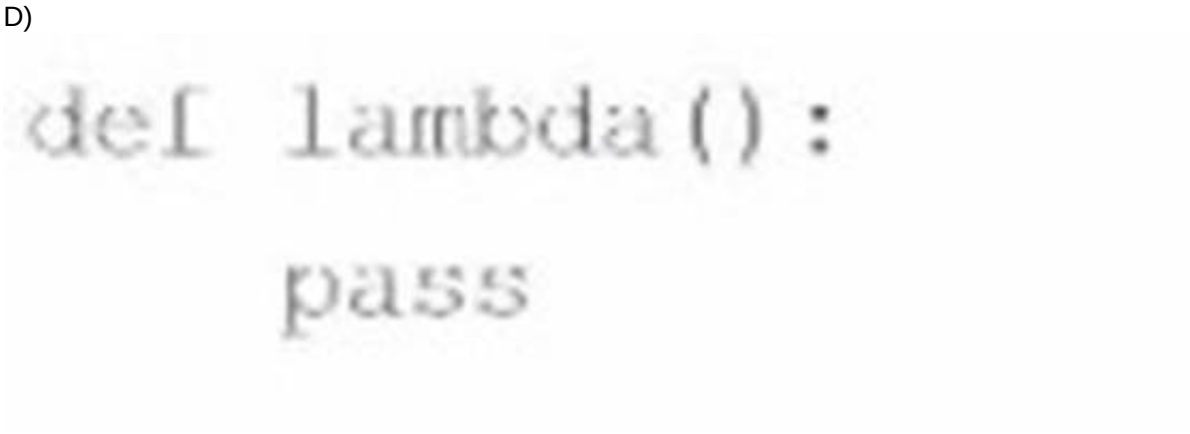
B)

```
def iota(level, size = 0):
    pass
```

C)

```
def kappa(level):
    pass
```

D)



A. Option A
B. Option B
C. Option C
D. Option D

Answer: B

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 10
.....

Thank You for Trying Our Product

* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

* One year free update

You can enjoy free update one year. 24x7 online support.

* Trusted by Millions

We currently serve more than 30,000,000 customers.

* Shop Securely

All transactions are protected by VeriSign!

100% Pass Your PCEP-30-02 Exam with Our Prep Materials Via below:

https://www.certleader.com/PCEP-30-02-dumps.html