

Python-Institute

Exam Questions PCEP-30-02

PCEP - Certified Entry-Level Python Programmer



NEW QUESTION 1

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

Answer: D

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.

Reference: Python - List Comprehension - W3Schools Python - List Comprehension -

GeeksforGeeks Python Exceptions: An Introduction – Real Python

NEW QUESTION 2

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.

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.

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

NEW QUESTION 3

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

Answer: C

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.

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

NEW QUESTION 4

A set of rules which defines the ways in which words can be coupled in sentences is called:

- A. lexis
- B. syntax
- C. semantics
- D. dictionary

Answer: B

Explanation:

Syntax is the branch of linguistics that studies the structure and rules of sentences in natural languages. Lexis is the vocabulary of a language. Semantics is the study of meaning in language. A dictionary is a collection of words and their definitions, synonyms, pronunciations, etc.

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

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

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 6

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 7

Python is an example of which programming language category?

- A. interpreted
- B. assembly

- C. compiled
- D. machine

Answer: A

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.

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

NEW QUESTION 8

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: B

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.

Reference: Python String Slicing - W3SchoolsPython String Concatenation - W3Schools

NEW QUESTION 9

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. *

Answer: C

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.

* *

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

NEW QUESTION 10

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).

0,

range

(

-2

level

in

for

)

5,

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

0,

range

(

-2

level

in

for

)

5,

for

level

in

range

(

5,

0,

-2

)

NEW QUESTION 10

DRAG DROP

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.

	*	
	-	
	**	

- A. Mastered
 B. Not Mastered

Answer: A

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 14

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.

Answer: AD

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 tuple `t = ("a", "b", "c")`, then `t[0]` returns "a", and `t[- 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 tuple `t = ("a", "b", "c", "d", "e")`, then `t[2]` returns "c", and `t[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 tuple `t = (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 tuple `t = ("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 tuple `t = "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 tuple `t = ("a", "b", "c")`, then `len(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 tuple `t = ()` 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 item `t = ("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.

Reference: Python Tuples - W3SchoolsTuples in Python - GeeksforGeeks

NEW QUESTION 16

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

PCEP-30-02 Practice Exam Features:

- * PCEP-30-02 Questions and Answers Updated Frequently
- * PCEP-30-02 Practice Questions Verified by Expert Senior Certified Staff
- * PCEP-30-02 Most Realistic Questions that Guarantee you a Pass on Your First Try
- * PCEP-30-02 Practice Test Questions in Multiple Choice Formats and Updates for 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The PCEP-30-02 Practice Test Here](#)