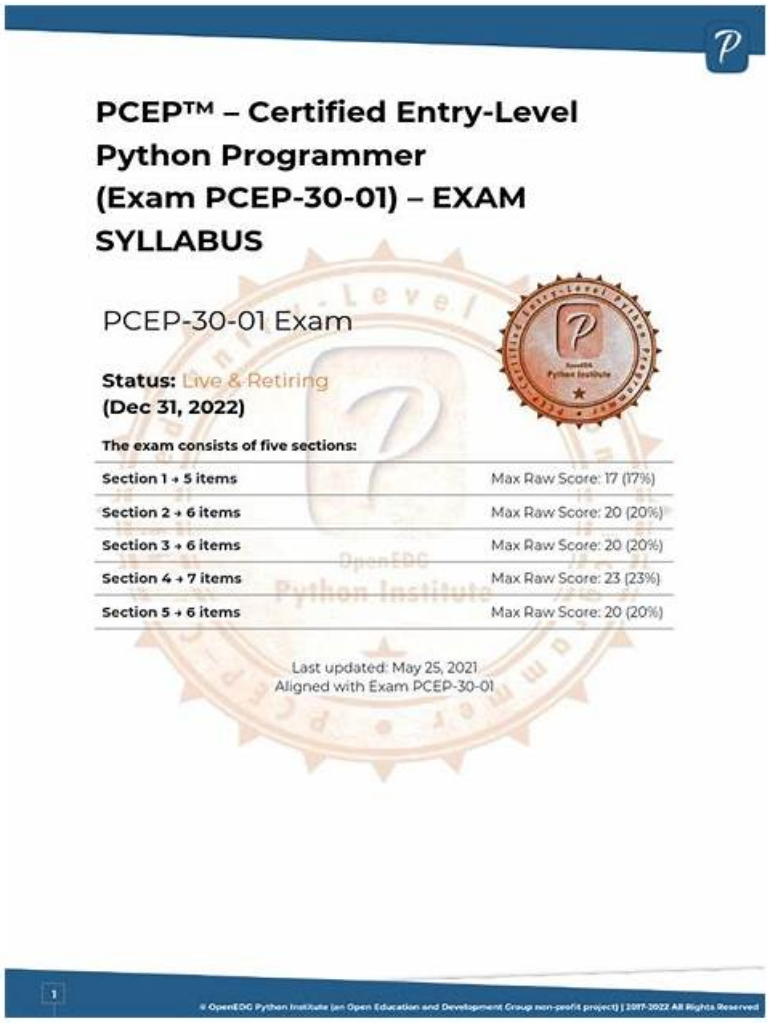
Free PDF PCEP-30-02 - PCEP - Certified Entry-Level Python Programmer–Reliable Exam Tips


PCEP™ – Certified Entry-Level Python Programmer (Exam PCEP-30-01) – EXAM SYLLABUS

PCEP-30-01 Exam

Status: Live & Retiring (Dec 31, 2022)

The exam consists of five sections:

Section 1 • 5 items	Max Raw Score: 17 (17%)
Section 2 • 6 items	Max Raw Score: 20 (20%)
Section 3 • 6 items	Max Raw Score: 20 (20%)
Section 4 • 7 items	Max Raw Score: 23 (23%)
Section 5 • 6 items	Max Raw Score: 20 (20%)

Last updated: May 25, 2021
Aligned with Exam PCEP-30-01

What's more, part of that ActualVCE PCEP-30-02 dumps now are free: https://drive.google.com/open?id=1RXljd7R0R9INbck8z-tNgPLTk_hEJBq0

Dear candidates, have you thought to participate in any Python Institute PCEP-30-02 exam training courses? In fact, you can take steps to pass the certification. ActualVCE Python Institute PCEP-30-02 Exam Training materials bear with a large number of the exam questions you need, which is a good choice. The training materials can help you pass the certification.

Python Institute PCEP-30-02 Exam Syllabus Topics:

Topic	Details
Topic 1	• Control Flow: This section covers conditional statements such as if, if-else, if-elif, if-elif-else
Topic 2	• Functions and Exceptions: This part of the exam covers the definition of function and invocation
Topic 3	• Data Collections: In this section, the focus is on list construction, indexing, slicing, methods, and comprehensions; it covers Tuples, Dictionaries, and Strings.

Topic 4	• Computer Programming Fundamentals: This section of the exam covers fundamental concepts such as interpreters, compilers, syntax, and semantics. It covers Python basics: keywords, instructions, indentation, comments in addition to Booleans, integers, floats, strings, and Variables, and naming conventions. Finally, it covers arithmetic, string, assignment, bitwise, Boolean, relational, and Input • output operations.
Topic 5	• parameters, arguments, and scopes. It also covers Recursion, Exception hierarchy, Exception handling, etc.

Test PCEP-30-02 Registration & PCEP-30-02 Latest Dumps Ebook

The PCEP - Certified Entry-Level Python Programmer (PCEP-30-02) certification has become a basic requirement to advance rapidly in the information technology sector. Since PCEP - Certified Entry-Level Python Programmer (PCEP-30-02) actual dumps are vital to prepare quickly for the examination. Therefore, you will need them if you desire to ace the PCEP - Certified Entry-Level Python Programmer (PCEP-30-02) exam in a short time.

Python Institute PCEP - Certified Entry-Level Python Programmer Sample Questions (Q37-Q42):

NEW QUESTION # 37
What is the expected output of the following code?

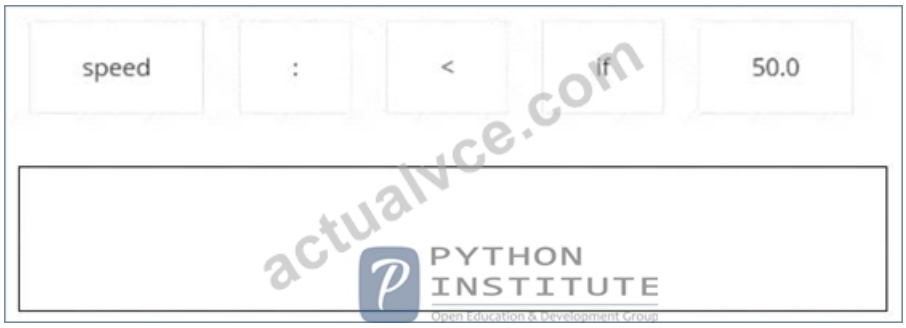
```
equals = 0
for i in range(2):
    for j in range(2):
        if i == j:
            equals += 1
else:
    equals += 1
print(equals)
```


- A. 0
- B. The code outputs nothing.
- C. 1
- D. 2

Answer: A

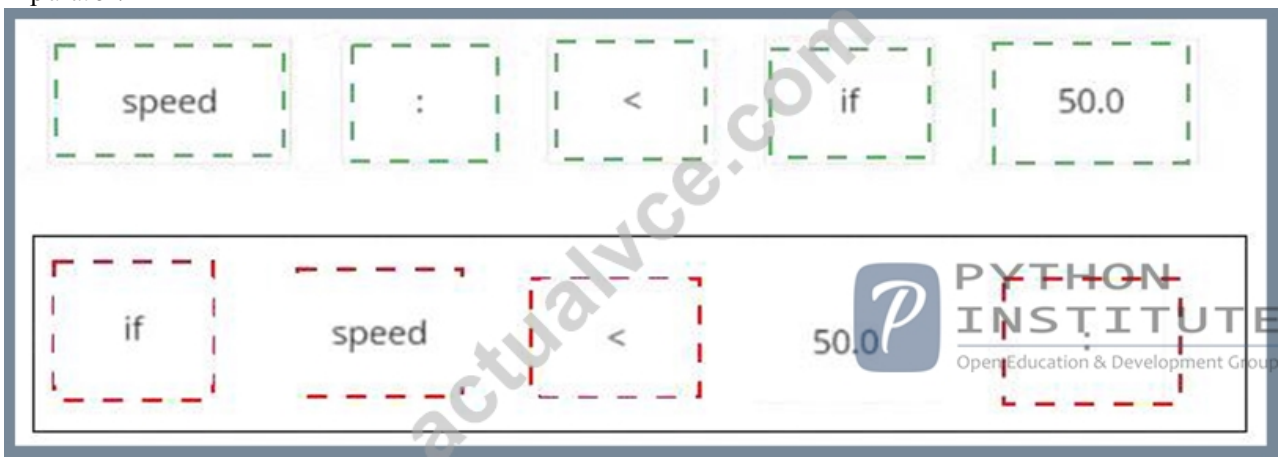
Explanation:
Explanation
The code snippet that you have sent is checking if two numbers are equal and printing the result. The code is as follows:
num1 = 1 num2 = 2 if num1 == num2: print(4) else: print(1)
The code starts with assigning the values 1 and 2 to the variables "num1" and "num2" respectively. 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.

NEW QUESTION # 38
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:



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 # 39

What happens when the user runs the following code?

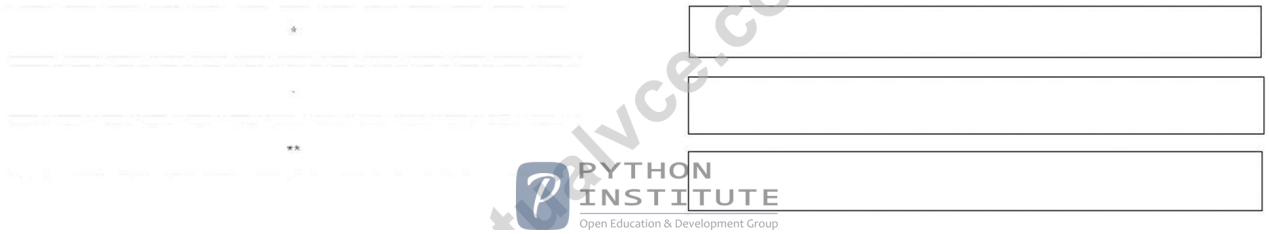
```
speed = 0
while speed < 30:
    speed *= 2
    if speed > 10:
        continue
    print("*", end="")
else:
    print("*")
```

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

Answer: A

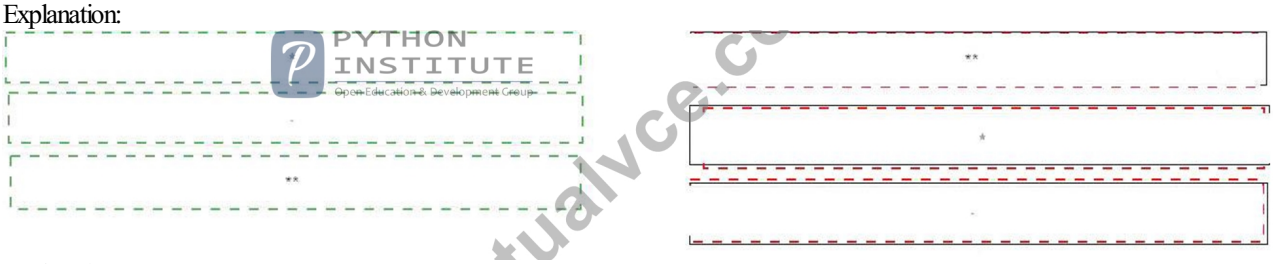
Explanation:
Explanation
The code snippet that you have sent is a while loop with an if statement and a print statement inside it. The code is as follows:
while True: if counter < 0: print("") else: print("**")
The code starts with entering a while loop that repeats indefinitely, because the condition "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 # 40
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:

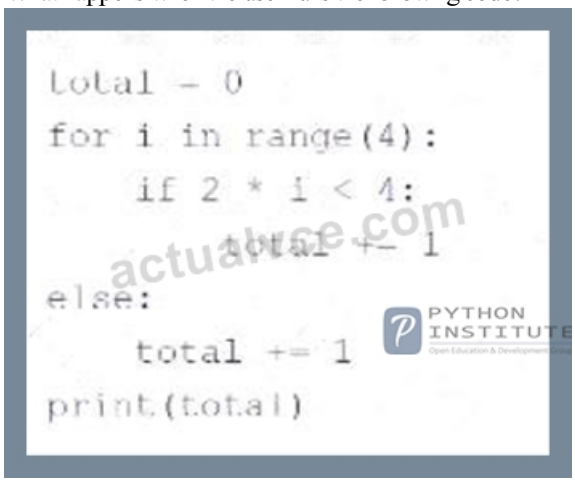
Explanation:



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 # 41
What happens when the user runs the following code?



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

Answer: C

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

NEW QUESTION # 42
......