

Python PCEP-30-02 : Practice Test



Exam Code: PCEP-30-02

Title : Certified Entry-Level Python Programmer

Question Set

QUESTION 1

DRAG DROP

Insert the code boxes in the correct positions in order to build a line of code which asks the user for an integer value and assigns it to the depth variable.

(Note: some code boxes will not be used.)

The interface shows a list of code snippets on the left and a target code line on the right. The snippets are: `input(`, `)`, `"Enter immersion depth:"`, `=`, `int(`, `-`, and `float(`. The target line is `depth` followed by five empty boxes for code completion.

A. `depth = int(input("Enter the immersion depth: "))`

Correct Answer: A

Explanation

Explanation/Reference:

One possible way to insert the code boxes in the correct positions in order to build a line of code which asks the user for an integer value and assigns it to the depth variable is:

```
depth = int(input("Enter the immersion depth: "))
```

This line of code uses the input function to prompt the user for a string value, and then uses the int function to convert that string value into an integer number. The result is then assigned to the variable depth.

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

[Python input() Function]

[Python int() Function]

QUESTION 2

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

Correct Answer: B

Explanation

Explanation/Reference:

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

QUESTION 3**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.

| |
|----|
| ** |
| * |
| / |

Correct Answer: A

Explanation**Explanation/Reference:**

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

QUESTION 4

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: AB

Explanation

Explanation/Reference:

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.

[Python Institute - Entry-Level Python Programmer Certification]

QUESTION 5

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.)

input

)

int

print

;

float

(

("Enter mass: ")

mass =

A. `mass = float(input("Enter the mass: "))`

Correct Answer: A

Explanation

Explanation/Reference:

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]

QUESTION 6

Python is an example of which programming language category?

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

Correct Answer: A

Explanation

Explanation/Reference:

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

QUESTION 7

DRAG DROP

Drag and drop the literals to match their data type names.

42

-6.62607015E-34

"All The King's Men"

\\\\

False

STRING

BOOLEAN

INTEGER

INTEGER

FLOAT

- A. 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

Correct Answer: A
Explanation

Explanation/Reference:

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, `oeAll The Kings 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]

QUESTION 8

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

Explanation

Explanation/Reference:

The code snippet that you have sent is a loop that checks if a variable `oefloor` is less than or equal to 0 and prints a string accordingly. The code is as follows:

`floor = 5` while `floor > 0`: `print(oef+)` `floor = floor - 1` The code starts with assigning the value 5 to the variable `oefloor`. Then, it enters a while loop that repeats as long as the condition `oefloor > 0` is true. Inside the loop, the code prints a `oe+` symbol to the screen, and then subtracts 1 from the value of `oefloor`. The loop ends when `oefloor` becomes 0 or negative, and the code exits.

The code outputs five `oe+` symbols to the screen, one for each iteration of the loop. Therefore, the correct answer is C. five.

[Python Institute - Entry-Level Python Programmer Certification]

QUESTION 9

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.)

pool => 0

pool < 0

pool = 0

pool > 0

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

A.

pool = 0

pool -> 0

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

Correct Answer: A

Explanation

Explanation/Reference:

One possible way to drag and drop the conditional expressions to obtain a code which outputs * to the screen is:

```
if pool > 0:
    print("*")
```

```
elif pool < 0:  
    print("***")  
else:  
    print("****")
```

This code uses the if, elif, and else keywords to create a conditional statement that checks the value of the variable pool. Depending on whether the value is greater than, less than, or equal to zero, the

code will print a different pattern of asterisks to the screen. The print function is used to display the output. The code is indented to show the blocks of code that belong to each condition. The code will output * if the value of pool is positive, ** if the value of pool is negative, and *** if the value of pool is zero.

You can find more information about the conditional statements and the print function in Python in the following references:

[Python If | Else]

[Python Print Function]

[Python Basic Syntax]

QUESTION 10

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

Explanation

Explanation/Reference:

The code snippet that you have sent is calculating the value of a variable oetotal 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 oetotal . 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 oei is even or odd using the modulo operator (%). If oei is even, the code adds 1 to the value of oetotal . If oei is odd, the code adds 2 to the value of oetotal . The loop ends when oei reaches 3, and the code prints the final value of oetotal to the screen. The code outputs 2 to the screen, because the value of oetotal 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. [Python Institute - Entry-Level Python Programmer Certification]

QUESTION 11

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

Explanation

Explanation/Reference:

The code snippet that you have sent is a conditional statement that checks if a variable oecounter is less than 0, greater than or equal to 42, or neither. The code is as follows:

if counter < 0: print(oeoe) elif counter >= 42: print(oe) else: print() The code starts with checking if the value of oecounter 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 oecounter 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 oecounter . If the value of oecounter 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. * *

[Python Institute - Entry-Level Python Programmer Certification]

QUESTION 12

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

Correct Answer: A

Explanation

Explanation/Reference:

QUESTION 13

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

Explanation

Explanation/Reference:

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 oeTrue is always true. Inside the loop, the code checks if the value of oecounter 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 oecounter 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 oecounter . Therefore, the correct answer is D. The program enters an infinite loop.

[Python Institute - Entry-Level Python Programmer Certification]

QUESTION 14

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

Explanation

Explanation/Reference:

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 oenum1 and oenum2 respectively. Then, it enters an if statement that compares the values of oenum1 and oenum2 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 oenum1 and oenum2 are not equal.

Therefore, the correct answer is C. 1.

[Python Institute - Entry-Level Python Programmer Certification]

QUESTION 15

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.



A.



Correct Answer: A

Explanation

Explanation/Reference:

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](#)

QUESTION 16

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

Explanation

Explanation/Reference:

The code snippet that you have sent is trying to print the combined length of two lists, oecollection and oeduplicate . 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 oecollection 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 oeduplicate and assigns it the value of oecollection . However, this does not create a copy of the list, but rather a reference to the same list object. Therefore, any changes made to oeduplicate will also affect oecollection , and vice versa. Then,

the code appends the number 3 to oeduplicate . The list now contains [2, 1, 3], and so does oecollection . Finally, the code tries to print the sum of the lengths of oecollection and oeduplicate . 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. [Python Institute - Entry-Level Python Programmer Certification]

QUESTION 17

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_list[3]]
- C. my_list[6]
- D. my_list[0:1]

Correct Answer: BD

Explanation

Explanation/Reference:

The code snippet that you have sent is assigning a list of four numbers to a variable called oemy_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 oemy_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_list | 3|`: 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_list | 3|`: 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_list | 3|` and D. `my_list- [0:1]`. [Python Institute - Entry-Level Python Programmer Certification]

QUESTION 18

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: AD

Explanation

Explanation/Reference:

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. Python Tuples - W3Schools Tuples in Python - GeeksforGeeks

QUESTION 19

DRAG DROP

Assuming that the `phone_dir` dictionary contains name: number 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.

]

number

"Martin Eden"

[

phone_dir

=

A.

number

=

phone_dir

[

"Martin Eden"

]

Correct Answer: A

Explanation

Explanation/Reference:

```
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 "]

QUESTION 20

Assuming that the following assignment has been successfully executed:

The code is as follows:

```
the_list = ["1", 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: CD

Explanation

Explanation/Reference:

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 {oe1 } 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 {oe1 }` 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`. Python List Methods - W3Schools5. Data Structures " Python 3.11.5 documentationList methods in Python - GeeksforGeeks

QUESTION 21

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. ppf
- C. 213
- D. pizzapastafolpetti

Correct Answer: B
Explanation

Explanation/Reference:

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 = oepizza` `pasta = oepasta` `folpetti = (folpetti` `print(pizza[0] + pasta[0] + folpetti[0])` The code starts with assigning the strings `oepizza` , `oepasta` , and `oefolpetti` 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 `oep` . The concatenation operation is used to join two or more strings together by using the `+` operator. For example, `oea + oeb` returns `oeab` . The code prints the result of `pizza[0] + pasta[0] + folpetti[0]`, which is `oep + oep + oef` , which is `oeppf` . The expected output of the code is `ppt`, because the code prints the first characters of each string.

Therefore, the correct answer is B. `ppf`.

Python String Slicing - W3Schools Python String Concatenation - W3Schools

QUESTION 22

What is the expected result of the following code?

```
rates = (1.2, 1.4, 1.0)
```

```
new = rates [3:]
```

```
for rate in rates [-1:]:
```

```
new += (rate, )
```

```
print (len (new))
```

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

Correct Answer: D
Explanation

Explanation/Reference:

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 `oemy_list` that contains the numbers 1, 2, 3, 4, and 5. Then, it tries to create a new list called `oenew_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.

Python - List Comprehension - W3Schools Python - List Comprehension - GeeksforGeeks Python Exceptions: An Introduction " Real Python

QUESTION 23

DRAG DROP

Drag and drop the code boxes in order to build a program which prints Unavailable to the screen.

(Note: one code box will not be used.)

pass

except: KeyError:

except:

```
prices = { "pizza": 3.99, "burger": 2.99, "fries": 1.99, "soda": 1.49 }

try:
    charge = prices["ca"]
    print("Charged")
    print("Unavailable")
    print("Out of bound")
```

A.



```
prices = { "pizza": 3.99
try:
    charge = prices["calz
    print("Charged")
except KeyError:
    print("Unavailable")
except:
    print("Out of bounds"
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 24

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

Correct Answer: C

Explanation

Explanation/Reference:

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. Python List index() Method - W3Schools Python Exceptions: An Introduction " Real Python

QUESTION 25

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')

Correct Answer: D

Explanation

Explanation/Reference:

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(oeFermi ))
```

The code starts with defining a function called `runner` with three parameters: `oebrand`, `oemodel`, and `oeyear`. 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 `oeFermi` for the `oebrand` 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). Python Functions - W3Schools Python Tuples - W3Schools Python Exceptions: An Introduction " Real Python

QUESTION 26

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.

Correct Answer: AC

Explanation

Explanation/Reference:

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 tryexcept 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. Python Debugger " Python pdb - GeeksforGeeks How 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 thrown Python Try Except [Error Handling and Debugging " Programming with Python for Engineers]

QUESTION 27

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
```

Correct Answer: B

Explanation

Explanation/Reference:

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 `oeHello`. 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.

QUESTION 28

Which of the following are the names of Python passing argument styles? (Select two answers.)

- A. keyword
- B. reference
- C. indicatory
- D. positional

Correct Answer: AD

Explanation

Explanation/Reference:

Keyword arguments are arguments that are specified by using the name of the parameter, followed by an equal sign and the value of the argument. For example, `print (sep='-', end='!')` is a function call with keyword arguments. Keyword arguments can be used to pass arguments in any order, and to provide default values for some arguments¹.

Positional arguments are arguments that are passed in the same order as the parameters of the function definition. For example, `print ('Hello', 'World')` is a function call with positional arguments. Positional arguments must be passed before any keyword arguments, and they must match the number and type of the parameters of the function².

1: 5 Types of Arguments in Python Function Definitions | Built In 2: python - Whats the pythonic way to pass arguments between functions |

QUESTION 29

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

Correct Answer: A

Explanation

Explanation/Reference:

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 `oespeed` 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 `oevelocity` that takes no parameters and returns the value of `oespeed` after adding 10 to it. Inside the function, the code uses the global keyword to

declare that it wants to use the global variable `oespeed`, 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 `oespeed` and returns it. Finally, the code calls the function `oevelocity` 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.

Python Global Keyword - W3Schools Python Exceptions: An Introduction " Real Python The code is erroneous because it is trying to call the `oevelocity` function without passing any parameter, which will raise a `TypeError` exception. The `oevelocity` function requires one parameter `oex`, which is used to calculate the return value of `oespeed` multiplied by `oex`. If no parameter is passed, the function will not know what value to use for `oex`. The code is also erroneous because it is trying to use the `oenew_speed` variable before it is defined. The `oenew_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.

[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]

QUESTION 30

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. 0
- C. 3
- D. 1

Correct Answer: D

Explanation

Explanation/Reference:

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 oemy_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.

Python List count() Method - W3Schools

QUESTION 31

What is the expected output of the following code?

```

1  fruits1 = ['Apple', 'Pear', 'Banana']
2  fruits2 = fruits1
3  fruits3 = fruits1[:]
4
5  fruits2[0] = 'Cherry'
6  fruits3[1] = 'Orange'
7
8  res = 0
9
10 for i in (fruits1, fruits2, fruits3):
11     if i[0] == 'Cherry':
12         res += 1
13     if i[1] == 'Orange':
14         res += 10
15
16 print(res)

```

- A. 22
- B. 12
- C. 0
- D. 11

Correct Answer: B

Explanation

Explanation/Reference:

Result Calculation: The for loop iterates over (fruits1, fruits2, fruits3) and checks conditions:

If `i[0] == 'Cherry'`, add 1 to res.

If `i[1] == 'Orange'`, add 10 to res.

Details:

For fruits1 = ['Cherry', 'Pear', 'Banana']:

`i[0] == 'Cherry'`: True → `res += 1` → `res = 1`.

`i[1] == 'Orange'`: False → no change.

For fruits2 = ['Cherry', 'Pear', 'Banana']:

`i[0] == 'Cherry'`: True → `res += 1` → `res = 2`.

`i[1] == 'Orange'`: False → no change.

For fruits3 = ['Apple', 'Orange', 'Banana']:

`i[0] == 'Cherry'`: False → no change.

`i[1] == 'Orange'`: True → `res += 10` → `res = 12`.

Output: The final value of res is 12.

Final Output

12

QUESTION 32

Which of the following sentences are true about the code? (Choose two.)

```
1 | nums = [1, 2, 3]
2 | vals = nums
```

- A. nums and vals are different lists
- B. vals is longer than nums
- C. nums and vals are different names of the same list
- D. nums and vals have the same length

Correct Answer: CD

Explanation

Explanation/Reference:

The two true sentences about the code are:

- C. nums and vals are different names of the same list
- D. nums and vals have the same length

Explanation:

When you assign `vals = nums`, `vals` becomes a reference to the same list that `nums` refers to. Therefore, any changes made to the list through either `nums` or `vals` will affect the same underlying list. Since `vals` and `nums` are references to the same list, they both have the same length.

QUESTION 33

What is the expected output of the following code?

```
1 | data = {}
2 | data[1] = 1
3 | data['1'] = 2
4 | data[1.0] = 4
5 |
6 | res = 0
7 | for d in data:
8 |     res += data[d]
9 |
10 | print(res)
```

- A. 3
- B.
- C. 6
- D. 7

A. 3

- B. The code is erroneous.
- C. 6
- D. 7

Correct Answer: C

Explanation

Explanation/Reference:

The expected output of the given code is 6.

Here's a step-by-step explanation of how the code works:

An empty dictionary data is created.

The key 1 is assigned the value 1, so data becomes {1: 1}.

The key '1' (a string) is assigned the value 2, so data becomes {1: 1, '1': 2}.

The key 1.0 (a float) is assigned the value 4. In Python, the float 1.0 is considered equal to the integer 1, so this updates the value for the key 1 to 4. Now data becomes {1: 4, '1': 2}.

The dictionary data now contains two key-value pairs: {1: 4, '1': 2}.

The for loop iterates over the keys of the dictionary:

For the first key 1, it adds data[1] (which is 4) to res, so res becomes 4.

For the second key '1', it adds data['1'] (which is 2) to res, so res becomes 6.

Finally, the print(res) statement outputs 6.

QUESTION 34

What is the expected output of the following code?

```
1 | data = ['Peter', 'Paul', 'Mary']  
2 | print(data[int(-1 / 2)])
```

- A. Paul
- B. Mary
- C. The code is erroneous.
- D. None of the above.
- E. Peter

Correct Answer: E

Explanation

Explanation/Reference:

The expression int(-1 / 2) prints Peter because of the way integer division and type conversion work in Python. Let's break it down step by step:

Division Operation:

When you perform the division -1 / 2, the result is -0.5.

Type Conversion:

The int function converts a floating-point number to an integer by truncating the decimal part. This means it removes the fractional part without rounding.

Therefore, int(-0.5) becomes 0.

List Indexing:

In Python, list indexing starts from 0. So, data[0] refers to the first element of the list.

Combining these steps, the expression data[int(-1 / 2)] is equivalent to data[0], which accesses the first element of the list data, which is 'Peter'.

So, the reason `(-1 / 2)` prints 'Peter' is because the expression evaluates to 0, and `data[0]` refers to the first element of the list.

QUESTION 35

What is the expected output of the following code?

```
1 | data = [1, 2, 3, 4, 5, 6]
2 |
3 | for i in range(1, 6):
4 |     data[i - 1] = data[i]
5 |
6 | for i in range(0, 6):
7 |     print(data[i], end=' ')
```

- A. 1 1 2 3 4 5
- B. 2 3 4 5 6 1
- C. 2 3 4 5 6 6
- D. 1 2 3 4 5 6

Correct Answer: C

Explanation

Explanation/Reference:

`data = [1, 2, 3, 4, 5, 6]`

In the first for loop, the code updates each element of data from index 0 to 4 with the value of the next element:

for i in range(1, 6):

`data[i - 1] = data[i]`

This loop will perform the following assignments:

When i = 1: `data[0] = data[1]` → `data = [2, 2, 3, 4, 5, 6]`

When i = 2: `data[1] = data[2]` → `data = [2, 3, 3, 4, 5, 6]`

When i = 3: `data[2] = data[3]` → `data = [2, 3, 4, 4, 5, 6]`

When i = 4: `data[3] = data[4]` → `data = [2, 3, 4, 5, 5, 6]`

When i = 5: `data[4] = data[5]` → `data = [2, 3, 4, 5, 6, 6]`

In the second for loop, the code prints each element of data from index 0 to 5:

for i in range(0, 6):

`print(data[i], end=' ')`

The output of this loop will be:

2 3 4 5 6 6

QUESTION 36

What is the expected output of the following code?

```
1 | data = ['abc', 'def', 'abcde', 'efg']
2 | print(max(data))
```

- A. efg
- B. abc
- C. def
- D. The code is erroneous.
- E. abcde

F. None of the above.

Correct Answer: A

Explanation

Explanation/Reference:

The `max()` function returns the largest item in an iterable. When applied to a list of strings, it compares the strings lexicographically (i.e., based on the ASCII values of the characters). In this case, the comparison is done character by character from left to right.

Here's how the comparison works for the given list:

'abc' vs 'def' -> 'def' is larger

'def' vs 'abcde' -> 'def' is larger

'def' vs 'efg' -> 'efg' is larger

Therefore, the largest string in the list according to lexicographical order is 'efg'.

The expected output of the code is: A. efg

This is because the `max` function, when applied to a list of strings, returns the string that would appear last if the list were sorted lexicographically.

QUESTION 37

What is the output of the following snippet?

```
1 | tup = (1, ) + (1, )
2 | tup = tup + tup
3 | print(len(tup))
```

A. 2

B. 4

C. The snippet is erroneous (invalid syntax)

Correct Answer: B

Explanation

Explanation/Reference:

Initial Code:

`tup = (1,) + (1,)`

`(1,)` is a tuple with a single element, 1.

`(1,) + (1,)` concatenates two tuples, resulting in `(1, 1)`.

So now:

`tup = (1, 1)`

`tup = tup + tup`

`tup` is `(1, 1)`.

Concatenating `tup + tup` results in `(1, 1, 1, 1)`.

So now:

`tup = (1, 1, 1, 1)`

`print(len(tup))`

`len(tup)` calculates the number of elements in the tuple, which is 4.

Final Output:

4

QUESTION 38

What is the output of the following snippet?

```
1 | l1 = [1, 2, 3]
2 |
3 | for v in range(len(l1)):
4 |     l1.insert(1, l1[v])
5 |
6 | print(l1)
```

- A. [1, 2, 3, 1, 2, 3]
- B. [3, 2, 1, 1, 2, 3]
- C. [1, 1, 1, 1, 2, 3]
- D. [1, 2, 3, 3, 2, 1]

Correct Answer: C

Explanation

Explanation/Reference:

1. You start with the list `l1 = [1, 2, 3]`.

2. You iterate through the range of the length of `l1`, which initially is 3.

3. During each iteration, you insert the value at the current index `v` into the second position of the list (index 1).

Here's the detailed iteration breakdown:

- Initial `l1 = [1, 2, 3]`.

- The length of `l1` is 3, so the range is `range(3)` which is `[0, 1, 2]`.

Iteration 1 (`v = 0`):

- `l1[0]` is 1.

- Insert `1` at position `1`.

- `l1` becomes `[1, 1, 2, 3]`.

Iteration 2 (`v = 1`):

- `l1[1]` is 1.

- Insert `1` at position `1`.

- `l1` becomes `[1, 1, 1, 2, 3]`.

Iteration 3 (`v = 2`):

- `l1[2]` is 1.

- Insert `1` at position `1`.

- `l1` becomes `[1, 1, 1, 1, 2, 3]`.

Finally, the list `l1` is `[1, 1, 1, 1, 2, 3]`.

QUESTION 39

Which one of the lines should you put in the snippet below to match the expected output?

Expected output:

```
1 | [1, 2, 4, 7]
```

Code:

```
1 | list = [2, 7, 1, 4]
2 |
3 | # enter code here
4 |
5 | print(list)
```

- A. sorted(list)
- B. sort(list)
- C. list.sort()
- D. list.sorted()

Correct Answer: C

Explanation

Explanation/Reference:

To achieve the expected output [1, 2, 4, 7] from the list [2, 7, 1, 4], you should use the list.sort() method, which sorts the list in place.

Here is the complete code snippet:

Copy

```
list = [2, 7, 1, 4]
list.sort() # This sorts the list in place
print(list)
```

So, the correct line to use is:

C. list.sort()

QUESTION 40

What is the expected output of the following code?

```
1 data1 = 'a', 'b'
2 data2 = ('a', 'b')
3 print(data1 == data2)
```

- A. 0
- B. 1
- C. False
- D. True

Correct Answer: D

Explanation

Explanation/Reference:

The expected output of the given code is True.

Here's a breakdown of why:

data1 = 'a', 'b' creates a tuple with two elements, 'a' and 'b'.

data2 = ('a', 'b') also creates a tuple with two elements, 'a' and 'b'.

In Python, tuples can be created either by using parentheses or by simply separating elements with commas.

Therefore, both data1 and data2 are tuples containing the same elements in the same order.

When we compare data1 == data2, we are checking if the two tuples are equal. Since both tuples contain the same elements in the same order, the comparison returns True.

So, the expected output is:

True

QUESTION 41

Take a look at the snippet and choose one of the following statements which is true:

```
1 | nums = []
2 | vals = nums
3 | vals.append(1)
```

- A. vals is longer than nums
- B. nums and vals are of the same length
- C. nums is longer than vals

Correct Answer: B

Explanation

Explanation/Reference:

In Python, lists are mutable objects. The assignment `vals = nums` does not create a new list; it simply creates a reference to the same list object.

Therefore, both `nums` and `vals` point to the same underlying list.

When `vals.append(1)` is executed, it modifies the shared list.

As a result, both `nums` and `vals` will have the same length (and the same content) because they are essentially two names for the same object.

QUESTION 42

Insert the correct snippet to convert the `t` tuple to a dictionary named `d`.

Expected output:

```
1 | {'A': 1, 'B': 2, 'C': 3}
```

Code:

```
1 | t = (('A', 1), ('B', 2), ('C', 3))
2 | # insert code here
3 | print(d)
```

- A. `t >> d.dict`
- B. `d = dict(t)`
- C. `d.dict(t)`
- D. `d = t(dict)`

Correct Answer: B

Explanation

Explanation/Reference:

The correct snippet to convert the tuple `t` to a dictionary named `d` is:

B. `d = dict(t)`

So the full code will be:

```
t = (('A', 1), ('B', 2), ('C', 3))
```

```
d = dict(t)
```

```
print(d)
```

The expected output will be:

```
{'A': 1, 'B': 2, 'C': 3}
```

QUESTION 43

What is the expected output of the following code?

```
1 | nums = [3, 4, 5, 20, 5, 25, 1, 3]
2 | nums.pop(1)
3 | print(nums)
```

- A. [3, 1, 25, 5, 20, 5, 4]
- B. [1, 3, 4, 5, 20, 5, 25]
- C. [3, 5, 20, 5, 25, 1, 3]
- D. [1, 3, 3, 4, 5, 5, 20, 25]
- E. [3, 4, 5, 20, 5, 25, 1, 3]

Correct Answer: C

Explanation

Explanation/Reference:

The list `nums` is initialized as:
`nums = [3, 4, 5, 20, 5, 25, 1, 3]`

The method `pop(1)` is called. The `pop()` method removes the element at the specified index and returns it. Here, 1 is the index, so the element at index 1 (which is 4) is removed.
After `nums.pop(1)`, the list becomes:
`nums = [3, 5, 20, 5, 25, 1, 3]`

The list is printed using `print(nums)`.
Expected Output:
`[3, 5, 20, 5, 25, 1, 3]`

QUESTION 44

What is the expected output of the following code?

```
1 | data = (1,) * 3
2 | data[0] = 2
3 | print(data)
```

- A. (2, 1, 1)
- B. (1, 1, 1)
- C. (2, 2, 2)
- D. The code is erroneous.

Correct Answer: D

Explanation

Explanation/Reference:

The tuple `data = (1,) * 3` creates a tuple containing three elements: (1, 1, 1).
Tuples in Python are immutable, meaning their elements cannot be modified after creation.
The statement `data[0] = 2` attempts to modify the first element of the tuple, which is not allowed.
Because of the immutability of tuples, attempting to modify an element will result in a `TypeError`.
Thus, the expected output is an error message similar to:
`TypeError: 'tuple' object does not support item assignment`

QUESTION 45

What is the output of the following snippet?

```

1 | my_list_1 = [1, 2, 3]
2 | my_list_2 = []
3 | for v in my_list_1:
4 |     my_list_2.insert(0, v)
5 | print(my_list_2)

```

- A. [1, 2, 3]
- B. [3, 3, 3]
- C. [3, 2, 1]
- D. [1, 1, 1]

Correct Answer: C

Explanation

Explanation/Reference:

The code iterates over each element *v* in *my_list_1* and inserts *v* at the beginning (index 0) of *my_list2*. This results in reversing the order of the elements from *my_list_1* in *my_list2*. Here is the step-by-step process:

v = 1 -> *my_list2.insert*(0, 1) -> *my_list2* becomes [1]

v = 2 -> *my_list2.insert*(0, 2) -> *my_list2* becomes [2, 1]

v = 3 -> *my_list2.insert*(0, 3) -> *my_list2* becomes [3, 2, 1]

Therefore, the final output of *print(my_list2)* is [3, 2, 1].

QUESTION 46

What is the expected output of the following code?

```

1 | data = {'one': 'two', 'two': 'three', 'three': 'one'}
2 | res = data['three']
3 |
4 | for _ in range(len(data)):
5 |     res = data[res]
6 |
7 | print(res)

```

- A. three
- B. ('one', 'two', 'three')
- C. two
- D. one

Correct Answer: D

Explanation

Explanation/Reference:

Code breakdown:

Initialization:

data = {'one': 'two', 'two': 'three', 'three': 'one'}

res = *data*['three'] # *res* = 'one'

Loop execution:

```
for _ in range(len(data)): # len(data) = 3
res = data[res]
The loop runs 3 times (since len(data) = 3).
Iterations:
First iteration: res = data[res] = data['one'] = 'two'
Second iteration: res = data[res] = data['two'] = 'three'
Third iteration: res = data[res] = data['three'] = 'one'
```

Final value of res: After 3 iterations, res is 'one'.

Output:

```
print(res) # Outputs 'one'
```

Expected Output:

one

QUESTION 47

What code would you insert instead of the comment to obtain the expected output?

Expected output:

| | |
|---|---|
| 1 | a |
| 2 | b |
| 3 | c |

Code:

```
1 | dictionary = {}
2 | my_list = ['a', 'b', 'c', 'd']
3 |
4 | for i in range(len(my_list) - 1):
5 |     dictionary[my_list[i]] = (my_list[i], )
6 |
7 | for i in sorted(dictionary.keys()):
8 |     k = dictionary[i]
9 |     # Insert your code here.
```

- A. print(k[0])
- B. print(k)
- C. print(k['0'])
- D. print(k["0"])

Correct Answer: A

Explanation

Explanation/Reference:

```
dict = {}
list = ['a', 'b', 'c', 'd']
for i in range(len(list)-1):
dict[list[i]] = (list[i],) >>> adding list's element to the dictionary by creating a tuple containing a single list element
>>> dict = {'a': ('a',), 'b': ('b',), 'c': ('c',)}
for i in sorted(dict.keys()):
k = dict[i]
```

```
print(k[0])
>>>> a
b
c
print(k) >>> ('a',)
('b',)
('c',)
```

The other two options will generate errors

QUESTION 48

What is the output of the following snippet?

```
1 | my_list = [3, 1, -2]
2 | print(my_list[my_list[-1]])
```

- A. -2
- B. 3
- C. -1
- D. 1

Correct Answer: D

Explanation

Explanation/Reference:

```
my_list = [3, 1, -2]
print(my_list[my_list[-1]])
```

my_list[-1] retrieves the last element of the list, which is -2.

Then, my_list[my_list[-1]] becomes my_list[-2], which is the second-to-last element of the list.

my_list[-2] is 1 (the second-to-last element in the list [3, 1, -2]).

Therefore, the output of the snippet is:

- D. 1

QUESTION 49

What is the expected output of the following code?

```
1 | x = {(1, 2): 1, (2, 3): 2}
2 | print(x[1, 2])
```

- A. {(2, 3): 2}
- B. {(1, 2): 1}
- C. The code is erroneous.
- D. 1

Correct Answer: D

Explanation

Explanation/Reference:

In the given code, you have a dictionary `x` where the keys are tuples and the values are integers. The dictionary is defined as:

```
x = {(1, 2): 1, (2, 3): 2}
```

When you access the value associated with the key `(1, 2)` using `x[1, 2]`, you are essentially looking for the value stored for the key `(1, 2)` in the dictionary `x`.

The key `(1, 2)` has an associated value of `1`.
So, the expected output of the code:
`print(x[1, 2])`
is:
1

QUESTION 50

What is the expected output of the following code?

```
1 data = {'z': 23, 'x': 7, 'y': 42}
2
3 for _ in sorted(data):
4     print(data[_], end=' ')
```

- A. 7 23 42
- B. 7 42 23
- C. 42 23 7

Correct Answer: B

Explanation

Explanation/Reference:

The code provided will output the values of the dictionary data in the order of their keys sorted alphabetically. Here is a step-by-step explanation:

The dictionary data is defined as `{'z': 23, 'x': 7, 'y': 42}`.

The `sorted(data)` function sorts the keys of the dictionary alphabetically, resulting in the list `['x', 'y', 'z']`.

The for loop iterates over this sorted list of keys.

During each iteration, the corresponding value from the dictionary is accessed using the key and printed with a space as the separator.

So, the expected output of the code is:

7 42 23

QUESTION 51

What is the expected output of the following code?

```
1 data = {'a': 1, 'b': 2, 'c': 3}
2 print(data['a', 'b'])
```

- A. (1, 2)
- B. The code is erroneous.
- C. `{'a':1, 'b':2}`
- D. `[1,2]`

Correct Answer: B

Explanation

Explanation/Reference:

In the given code:

```
data = {'a': 1, 'b': 2, 'c': 3}
```

```
print(data['a', 'b'])
```

data is a dictionary with keys 'a', 'b', and 'c'.

When trying to access dictionary values, the key provided must be a single hashable object. In this case, `['a',`

'b'] is a tuple (('a', 'b')), not a valid key in the dictionary.
Since the dictionary data does not contain the key ('a', 'b'), Python raises a KeyError.

Error:

KeyError: ('a', 'b')

QUESTION 52

The fact that tuples belong to sequence types means:

- A. they can be modified using the del instruction
- B. they can be extended using the .append() method
- C. they are actually lists
- D. they can be indexed and sliced like lists

Correct Answer: D

Explanation

Explanation/Reference:

Tuples are immutable sequence types in Python, meaning they cannot be modified (no item assignment or deletion) and do not have methods like .append() for extension. However, like other sequence types such as strings and lists, tuples support indexing and slicing to access their elements.

QUESTION 53

Which function does in-place reversal of objects in a list?

- A. list.sort([func])
- B. list.pop(obj=list[-1])
- C. list.remove(obj)
- D. list.reverse()

Correct Answer: D

Explanation

Explanation/Reference:

The list.reverse() method reverses the elements of the list in place, meaning it modifies the original list directly without creating a new list. This is the function that performs an in-place reversal of objects in a list.

QUESTION 54

What is the expected output of the following code?

```
1 data1 = '1', '2'
2 data2 = ('3', '4')
3 print(data1 + data2)
```

- A. ['1', '2', '3', '4']
- B. (1, 2, 3, 4)
- C. ('1', '2', '3', '4')
- D. The code is erroneous.

Correct Answer: C

Explanation

Explanation/Reference:

data1 = '1', '2': This is a tuple assignment. In Python, writing values separated by commas without parentheses

creates a tuple. So, data1 becomes ('1', '2').

data2 = ('3', '4'): This explicitly creates a tuple with parentheses. So, data2 becomes ('3', '4').

data1 + data2: The + operator concatenates two tuples. So, ('1', '2') + ('3', '4') results in ('1', '2', '3', '4').

print(data1 + data2): This prints the concatenated tuple ('1', '2', '3', '4').

Hence, the correct answer is C.

QUESTION 55

An alternative name for a data structure called a stack is:

- A. LIFO
- B. FIFO
- C. FOLO

Correct Answer: A

Explanation

Explanation/Reference:

LIFO stands for "Last In, First Out," which describes the behavior of a stack. The last element added to the stack is the first one to be removed.

QUESTION 56

What is the expected output of the following code?

```
1 | w = [7, 3, 23, 42]
2 | x = w[1:]
3 | y = w[1:]
4 | z = w
5 | y[0] = 10
6 | z[1] = 20
7 | print(w)
```

- A. [7, 3, 23, 42]
- B. [7, 20, 23, 42]
- C. [10, 20, 42]
- D. [10, 20, 23, 42]

Correct Answer: B

Explanation

Explanation/Reference:

Let's analyze the code step by step:

w = [7, 3, 23, 42]

x = w[1:]

y = w[1:]

z = w

y[0] = 10

z[1] = 20

print(w)

w = [7, 3, 23, 42]: A list w is created with the values [7, 3, 23, 42].

`x = w[1:]`: This creates a new list `x` that is a slice of `w`, starting from index 1, i.e., `x = [3, 23, 42]`. Note that this is a shallow copy, meaning changes to `x` do not affect `w`.
`y = w[1:]`: This creates another new list `y` which is also a slice of `w`, similar to `x`. So `y = [3, 23, 42]`. Like `x`, this is also a shallow copy.
`z = w`: Here, `z` is assigned the reference to `w`, meaning any changes made to `z` will directly modify `w`.
`y[0] = 10`: This modifies the first element of `y`, so `y` becomes `[10, 23, 42]`. However, since `y` is a separate copy, `w` remains unaffected by this change.
`z[1] = 20`: This modifies the second element of `z` (and `w`, since `z` refers to `w`), so `w` becomes `[7, 20, 23, 42]`.
Finally, when `print(w)` is called, it outputs:
`[7, 20, 23, 42]`
So the correct answer is B. `[7, 20, 23, 42]`.

QUESTION 57

What is the output of the following code?

```
1 | my_list = [3, 1, -1]
2 | my_list[-1] = my_list[-2]
3 | print(my_list)
```

- A. `[1, 1, 1]`
- B. `[3, -1, 1]`
- C. `[3, 1, 1]`

Correct Answer: C
Explanation

Explanation/Reference:

Initialization of the list:

`my_list = [3, 1, -1]`

At this point, `my_list` is `[3, 1, -1]`.

Assignment operation:

`my_list[-1] = my_list[-2]`

Here:

`my_list[-1]` refers to the last element of the list (-1).

`my_list[-2]` refers to the second-to-last element of the list (1).

The operation assigns the value of `my_list[-2]` (which is 1) to `my_list[-1]`.

After this, the list becomes `[3, 1, 1]`.

Print the list:

`print(my_list)`

The output is `[3, 1, 1]`.

Final Output:

`[3, 1, 1]`

QUESTION 58

Which of the following sentences is true?

```
1 | str1 = 'Peter'
2 | str2 = str1[:]
```

- A. `str1` and `str2` are different (but equal) strings.
- B. `str1` and `str2` are different names of the same strings.
- C. `str1` is longer than `str2`
- D. `str2` is longer than `str1`

Correct Answer: B

Explanation

Explanation/Reference:

if we print the id of both str1 and str2 it will give the same id. Hence, both are referring to the same memory location, which means both are sharing the same memory.

QUESTION 59

What is the expected output of the following code?

```
1 | data = ((1, 2),) * 7
2 | print(len(data[3:8]))
```

- A. The code is erroneous.
- B. 6
- C. 5
- D. 4

Correct Answer: D

Explanation

Explanation/Reference:

Creating the data variable:

data = ((1, 2),) * 7

The tuple (1, 2) is wrapped inside another tuple, ((1, 2),).

This tuple is repeated 7 times using the * operator, creating:

data = ((1, 2), (1, 2), (1, 2), (1, 2), (1, 2), (1, 2), (1, 2))

Slicing data:

data[3:8]

Slicing starts from index 3 (inclusive) and ends at index 8 (exclusive).

However, since data only has 7 elements (index 0 to 6), the slice data[3:8] only includes elements from index 3 to the end of the tuple:

((1, 2), (1, 2), (1, 2), (1, 2))

Calculating the length of the slice:

len(data[3:8])

The slice contains 4 elements, so len(data[3:8]) returns 4.

Final Output:

QUESTION 60

What is the expected output of the following code?

```
1 | t1 = (1, 4, 9)
2 | t2 = ('A', 'D', 'Z')
3 | t3 = (True, False, None)
4 | t4 = (5.0, 7.5, 9.9)
5 |
6 | t1, t3 = t2, t4
7 | print(t1)
```

- A. The program will cause an error.
- B. (1, 4, 9)
- C. ('A', 'D', 'Z')

D. (5.0, 7.5, 9.9)

Correct Answer: C

Explanation

Explanation/Reference:

Four tuples are defined:

t1 = (1, 4, 9)

t2 = ('A', 'D', 'Z')

t3 = (True, False, None)

t4 = (5.0, 7.5, 9.9)

The next line performs tuple unpacking:

t1, t3 = t2, t4

Here, t1 is assigned the value of t2 and t3 is assigned the value of t4. After this line:

t1 now refers to ('A', 'D', 'Z')

t3 now refers to (5.0, 7.5, 9.9)

Finally, the code prints t1:

print(t1)

Since t1 has been reassigned to the value of t2, the output will be:

('A', 'D', 'Z')

QUESTION 61

How would you remove all the items from the d dictionary?

Expected output:

```
1 | {}
```

Code:

```
1 | d = {'A' : 1, 'B' : 2, 'C' : 3}
```

A. d.del()

B. d.remove()

C. del d

D. d.clear()

Correct Answer: D

Explanation

Explanation/Reference:

d.clear() removes all items from the dictionary without deleting the dictionary itself.

Other options:

d.del() and d.remove() are invalid as these methods do not exist for dictionaries.

del d deletes the dictionary entirely, resulting in d no longer being defined.

Example:

```
d = {'A': 1, 'B': 2, 'C': 3}
```

```
d.clear()
```

```
print(d) # Output: {}
```

QUESTION 62

What is the expected output of the following code?

```
1 | a = [1, 2, 3, 4, 5]
```

```
2 | print(a[3:0:-1])
```

A. [4, 3, 2, 1]

- B. [4, 3, 2]
- C. [4, 3]
- D. The code is erroneous.

Correct Answer: B

Explanation

Explanation/Reference:

Initial List

The initial list a is [1, 2, 3, 4, 5].

Slicing Operation

The slicing operation is a[3:0:-1].

The slicing syntax is list[start:stop:step]:

start is the index where the slice starts (inclusive).

stop is the index where the slice ends (exclusive).

step is the step size or the direction of the slice.

In this case:

start is 3, so the slice starts at index 3 (the element 4).

stop is 0, so the slice ends before index 0 (the element 1), and since the step is negative, it will not include the element at index 0.

step is -1, so the slice moves backward.

Extracting Elements

Following the slicing operation a[3:0:-1]:

Start at index 3 (the element 4).

Move backward with a step of -1.

Stop before reaching index 0.

The elements included in the slice are:

a[3] is 4

a[2] is 3

a[1] is 2

Therefore, the resulting sliced list is [4, 3, 2].

Output

[4, 3, 2]

QUESTION 63

What is the expected output of the following code?

```

1 | box = {}
2 | jars = {}
3 | crates = {}
4 |
5 | box['biscuit'] = 1
6 | box['cake'] = 3
7 |
8 | jars['jam'] = 4
9 |
10 | crates['box'] = box
11 | crates['jars'] = jars
12 |
13 | print(len(crates[box]))

```

- A. 2
- B. 1
- C. The code is erroneous.
- D. 3
- E. 4

Correct Answer: C

Explanation

Explanation/Reference:

The code is erroneous.

The error occurs in the line `print(len(crates[box]))`. The key `box` is not a string but a dictionary, which is not a valid key type for a dictionary in Python. Dictionary keys must be of an immutable type, such as strings, numbers, or tuples containing only immutable types.

To fix the code, you should use the string `'box'` as the key instead of the dictionary `box`.

QUESTION 64

What snippet would you insert in the line indicated below to print The highest number is 10 and the lowest number is 1. to the monitor?

```

1 | data = [10, 2, 1, 7, 5, 6, 4, 3, 9, 8]
2 | # insert your code here
3 | print(
4 |     ('The highest number is {} ' +
5 |     'and the lowest number is {}.'.format(high, low)
6 | )

```

A. None of the above.

B.

```
1 | def find_high_low(nums):
2 |     nums.sort()
3 |     return nums[0], nums[-1]
4 |
5 |
6 | high, low = find_high_low(data)
```

C.

```
1 | def find_high_low(nums):
2 |     nums.sort()
3 |     return nums[len(nums)], nums[0]
4 |
5 |
6 | high, low = find_high_low(data)
```

D.

```
1 | def find_high_low(nums):
2 |     nums.sort()
3 |     return nums[-1], nums[0]
4 |
5 |
6 | high, low = find_high_low(data)
```

Correct Answer: D

Explanation

Explanation/Reference:

```
>>> data = [10, 2, 1, 7, 5, 6, 4, 3, 9, 8]
>>>
>>> def find_high_low(nums):
...     nums.sort()
...     return nums[-1], nums[0]
...
>>> high, low = find_high_low(data)
>>>
>>> print(
... ('The highest number is {} ' +
... 'and the lowest number is {}.'.format(high, low)
... )
The highest number is 10 and the lowest number is 1.
```

QUESTION 65

What is the expected output of the following code?

```
1 | data = {'Peter': 30, 'Paul': 31}
2 | print(list(data.keys()))
```

A. ('Peter': 30, 'Paul': 31)

B. ('Peter', 'Paul')

C. ['Peter': 30, 'Paul': 31]

D. ['Peter', 'Paul']

Correct Answer: D

Explanation

Explanation/Reference:

The expected output of the code is:

['Peter', 'Paul']

`data.keys()` returns a view object that displays a list of all the keys in the dictionary `data`.

`list(data.keys())` converts this view object into a list, which contains the keys of the dictionary.

The print statement outputs this list.

QUESTION 66

Consider the following list.

`data = [1, 5, 10, 19, 55, 30, 55, 99]`

Which of the code snippets below would produce a new list like the following?

`[1, 5, 10, 99]`

A. `1 data.pop(5)`
`2 data.pop(19)`
`3 data.pop(55)`

B. `1 data.pop(1)`
`2 data.pop(3)`
`3 data.pop(4)`
`4 data.pop(6)`

C. None of the above.

D. `1 data.pop(5)`
`2 data.remove(19)`
`3 data.remove(55)`
`4 data.remove(55)`

E. `1 data.remove(5)`
`2 data.remove(19)`
`3 data.remove(55)`

Correct Answer: D

Explanation

Explanation/Reference:

`data.pop(5)` removes the element at index 5 (which is 30), leaving `[1, 5, 10, 19, 55, 55, 99]`.

`data.remove(19)` removes the first occurrence of 19, leaving `[1, 5, 10, 55, 55, 99]`.

`data.remove(55)` removes the first occurrence of 55, leaving `[1, 5, 10, 55, 99]`.

`data.remove(55)` removes the second occurrence of 55, leaving `[1, 5, 10, 99]`, which matches the target list.

QUESTION 67

What will be the output of the following code snippet?

```

1 | d = {}
2 | d[1] = 1
3 | d['1'] = 2
4 | d[1] += 1
5 |
6 | sum = 0
7 |
8 | for k in d:
9 |     sum += d[k]
10 |
11 | print(sum)

```

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

Correct Answer: C

Explanation

Explanation/Reference:

Here's the breakdown of the code and its execution:

Code Breakdown:

`d = {}`

Initializes an empty dictionary `d`.

`d[1] = 1`

Adds a key-value pair to the dictionary: key 1 (an integer) with value 1.

`d['1'] = 2`

Adds another key-value pair to the dictionary: key '1' (a string) with value 2.

Note: 1 (integer) and '1' (string) are considered different keys in Python.

`d[1] += 1`

Increments the value associated with the key 1 (integer) by 1.

The value of `d[1]` becomes 2.

Current State of `d`:

The dictionary now contains:

`d = {1: 2, '1': 2}`

`sum = 0`

Initializes a variable `sum` to 0.

`for k in d: sum += d[k]`

Iterates through the dictionary's keys (1 and '1') and adds their corresponding values to `sum`:

For `k = 1`, `sum += d[1]` ? `sum = 0 + 2 = 2`

For `k = '1'`, `sum += d['1']` ? `sum = 2 + 2 = 4`

Final Output:

The value of `sum` is 4.

Output: 4

QUESTION 68

Which one of the lines should you put in the snippet below to match the expected output?

Expected output:

```
1 | [4, 1, 7, 2, 'A']
```

Code:

```
1 | list = ['A', 2, 7, 1, 4]
2 |
3 | # enter code here
4 |
5 | print(list)
```

- A. reverse(list)
- B. list.reversed()
- C. list.reverse()
- D. reversed(list)

Correct Answer: C

Explanation

Explanation/Reference:

A. reverse(list): This is incorrect because reverse() is not a standalone function; it is a method of the list object.

B. list.reversed(): This is incorrect because reversed() is not a method of the list object. Instead, reversed(list) returns an iterator, not a modified list.

C. list.reverse(): This is correct. The reverse() method reverses the list in place, modifying the original list.

D. reversed(list): This is incorrect because reversed(list) returns an iterator, not a modified list. To get a reversed list, you would need to convert the iterator to a list, e.g., list(reversed(list)).

QUESTION 69

What is the expected output of the following code?

```
1 | data = {}
2 | data['2'] = [1, 2]
3 | data['1'] = [3, 4]
4 |
5 | for i in data.keys():
6 |     print(data[i][1], end=' ')
```

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

Correct Answer: A

Explanation

Explanation/Reference:

The dictionary data is created with two keys: '2' and '1'.

```
data['2'] = [1, 2]
```

```
data['1'] = [3, 4]
```

The for loop iterates over the keys of the dictionary data.

For each key i, the code prints the second element (index 1) of the corresponding list.

For the key '2', data['2'][1] is 2.

For the key '1', data['1'][1] is 4.

The end="" argument in the print function ensures that the outputs are printed on the same line without any spaces in between.

Therefore, the output is 24.

QUESTION 70

What is the expected output of the following code?

```
1 | data = [[0, 1, 2, 3] for i in range(2)]
2 | print(data[2][0])
```

A. The code is erroneous.

B. 2

C. 1

D. 0

Correct Answer: A

Explanation**Explanation/Reference:**

The code provided will result in an IndexError. Let's break it down step by step to understand why:

```
data = [[0, 1, 2, 3] for i in range(2)]
print(data[2][0])
```

The list comprehension `[[0, 1, 2, 3] for i in range(2)]` generates a list of lists. The `range(2)` means that the outer list will have 2 elements, each of which is the list `[0, 1, 2, 3]`.

So, data will be:

```
data = [
[0, 1, 2, 3],
[0, 1, 2, 3]
]
```

The `print(data[2][0])` statement attempts to access the element at index 2 of the data list. However, since data only has indices 0 and 1 (it has only 2 sublists), trying to access `data[2]` will raise an IndexError.

So, the expected output when you run the code will be:

IndexError: list index out of range

QUESTION 71

How many elements does the L list contain?

```
1 | L = [i for i in range(-1, -2)]
```

- A. one
- B. two
- C. three
- D. zero

Correct Answer: D

Explanation

Explanation/Reference:

The given list comprehension is:

`L = [i for i in range(-1, -2)]`

The range(start, stop) function generates numbers starting from start and stops before stop.

In this case, range(-1, -2) starts at -1 but stops before -2, and since -1 > -2, the range produces no numbers.

As a result, the list comprehension produces an empty list.

When you print L, it will output [].

Correct Answer:

D. zero

QUESTION 72

What is the expected output of the following code?

```
1 | data = {'1': '0', '0': '1'}
2 |
3 | for d in data.vals():
4 |     print(d, end=' ')
```

- A. 0 1
- B. 1 0
- C. The code is erroneous.
- D. 0 0

Correct Answer: C

Explanation

Explanation/Reference:

The code you provided contains an error. Specifically, the method vals() does not exist for dictionary objects in Python. The correct method to retrieve the values of a dictionary is values().

Here's the corrected version of the code:

```
data = {'1': '0', '0': '1'}
for d in data.values():
    print(d, end="")
```

QUESTION 73

Take a look at the snippet, and choose the true statements: (Choose two.)

```
1 | nums = [1, 2, 3]
2 | vals = nums
3 | del vals[1:2]
```

- A. nums is longer than vals

- B. nums and vals are of the same length
- C. vals is longer than nums
- D. nums and vals refer to the same list

Correct Answer: BD

Explanation

Explanation/Reference:

```
nums = [1, 2, 3]
```

Creates a list nums with three elements: [1, 2, 3].

```
vals = nums
```

The variable vals is assigned the same reference as nums. Now both vals and nums point to the same list object in memory.

```
del vals[1:2]
```

The slice vals[1:2] selects the element at index 1 (value 2).

The del statement removes this element from the list. Since vals and nums point to the same list, the change affects both.

After this operation, the list becomes [1, 3].

Now evaluate the options:

A. nums is longer than vals

False. Both nums and vals refer to the same list, which now has a length of 2.

B. nums and vals are of the same length

True. Both nums and vals are the same list, so their lengths are equal.

C. vals is longer than nums

False. Both nums and vals refer to the same list.

D. nums and vals refer to the same list

True. nums and vals point to the same memory location, so they refer to the same list.

Correct Answers:

B. nums and vals are of the same length

D. nums and vals refer to the same list

QUESTION 74

Take a look at the snippet and choose one of the following statements which is true:

```
1 | nums = []  
2 | vals = nums[:]  
3 | vals.append(1)
```

- A. nums and vals are of the same length
- B. nums is longer than vals
- C. vals is longer than nums

Correct Answer: C

Explanation

Explanation/Reference:

Let's analyze the given code snippet step-by-step to determine which statement is true:

```
nums = []
```

```
vals = nums[:]
```

```
vals.append(1)
```

```
nums = []
```

nums is initialized as an empty list: [].

```
vals = nums[:]
```

vals is created as a shallow copy of nums.
Since nums is an empty list, vals will also be an empty list: [].
vals.append(1)

1 is appended to the list vals.
After this operation, vals will be: [1].

Final State of the Lists
nums remains an empty list: [].
vals is now: [1].

Comparing the Lengths
The length of nums is 0.
The length of vals is 1.

Conclusion
vals is longer than nums.
Therefore, the correct statement is:
C. vals is longer than nums

QUESTION 75

What is the expected output of the following code?

```
1 | data = (1, 2, 4, 8)
2 | data = data[1:-1]
3 | data = data[0]
4 | print(data)
```

- A. (2)
- B. (2,)
- C. 2
- D. The code is erroneous.

Correct Answer: C

Explanation

Explanation/Reference:

data = (1, 2, 4, 8)

Here, data is a tuple: (1, 2, 4, 8).

data = data[1:-1]

This slices the tuple starting from index 1 and ending at index -1. So, it extracts the elements between index 1 and index -1, excluding the element at index -1. The result is: (2, 4).

data = data[0]

Now, data is a tuple (2, 4). The expression data[0] takes the first element of the tuple, which is 2.

print(data)

Finally, the code prints the value of data, which is now 2.

Therefore, the expected output is:

C. 2

QUESTION 76

What is the expected output of the following code?

```
1 | x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
2 | x[::2] = 10, 20, 30, 40, 50, 60
3 | print(x)
```

- A. [1, 10, 3, 20, 5, 30, 7, 40, 9, 50, 60]
- B. [1, 2, 10, 20, 30, 40, 50, 60]
- C. The code is erroneous.
- D. [10, 2, 20, 4, 30, 6, 40, 8, 50, 60]

Correct Answer: C

Explanation

Explanation/Reference:

The code will result in an error because you are trying to assign 6 values (10, 20, 30, 40, 50, 60) to every other element of the list x, which has only 5 positions (1, 3, 5, 7, 9) available for assignment. The lengths do not match, and thus it raises a ValueError.

QUESTION 77

What is the expected output of the following code?

```
1 data = {'name': 'Peter', 'age': 30}
2 person = data.copy()
3 print(id(data) == id(person))
```

- A. False
- B. 1
- C. 0
- D. True

Correct Answer: A

Explanation

Explanation/Reference:

data.copy() creates a shallow copy of the dictionary data. The new dictionary (person) contains the same key-value pairs but resides at a different memory location.

id(data) retrieves the memory address of the data dictionary.

id(person) retrieves the memory address of the person dictionary.

Since the two dictionaries are separate objects, id(data) is not equal to id(person), and the comparison id(data) == id(person) evaluates to False.

QUESTION 78

What is the expected output of the following code?

```
print(list('hello'))
```

- A. hello
- B. [h, e, l, l, o]
- C. ['h', 'e', 'l', 'l', 'o']
- D. ['h' 'e' 'l' 'l' 'o']
- E. None of the above.

Correct Answer: C

Explanation

Explanation/Reference:

The string 'hello' is passed to the list() function.

list() converts the string into a list, where each character of the string becomes an individual element of the list. The resulting list ["h", "e", "l", "l", "o"] is then printed.

Each character from the string 'hello' becomes a separate string element in the list, preserving the order of the original string. The output shows this list of single-character strings.

QUESTION 79

What is the expected output of the following code?

```
1 | list1 = [1, 3]
2 | list2 = list1
3 | list1[0] = 4
4 | print(list2)
```

- A. [1, 3]
- B. [1, 4]
- C. [4, 3]
- D. [1, 3, 4]

Correct Answer: C

Explanation

Explanation/Reference:

`list1 = [1, 3]`

This creates a list `list1` with elements `[1, 3]`.

`list2 = list1`

This does not create a new list. Instead, `list2` is a reference to the same list object that `list1` points to. Both `list1` and `list2` now refer to the same memory location.

`list1[0] = 4`

This modifies the first element of the list referred to by `list1` (and `list2`, since they refer to the same object). The list is now `[4, 3]`.

`print(list2)`

Since `list2` refers to the same list as `list1`, it will also reflect the change. Therefore, the output will be `[4, 3]`.

QUESTION 80

What is the expected output of the following code?

```
1 | data = ['Peter', 404, 3.03, 'Wellert', 33.3]
2 | print(data[1:3])
```

- A. ['Peter', 404, 3.03, 'Wellert', 33.3]
- B. None of the above.
- C. [404, 3.03]
- D. ['Peter', 'Wellert']

Correct Answer: C

Explanation

Explanation/Reference:

The code performs the following steps:

1. `data = ['Peter', 404, 3.03, 'Wellert', 33.3]`: Initializes the list `data` with the elements `['Peter', 404, 3.03, 'Wellert', 33.3]`.

2. `print(data[1:3])`: Slices the list `data` starting from index `1` up to, but not including, index `3`, and prints the resulting sublist.

The slice `data[1:3]` will include the elements at indices `1` and `2`, which are `404` and `3.03`.

****Expected output:**** `[404, 3.03]`

QUESTION 81

What will be the output of the following code snippet?

```
1 | a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
2 | print(a[::2])
```

- A. [1, 3, 5, 7, 9]
- B. [8, 9]
- C. [1, 2, 3]
- D. [1, 2]

Correct Answer: A

Explanation

Explanation/Reference:

`a[::2]` uses Python slicing syntax:

The first `:` indicates the starting point (default is the beginning of the list).

The second `:` indicates the stopping point (default is the end of the list).

The 2 after the second colon is the step size, which means it skips every second element.

Starting from the first element (1), it selects every second element: 1, 3, 5, 7, 9.

QUESTION 82

What is the expected output of the following code?

```
1 | data = [
2 |     [1, 2, 3, 4],
3 |     [5, 6, 7, 8],
4 |     [9, 10, 11, 12],
5 |     [13, 14, 15, 16]
6 | ]
7 | for i in range(0, 4):
8 |     print(data[i].pop(), end=' ')
```

- A. 1 5 9 13
- B. 1 2 3 4
- C. 13 14 15 16
- D. 4 8 12 16

Correct Answer: D

Explanation

Explanation/Reference:

The provided code iterates over each sublist in the data list and removes (and prints) the last element from each sublist using the `pop()` method. The `pop()` method removes and returns the last item in the list if no index is specified.

the expected output of the code is:

4 8 12 16

QUESTION 83

What is the expected output of the following code?

```
1 | x = [0, 1, 2]
2 | x.insert(0, 1)
3 | del x[1]
4 | print(sum(x))
```

- A. 2
- B. 4
- C. 5
- D. 3

Correct Answer: B

Explanation

Explanation/Reference:

`x = [0, 1, 2]` # Step 1: x is initialized as [0, 1, 2].

`x.insert(0, 1)` # Step 2: Inserts 1 at index 0. x becomes [1, 0, 1, 2].

`del x[1]` # Step 3: Deletes the element at index 1. x becomes [1, 1, 2].

`print(sum(x))` # Step 4: Calculates the sum of elements in x, which is $1 + 1 + 2 = 4$.

Result:

The output of the code is 4.

QUESTION 84

You develop a Python application for your company.

A list named `employees` contains 200 employee names, the last five being company management. You need to slice the list to display all employees excluding management.

Which code segments can you use? (Choose two.)

- A. `employees[1:-5]`
- B. `employees[0:-5]`
- C. `employees[:-5]`
- D. `employees[0:4]`
- E. `employees[1:4]`

Correct Answer: BC

Explanation

Explanation/Reference:

To slice the list to display all employees excluding the last five who are company management, you can use the following code segments:

B. `employees[0:-5]`

C. `employees[:-5]`

Both of these options will correctly exclude the last five elements from the list.

QUESTION 85

What is the expected output of the following code?

```

1 data = {1: 0, 2: 1, 3: 2, 0: 1}
2 x = 0
3
4 for _ in range(len(data)):
5     x = data[x]
6
7 print(x)

```

- A. 0
- B. 1
- C. The code is erroneous.
- D. 2

Correct Answer: A

Explanation

Explanation/Reference:

et's go through the code step by step:

The dictionary data is defined as {1: 0, 2: 1, 3: 2, 0: 1}.

The variable x is initialized to 0.

The for loop runs len(data) times, which is 4 times since the length of the dictionary data is 4.

Let's trace the value of x through each iteration of the loop:

Iteration 1: x = data[0] which is 1. So x becomes 1.

Iteration 2: x = data[1] which is 0. So x becomes 0.

Iteration 3: x = data[0] which is 1. So x becomes 1.

Iteration 4: x = data[1] which is 0. So x becomes 0.

After completing the loop, the value of x is 0. Therefore, the output is: 0

QUESTION 86

After execution of the following snippet, the sum of all vals elements will be equal to:

```

1 vals = [0, 1, 2]
2 vals.insert(0, 1)
3 del vals[1]

```

- A. 4
- B. 5
- C. 2
- D. 3

Correct Answer: A

Explanation

Explanation/Reference:

et's analyze the given code snippet step-by-step to determine the sum of all elements in the vals list after execution:

```
vals = [0, 1, 2]
```

Initially, vals is set to [0, 1, 2].
vals.insert(0, 1)

This inserts the element 1 at index 0. After this operation, vals becomes [1, 0, 1, 2].
del vals[1]

This deletes the element at index 1. After this operation, vals becomes [1, 1, 2].
Now, let's calculate the sum of all elements in the vals list:

$1 + 1 + 2 = 4$

Therefore, the sum of all elements in the vals list after execution is 4.

QUESTION 87

What is the output of the following code?

```
1 | a = 1
2 | b = 0
3 | x = a or b
4 | y = not(a and b)
5 | print(x + y)
```

- A. The output cannot be predicted.
- B. 2
- C. The program will cause an error.
- D. 1

Correct Answer: B

Explanation

Explanation/Reference:

1. 'a' is assigned the value '1'.
2. 'b' is assigned the value '0'.

Next, we evaluate the expression 'x = a or b':

- The 'or' operator returns the first truthy value it encounters, or the last value if none are truthy.
- 'a' is '1', which is truthy, so 'x' will be '1'.

Then, we evaluate the expression 'y = not(a and b)':

- The 'and' operator returns the first falsy value it encounters, or the last value if none are falsy.
- 'a' is '1' (truthy) and 'b' is '0' (falsy), so 'a and b' is '0'.
- Applying the 'not' operator to '0' (falsy) results in 'True', which is equivalent to '1' in Python.

Finally, we compute 'x + y':

- 'x' is '1' and 'y' is '1'.
- So, 'x + y' is '1 + 1', which equals '2'.

Therefore, the output of the code is: 2

QUESTION 88

What is the expected output of the following code?

```
1 | data1 = (1, 2)
2 | data2 = (3, 4)
3 | [print(sum(x)) for x in [data1 + data2]]
```

- A. 4
- B. 10
- C. Nothing gets printed.
- D. The code is erroneous.

Correct Answer: B

Explanation

Explanation/Reference:

The given code concatenates two tuples data1 and data2 to form a new tuple and then sums up its elements. Here is the step-by-step

data1 = (1, 2)

data2 = (3, 4)

The expression data1 + data2 concatenates the two tuples to form a single tuple (1, 2, 3, 4).

The list comprehension [print(sum(x)) for x in [data1 + data2]] iterates over a list containing the single concatenated tuple [(1, 2, 3, 4)].

So, the list comprehension will execute print(sum((1, 2, 3, 4))).

The sum function calculates the sum of the elements in the tuple (1, 2, 3, 4), which is $1 + 2 + 3 + 4 = 10$.

The print function prints the result 10.

Therefore, the expected output of the code is: 10

QUESTION 89

What is the output of the following code?

```
1 | a = 10
2 | b = 20
3 | c = a > b
4 | print(not(c))
```

- A. The program will cause an error.
- B. False
- C. None
- D. True

Correct Answer: D

Explanation

Explanation/Reference:

Let's analyze the code step by step:

1. 'a = 10': This assigns the value 10 to the variable 'a'.

2. 'b = 20': This assigns the value 20 to the variable 'b'.

3. 'c = a > b': This compares whether 'a' is greater than 'b'. Since '10' is not greater than '20', the result is 'False'. Therefore, 'c' is assigned the value 'False'.

4. 'print(not(c))': This prints the logical negation of 'c'. Since 'c' is 'False', 'not(c)' is 'True'.

QUESTION 90

What is the expected output of the following code?

```
x = 28
y = 8
print(x / y)
print(x // y)
print(x % y)
```

- A. 1 | 3.0
2 | 3
3 | 2
- B. 1 | 3.5
2 | 3
3 | 4
- C. 1 | 3
2 | 3.5
3 | 4
- D. 1 | 3.5
2 | 3.5
3 | 2

Correct Answer: B

Explanation

Explanation/Reference:

he expected output of the code is:

```
x = 28
y = 8
print(x / y) # This performs floating-point division
print(x // y) # This performs integer (floor) division
print(x % y) # This calculates the remainder of the division
```

Let's calculate each step:

1. ``x / y`` performs floating-point division:
`\(28 / 8 = 3.5 \)`
2. ``x // y`` performs integer (floor) division:
`\(28 // 8 = 3 \)`
3. ``x % y`` calculates the remainder of the division:
`\(28 \% 8 = 4 \)`

So, the output will be:

3.5

QUESTION 91

What would you insert instead of ???, so that the program checks for even numbers?

```
1 | if ???:  
2 |     print('x is an even number')
```

- A. `x % 'even' == True`
- B. `x % 2 == 1`

- C. $x \% 2 == 0$
- D. $x \% x == 0$
- E. $x \% 1 == 2$

Correct Answer: C

Explanation

Explanation/Reference:

QUESTION 92

A data structure described as LIFO is actually a:

- A. stack
- B. tree
- C. list
- D. heap

Correct Answer: A

Explanation

Explanation/Reference:

LIFO stands for "Last In, First Out," which is a characteristic of a stack data structure. In a stack, the last element added is the first one to be removed.

QUESTION 93

What is the expected output of the following code?

```
x = 1 // 5 + 1 / 5
print(x)
```

- A. 0
- B. 0.2
- C. 0.0
- D. 0.4

Correct Answer: B

Explanation

Explanation/Reference:

First, $1 // 5$ is calculated. The `//` operator performs integer (floor) division.

$1 // 5$ results in 0 because 1 divided by 5 is 0 with a remainder of 1, and the floor division discards the remainder.

Next, $1 / 5$ is calculated. The `/` operator performs true division (floating-point division).

$1 / 5$ results in 0.2 because 1 divided by 5 is 0.2.

The results of these two operations are then added together:

$0 \text{ (result of } 1 // 5) + 0.2 \text{ (result of } 1 / 5) = 0.2$.

So, the value of `x` is 0.2.

QUESTION 94

What is the expected output of the following code?

```
print(1 // 2 * 3)
```

- A. 4.5

- B. 0.16666666666666666
- C. 0.0
- D. 0

Correct Answer: D

Explanation

Explanation/Reference:

Actually, in Python, both multiplication (*) and floor division (//) have the same precedence level, and they are evaluated from left to right.

So, in the given code `print(1 // 2 * 3)`:

`1 // 2` is evaluated first, resulting in 0 because 1 divided by 2 is 0.5 and the floor division of 0.5 is 0.

Then, `0 * 3` is evaluated, which is 0.

Therefore, the expected output is indeed 0.

QUESTION 95

What is the expected output of the following code?

```
1 | data = ()
2 | print(data.__len__())
```

- A. The code is erroneous.
- B. 1
- C. 0
- D. None

Correct Answer: C

Explanation

Explanation/Reference:

The given Python code snippet defines an empty tuple and prints its length using the `__len__()` method.
Code:

```
data = ()
print(data.__len__())
```

`data = ()` initializes an empty tuple.

`data.__len__()` is a method that returns the number of items in the tuple.

Since the tuple is empty, the length is 0.

QUESTION 96

How many elements does the `my_list` list contain?

```
1 | my_list = [0 for i in range(1, 3)]
```

- A. one
- B. three
- C. two

Correct Answer: C

Explanation

Explanation/Reference:

The list `my_list` is created using a list comprehension. The expression `[0 for i in range(1, 3)]` generates a list containing the value 0 for each number in the range from 1 to 2 (inclusive of 1 and exclusive of 3).

So, the range `range(1, 3)` produces two numbers: 1 and 2.

Therefore, the list `my_list` will contain 2 elements, both of which are 0.

Hence, `my_list` contains 2 elements.

QUESTION 97

The result of the following addition:

`123 + 0.0`

- A. cannot be evaluated
- B. is equal to 123
- C. is equal to 123.0

Correct Answer: C

Explanation

Explanation/Reference:

The result of the addition `123 + 0.0` is equal to 123.0.

QUESTION 98

What is the output of the following snippet?

```
1 | dictionary = {'one': 'two', 'three': 'one', 'two': 'three'}
2 | v = dictionary['one']
3 |
4 | for k in range(len(dictionary)):
5 |     v = dictionary[v]
6 |
7 | print(v)
```

- A. two
- B. one
- C. ('one', 'two', 'three')
- D. three

Correct Answer: A

Explanation

Explanation/Reference:

Initialization:

`dictionary` is a dictionary: `{'one': 'two', 'three': 'one', 'two': 'three'}`.

`v` is initialized to `dictionary['one']`, which is 'two'.

For Loop Iterations:

First iteration (`k = 0`):

Current value of `v` is 'two'.

`v = dictionary[v]` ? 'two' maps to 'three'.

So, `v = 'three'`.

Second iteration (k = 1):
Current value of v is 'three'.
v = dictionary['three'] ? 'three' maps to 'one'.
So, v = 'one'.

Third iteration (k = 2):
Current value of v is 'one'.
v = dictionary['one'] ? 'one' maps to 'two'.
So, v = 'two'.

Final Value of v:
After 3 iterations, the value of v is 'two'.

QUESTION 99

What is the output of the following snippet?

```
1 | my_list = [0, 1, 2, 3]
2 | x = 1
3 | for elem in my_list:
4 |     x *= elem
5 | print(x)
```

- A. 1
- B. 0
- C. 6

Correct Answer: B

Explanation

Explanation/Reference:

The given Python snippet multiplies the value of x by each element in my_list. Let's break down the execution step by step:

my_list = [0, 1, 2, 3]
This initializes the list my_list with the elements [0, 1, 2, 3].

x = 1
This initializes x with the value 1.

for elem in my_list:
This starts a loop where elem will take the value of each element in my_list.

x *= elem
Inside the loop, x is multiplied by the current element (elem).

Here's the loop breakdown:

In the first iteration, elem = 0. So, x = 1 * 0 = 0.
In the second iteration, elem = 1. So, x = 0 * 1 = 0.
In the third iteration, elem = 2. So, x = 0 * 2 = 0.
In the fourth iteration, elem = 3. So, x = 0 * 3 = 0.

After the loop finishes, x remains 0.

print(x)

This will print 0.

QUESTION 100

What is the expected output of the following code?

```
x = 1 / 2 + 3 // 3 + 4 ** 2  
print(x)
```

- A. 17
- B. 17.5
- C. 8.5
- D. 8

Correct Answer: B

Explanation

Explanation/Reference:

1 / 2 evaluates to 0.5 (division)

3 // 3 evaluates to 1 (integer division)

4 ** 2 evaluates to 16 (exponentiation)

Now we add these results together:

$$0.5 + 1 + 16 = 17.5$$

So, the correct output of the code is 17.5.

QUESTION 101

What will be the output of the following code snippet?

```
print(3 / 5)
```

- A. None of the above.
- B. 0
- C. 0.6
- D. 6/10

Correct Answer: C

Explanation

Explanation/Reference:

The output of the code snippet `print(3 / 5)` will be: 0.6

QUESTION 102

What is the output of the following snippet?

```
1 | dct = {}  
2 | dct['1'] = (1, 2)  
3 | dct['2'] = (2, 1)  
4 |  
5 | for x in dct.keys():  
6 |     print(dct[x][1], end='')
```

- A. 12
- B. (2, 1)

- C. (1, 2)
- D. 21

Correct Answer: D

Explanation

Explanation/Reference:

To determine the output of this code, let's go through it step by step:

`dct = {}` creates an empty dictionary.

`dct['1'] = (1, 2)` adds a key-value pair to the dictionary. The key is the string '1', and the value is the tuple (1, 2).

`dct['2'] = (2, 1)` adds another key-value pair. The key is the string '2', and the value is the tuple (2, 1).

The for loop iterates over the keys of the dictionary using `dct.keys()`.

For each key `x`, it prints `dct[x][1]` with `end=""`. This means it's printing the second element of each tuple (index 1), and the `end=""` argument prevents a newline after each print.

So, when we iterate over the keys:

For key '1', it will print 2 (the second element of (1, 2))

For key '2', it will print 1 (the second element of (2, 1))

These will be printed on the same line because of `end=""`.

Therefore, the output will be: 21

QUESTION 103

Only one of the following statements is true - which one?

- A. addition precedes multiplication
- B. multiplication precedes addition
- C. neither statement can be evaluated

Correct Answer: B

Explanation

Explanation/Reference:

In the order of operations (PEMDAS/BODMAS), multiplication is performed before addition.

QUESTION 104

Analyze the following code fragments that assign a boolean value to the variable `even`?

```

1 | num = 42
2 |
3 | # Code-1
4 | if num % 2 == 0:
5 |     even = True
6 | else:
7 |     even = False
8 |
9 | # Code-2
10 | even = True if num % 2 == 0 else False
11 |
12 | # Code-3
13 | even = num % 2 == 0

```

- A. All three are correct, but Code-2 is preferred.
- B. All three are correct, but Code-3 is preferred.
- C. Code-3 has a syntax error because you attempt to assign a number to even.
- D. All three are correct, but Code-1 is preferred.
- E. Code-2 has a syntax error because you cannot have True and False literals in the conditional expression.

Correct Answer: D

Explanation

Explanation/Reference:

Code-1 is more explicit and can be clearer for beginners or those unfamiliar with ternary operators.

QUESTION 105

What will be the output of the following code snippet?

```

1 | x = 2
2 | y = 1
3 | x *= y + 1
4 | print(x)

```

- A. 1
- B. 4
- C. 2
- D. 3
- E. None

Correct Answer: B

Explanation

Explanation/Reference:

Let's break down the code snippet step by step:

`x = 2` assigns the value 2 to the variable `x`.

`y = 1` assigns the value 1 to the variable `y`.

`x *= y + 1` is equivalent to `x = x * (y + 1)`.

First, the expression inside the parentheses is evaluated: `y + 1` which is `1 + 1` resulting in 2.

Then, the multiplication is performed: `x * 2` which is `2 * 2` resulting in 4.

`print(x)` outputs the value of `x`, which is now 4.

QUESTION 106

What is the expected output of the following code?

```
x = 1 + 1 // 2 + 1 / 2 + 2
```

```
print(x)
```

- A. 3
- B. 4.0
- C. 4
- D. 3.5

Correct Answer: D

Explanation**Explanation/Reference:**

Let's break down the expression step by step, considering floor division in Python:

```
x = 1 + 1 // 2 + 1 / 2 + 2
```

1. The `//` operator (floor division) and `/` operator (float division) have higher precedence than the `+` operator, so they will be evaluated first.

2. Evaluate `1 // 2`:

- `1 // 2` results in `0` because floor division of 1 by 2 rounds down to 0.

3. Evaluate `1 / 2`:

- `1 / 2` results in `0.5` because it is float division.

Now substitute these results back into the expression:

```
x = 1 + 0 + 0.5 + 2
```

4. Next, evaluate the additions from left to right:

- `1 + 0` results in `1`

- `1 + 0.5` results in `1.5`

- `1.5 + 2` results in `3.5`

Therefore, the expected output of the code is: 3.5

QUESTION 107

Which of the following statements are true? (Choose two.)

- A. The result of the `/` operator is always an integer value.
- B. The `**` operator uses right-sided binding.
- C. The right argument of the `%` operator cannot be zero.
- D. Addition precedes multiplication.

Correct Answer: BC

Explanation**Explanation/Reference:**

The two statements that are true are:

B. The `**` operator uses right-sided binding.

C. The right argument of the `%` operator cannot be zero.

Statement A is false because the `/` operator can result in a floating-point value if the operands are not both integers. Statement D is false because, according to the order of operations (PEMDAS/BODMAS rules), multiplication precedes addition.

QUESTION 108

You develop a Python application for your company. You have the following code.

```
1 | def main(a, b, c, d):  
2 |     value = a + b * c - d  
3 |     return value
```

Which of the following expressions is equivalent to the expression in the function?

- A. $(a + b) * (c - d)$
- B. $(a + (b * c)) - d$
- C. $(a + ((b * c) - d))$
- D. None of the above.

Correct Answer: D

Explanation

Explanation/Reference:

A does not match because it changes the order of operations by prioritizing addition/subtraction. B will produce a syntax error, as it includes 2 open parentheses, but 3 closing parentheses. C will produce a syntax error, as it includes 3 open parentheses, but 2 closing parentheses. This leaves D as the correct answer.

QUESTION 109

Assuming that the following assignment has been successfully executed:

```
the_list = ['list', False, 3e8]
```

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

- A. `int(the_list[2]) == len(the_list)`
- B. `the_list[1] in the_list`
- C. `300 in the_list and the_list[1]`
- D. `the_list.index(False) == 1`

Correct Answer: BD

Explanation

Explanation/Reference:

QUESTION 110

Consider the following code.

```
1 | x = 1  
2 | x = x == x
```

The value eventually assigned to x is equal to:

- A. 1
- B. 0
- C. True
- D. False

Correct Answer: C

Explanation

Explanation/Reference:

In the given code:

```
x = 1
x = x == x
```

Here is the step-by-step explanation of what happens:

1. `x` is initially assigned the value `1`.
2. The expression `x == x` is evaluated. Since `x` is `1`, the expression `1 == 1` evaluates to `True`.
3. The result of the expression (`True`) is then assigned back to `x`.

Therefore, the value eventually assigned to `x` is `True`.

So, the correct answer is: True

QUESTION 111

The += operator, when applied to strings, performs:

- A. Subtraction
- B. Concatenation
- C. Multiplication

Correct Answer: B

Explanation

Explanation/Reference:

The += operator, when applied to strings, appends the right-hand string to the left-hand string, effectively concatenating them.

QUESTION 112

How much will the delivery cost be, if the order value is 1700 and the state is FL (Florida)?

```

1 | order = int(input('Please enter the order value: '))
2 | state = input('Please enter the state (as postal abbreviation): ')
3 | delivery = 0
4 |
5 | if state in ['NC', 'SC', 'VA']:
6 |     if order <= 1000:
7 |         delivery = 70
8 |     elif 1000 < order < 2000:
9 |         delivery = 80
10 |    else:
11 |        delivery = 90
12 | else:
13 |     delivery = 50
14 | if state in ['GA', 'WV', 'FL']:
15 |     if order > 1000:
16 |         delivery += 30
17 |     if order < 2000 and state in ['WV', 'FL']:
18 |         delivery += 40
19 |     else:
20 |         delivery += 25
21 | print(delivery)

```

- A. 120
- B. 105
- C. 80
- D. 90

Correct Answer: D

Explanation

Explanation/Reference:

```

order = int(input('Please enter the order value: '))
state = input('Please enter the state (as postal abbreviation): ')
delivery = 0

```

```

if state in ['NC', 'SC', 'VA']:
if order <= 1000:
delivery = 70
elif 1000 < order < 2000:
delivery = 80
else:

```

```

delivery = 90

else:
    delivery = 50
if state in ['GA', 'WV', 'FL']:
    if order > 1000:
        delivery + 30
    if order < 2000 and state in ['WV', 'FL']:
        delivery += 40
    else:
        delivery + 25

print (delivery)

```

QUESTION 113

Consider the following code.

```

1 | data = ['Peter', 'Paul', 'Mary', 'Jane']
2 | res = 0

```

Which of the following code snippets will expand the code, so that 100 will be printed to the monitor? (Choose two.)

- A.
- ```

1 | for i in ('Peter', 'Steve', 'Jane'):
2 | if i in data:
3 | res += 100
4 | print(res)

```
- B.
- ```

1 | for i in ('Peter', 'Steve', 'Jane'):
2 |     if i in data:
3 |         res += 50
4 | print(res)

```
- C.
- ```

1 | for i in ('Peter', 'Steve', 'Jane'):
2 | if i not in data:
3 | res += 50
4 | print(res)

```
- D.
- ```

1 | for i in ('Peter', 'Steve', 'Jane'):
2 |     if i not in data:
3 |         res += 100
4 | print(res)

```

Correct Answer: BD

Explanation

Explanation/Reference:

QUESTION 114

What will happen when you attempt to run the following code?

```

1 | While True:
2 |     print("1")

```

- A. The program will not run due to syntax error.
- B. The program will run and cause an error.
- C. The program will fall into infinite loop, printing 1 on each line.
- D. The program will print 1 on one line only.

Correct Answer: A

Explanation

Explanation/Reference:

QUESTION 115

The ** operator:

- A. performs duplicated multiplication
- B. does not exist
- C. performs floating-point multiplication
- D. performs exponentiation

Correct Answer: D

Explanation

Explanation/Reference:

In Python, the ** operator is used to raise a number to the power of another number. For example, 2 ** 3 results in 8.

QUESTION 116

What is the expected output of the following code?

```

1 | def func(num):
2 |     res = '*'
3 |     for _ in range(num):
4 |         res += res
5 |     return res
6 |
7 |
8 | for x in func(2):
9 |     print(x, end='')

```

- A. ****
- B. The code is erroneous.
- C. **
- D. *

Correct Answer: A
Explanation

Explanation/Reference:

unc(2) means num = 2, so the for loop will execute twice.

First iteration (_ = 0):

res starts as '*'.

res += res doubles res, resulting in res = '**'.

Second iteration (_ = 1):

res is currently '**'.

res += res doubles res again, resulting in res = '****'.

After the loop completes, func(2) returns the string '****'.

QUESTION 117

Insert the correct snippet so that the program produces the expected output.

Expected output:

```
1 | True
```

Code:

```
1 | list = [False, True, "2", 3, 4, 5]
2 | # insert code here
3 | print(b)
```

- A. b = 0 not in list
- B. b = list[0]
- C. b = 0 in list
- D. b = False

Correct Answer: C
Explanation

Explanation/Reference:

QUESTION 118

Assuming that the tuple is a correctly created tuple, the fact that tuples are immutable means that the following instruction:

```
1 | my_tuple[1] = my_tuple[1] + my_tuple[0]
```

- A. is illegal
- B. may be illegal if the tuple contains strings
- C. can be executed if and only if the tuple contains at least two elements
- D. is fully correct

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 119

What is the output of the following snippet?

```
1 | my_list = [1, 2]
2 |
3 | for v in range(2):
4 |     my_list.insert(-1, my_list[v])
5 |
6 | print(my_list)
```

- A. [1, 1, 2, 2]
- B. [1, 1, 1, 2]
- C. [1, 2, 1, 2]
- D. [1, 2, 2, 2]

Correct Answer: B

Explanation

Explanation/Reference:

In Python, when using the `insert()` method with lists, the index `-1` behaves differently from how negative indexing works for accessing elements.

Negative index: With `insert(-1, value)`, Python interprets `-1` as "insert before the element at index `-1`". Since `-1` refers to the last element for access, inserting before this position means the new element goes before the current last element, not at the very end.

This might seem counterintuitive at first, but it's consistent with how Python treats indexing for modification versus access:

Access: `-1` gets you the last element.

Insertion: `-1` means insert just before where you would access with `-1`.

QUESTION 120

What is the expected output of the following code?

```
1 | data = (1, 2, 4, 8)
2 | data = data[-2:-1]
3 | data = data[-1]
4 | print(data)
```

- A. (4)
- B. 4
- C. (4,)
- D. 44

Correct Answer: B

Explanation

Explanation/Reference:

Start: data = (1, 2, 4, 8) (tuple with 4 elements).

Slicing: data = data[-2:-1] → data = (4,) (tuple with 1 element, getting the penultimate element 4).

Direct access: data = data[-1] → data = 4 (integer, getting the last element of the tuple (4,)).

Output: print(data) → 4.

QUESTION 121

What would you insert instead of ???, so that the program prints True to the monitor?

```
1 | x = 'Peter'
2 | y = 'Peter'
3 | res = ???
4 | print(res)
```

- A. x < y
- B. x is not y
- C. x is y
- D. x != y

Correct Answer: C

Explanation

Explanation/Reference:

QUESTION 122

What value will be assigned to the x variable?

```
1 | z = 10
2 | y = 0
3 | x = z > y or z == y
```

- A. 1
- B. False
- C. True

Correct Answer: C

Explanation

Explanation/Reference:

QUESTION 123

Consider the following Python code:

```
1 | distance = 1876.23
2 | amount = +42E7
3 | country = 'Italy'
```

What are the types of the variables distance, amount and country?

- A. float, int, str
- B. float, str, str
- C. double, str, float
- D. float, float, str

Correct Answer: D

Explanation

Explanation/Reference:

1876.23 is a float because it is a number w/o quotations and has a decimal +42E7 is scientific notation for $42 \times 10^{**7}$, which is a float, and 'Italy' is a string because it has quotation marks.

QUESTION 124

Consider the following code snippet:

```
1 | w = bool(23)
2 | x = bool('')
3 | y = bool(' ')
4 | z = bool([False])
```

Which of the variables will contain False?

- A. z
- B. w
- C. x
- D. y

Correct Answer: C

Explanation

Explanation/Reference:

QUESTION 125

You are an intern for ABC electric cars company. You must create a function that calculates the average velocity of their vehicles on a 1320 foot (1/4 mile) track. Consider the following code.

```
1 | distance = ???(input('Enter the distance travelled in feet'))
2 | distance_miles = distance/5280 # convert to miles
3 |
4 | time = ???(input('Enter the time elapsed in seconds'))
5 | time_hours = time/3600 # convert to hours
6 |
7 | velocity = distance_miles/time_hours
8 | print('The average Velocity : ', velocity, 'miles/hour')
```

The output must be as precise as possible. What would you insert instead of ??? and ??? ?

- A. 1 | int
2 | int

- B. 1 | float
2 | float
- C. 1 | int
2 | float
- D. 1 | float
2 | int

Correct Answer: B
Explanation

Explanation/Reference:

"The output must be as precise as possible" tells you that you need a float to provide a decimal to achieve precision.

QUESTION 126

What is the expected output of the following code if the user enters 3 and 2?

```
1 | x = int(input())
2 | y = int(input())
3 | x = x % y
4 | x = x % y
5 | y = y % x
6 | print(y)
```

- A. 1
- B. 2
- C. 3
- D. 0

Correct Answer: D
Explanation

Explanation/Reference:

```
x = int(input()) # x = 3
y = int(input()) # y = 2
```

```
x = x % y # x = 3 % 2 = 1
x = x % y # x = 1 % 2 = 1
y = y % x # y = 2 % 1 = 0
print(y) # Output: 0
```

QUESTION 127

What is the expected output of the following code?

```
1 | print(not 0)
2 | print(not 23)
3 | print(not '')
4 | print(not 'Peter')
5 | print(not None)
```

- A. 1 | False
2 | False
3 | True
4 | False
5 | True
- B. 1 | True
2 | False
3 | False
4 | False
5 | True
- C. 1 | True
2 | False
3 | True
4 | False
5 | False
- D. 1 | True
2 | False
3 | True
4 | False
5 | True

Correct Answer: D

Explanation

Explanation/Reference:

QUESTION 128

You want the name, the user puts in to be written back to the monitor. What snippet would you insert in the line indicated below:

```
1 | print('Enter Your Name: ')\n2 | # insert your code here\n3 | print(name)
```

- A. input(name)
- B. name = input
- C. name = input()
- D. input('name')

Correct Answer: C

Explanation

Explanation/Reference:

QUESTION 129

The user enters 123. Which of the following code snippets will not print 124 to the monitor?

- A.

```
1 | num = input('Please enter your number: ')\n2 | print(num + 1)
```

- B.

```
1 num = int(input('Please enter your number: '))
2 print(num + 1)
```
- C.

```
1 num = eval(input('Please enter your number: '))
2 print(num + 1)
```
- D.

```
1 num = input('Please enter your number: ')
2 print(int(num) + 1)
```

Correct Answer: A

Explanation

Explanation/Reference:

QUESTION 130

Consider the following code.

```
1 start = input('How old were you at the time of joining?')
2 now = input('How old are you today?')
```

Which of the following codes will print the right output?

- A.

```
1 print(
2     'Congratulations on '
3     + str(int(now) - int(start))
4     + ' Years of Service!'
5 )
```
- B.

```
1 print(
2     'Congrats on '
3     + (int(now) - int(start))
4     + ' years of service!'
5 )
```
- C.

```
1 print(
2     'Congrats on '
3     + str(now - start)
4     + ' years of service!'
5 )
```
- D.

```
1 print(
2     'Congrats on '
3     + int(now - start)
4     + ' years of service!'
5 )
```

Correct Answer: A

Explanation

Explanation/Reference:

1. Converts now and start to int
2. Performs the subtraction
3. Converts the result back to str for concatenation

QUESTION 131

The ABC company is building a basketball court for its employees to improve company morale. You are creating a Python program that employees can use to keep track of their average score. The program must allow users to enter their name and current scores. The program will output the user name and the user's average score. The output must meet the following requirements:

- The user name must be left-aligned.
- If the user name has fewer than 20 characters, additional space must be added to the right.
- The average score must have three places to the left of the decimal point and one place to the right of the decimal (xxx.x).

What would you insert instead of ??? and ??? ?

```
1 | name = input('What is your name?')
2 | sum = 0
3 | score = 0
4 | count = 0
5 | while score != -1:
6 |     score = int(input('Enter your scores: (-1 to end)'))
7 |     if score == -1:
8 |         break
9 |     sum += score
10 |    count += 1
11 |    average = sum / count
12 |    print('???, your average score is: ???' % (name, average))
```

- A. 1 | %-20f
2 | %4.1s
- B. 1 | %-20i
2 | %4.1
- C. 1 | %-20s
2 | %4.1f
- D. 1 | %-20f
2 | %1.4s
- E. 1 | %-20d
2 | %4.1f
- F. 1 | %-20s
2 | %1.4f

Correct Answer: C

Explanation

Explanation/Reference:

>> Format: %-20s

'%' starts the format specifier

'-' left-aligns the text

'20' reserves 20 character spaces

's' formats as a string

>> Format: %4.1f

'4' is the minimum width

'1' controls the decimal places

'f' formats as a float

QUESTION 132

Consider the following python code:

```
1 | x1 = '23'
2 | y1 = 7
3 | z1 = x1 * y1
4 |
5 | x2 = 42
6 | y2 = 7
7 | z2 = x2 / y2
8 |
9 | x3 = 4.7
10 | y3 = 1
11 | z3 = x3 / y3
```

What are the data types of the variables z1, z2 and z3?

- A. str, str, str
- B. str, float, float
- C. str, int, int
- D. str, int, float

Correct Answer: B

Explanation**Explanation/Reference:****QUESTION 133**

The following code reads two numbers.

Which of the following is the correct input for the code?

```
1 | x, y = eval(input('Enter two numbers: '))
2 | print(x)
3 | print(y)
```

- A. 3, 4
- B. 3 4

- C. None of the above.
- D. `<pre>3 4</pre>`

Correct Answer: A
Explanation

Explanation/Reference:

`eval()` evaluates the input string as Python code, then it tries to unpack the result into two variables, `x` and `y`. So the input must be something `eval()` can interpret as a tuple or expression that gives two values. This is a tuple of two numbers → `eval('3, 4')` becomes `(3, 4)`
Then it unpacks to `x = 3, y = 4`

QUESTION 134

You want to print the sum of two number.

What snippet would you insert in the line indicated below:

```
1 | x = input('Enter the first number: ')\n2 | y = input('Enter the second number: ')\n3 | # insert your code here
```

- A. `print('The Result is ' + (int(x) + int(y)))`
- B. `print('The Result is ' + (int(x + y)))`
- C. `print('The Result is ' + str(int(x + y)))`
- D. `print('The Result is ' + str(int(x) + int(y)))`

Correct Answer: D
Explanation

Explanation/Reference:

QUESTION 135

Consider the following Python code:

```
1 | name = 'Peter'\n2 | age = 23\n3 | flag = True
```

What are the types of the variables `name`, `age` and `flag`?

- A. float, bool, str
- B. str, int, bool
- C. int, bool, char
- D. str, int, int

Correct Answer: B
Explanation

Explanation/Reference:

QUESTION 136

Which of the following is the output of the below Python code?

```
1 | str = 'Hello World'
2 | print(str[::-1])
```

- A. Hello World
- B. dlroW olleH
- C. World
- D. Hello

Correct Answer: B

Explanation

Explanation/Reference:

QUESTION 137

Take a look at the snippet, and choose the true statements: (Choose two.)

```
1 | nums = [1, 2, 3]
2 | vals = nums
3 | del vals[1:2]
```

- A. nums is longer than vals
- B. nums and vals are of the same length
- C. vals is longer than nums
- D. nums and vals refer to the same list

Correct Answer: BD

Explanation

Explanation/Reference:

nums = [1, 2, 3]

Creates a list nums with three elements: [1, 2, 3].

vals = nums

The variable vals is assigned the same reference as nums. Now both vals and nums point to the same list object in memory.

del vals[1:2]

The slice vals[1:2] selects the element at index 1 (value 2).

The del statement removes this element from the list. Since vals and nums point to the same list, the change affects both.

After this operation, the list becomes [1, 3].

Now evaluate the options:

A. nums is longer than vals

False. Both nums and vals refer to the same list, which now has a length of 2.

B. nums and vals are of the same length

True. Both nums and vals are the same list, so their lengths are equal.

C. vals is longer than nums

False. Both nums and vals refer to the same list.

D. nums and vals refer to the same list

True. nums and vals point to the same memory location, so they refer to the same list.

Correct Answers:

B. nums and vals are of the same length

D. nums and vals refer to the same list

QUESTION 138

What is the result of the following code?

```
1 | x = (3, )
2 | print(len(x))
```

A. 2

B. The program will cause an error.

C. 1

D. 3

Correct Answer: C

Explanation

Explanation/Reference:

QUESTION 139

Consider the following code.

```
1 | data = eval(input('Input: '))
2 | print('Output:', data)
```

Which of the inputs below would produce the specified output?

A. None of the above.

B.

```
1 | Input: Hello Python
2 | Output: Hello Python
```

C. Both are correct.

D.

```
1 | Input: [x**2 for x in range(1, 4)]
2 | Output: [1, 4, 9]
```

Correct Answer: D

Explanation

Explanation/Reference:

QUESTION 140

What is the expected output of the following code?

```
1 | data = 'abbabadaadbbaccabc'
2 | print(data.count('ab', 1))
```

- A. 2
- B. 4
- C. 3
- D. 5

Correct Answer: A

Explanation

Explanation/Reference:

QUESTION 141

What is the expected output of the following code?

```
1 | data = [1, 2, [3, 4], [5, 6], 7, [8, 9]]
2 | count = 0
3 |
4 | for i in range(len(data)):
5 |     if type(data[i]) == list:
6 |         count += 1
7 |
8 | print(count)
```

- A. 6
- B. The code is erroneous.
- C. 9
- D. 3

Correct Answer: D

Explanation

Explanation/Reference:

data = [1, 2, [3, 4], [5, 6], 7, [8, 9]]

You are looping through the list data and counting how many elements are lists.

The list has 6 elements:

1, 2, 7 count as 3 elements, while [3, 4], [5, 6], [8, 9] count as an additional 3

count only increases if an element is a list

1, 2, 7 are not lists

[3, 4], [5, 6], [8, 9] are lists, so count = 3

QUESTION 142

Consider the following code.

```
1 | for n in range(1, 6, 1):
2 |     print(??? * 5)
```

What would you insert instead of ???, so that the program prints the following pattern to the monitor?

| | | |
|---|--|-------|
| 1 | | 11111 |
| 2 | | 22222 |
| 3 | | 33333 |
| 4 | | 44444 |
| 5 | | 55555 |

- A. -1
- B. n
- C. str(n)
- D. 1
- E. 2

Correct Answer: C

Explanation

Explanation/Reference:

QUESTION 143

You are creating a Python script to evaluate input and check for upper and lower case.
Code segment 1:

```
1 | else:
2 |     print(name, 'is mixed case.')
```

Code segment 2:

```
1 | else:
2 |     print(name, 'is lower case.')
```

Code segment 3:

```
1 | name = input('Enter your name: ')
```

Code segment 4:

```
1 | else:
2 |     print(name, 'is upper case.')
```

Code segment 5:

```
1 | elif name.upper() == name:
2 |     print(name, 'is all upper case.')
```

Code segment 6:

```
1 | if name.lower() == name:
2 |     print(name, 'is all lower case.')
```

Which four code segments should you use to develop the solution?

- A. Code segment 3 -
Code segment 6 -
Code segment 5 -

Code segment 2

- B. Code segment 1 -
Code segment 3 -
Code segment 5 -
Code segment 6
- C. Code segment 3 -
Code segment 6 -
Code segment 5 -
Code segment 4
- D. Code segment 3 -
Code segment 6 -
Code segment 5 -
Code segment 1

Correct Answer: D
Explanation

Explanation/Reference:

QUESTION 144

Consider the following code.

```
1 room = input('Enter the room number: ')
2 rooms = {101: 'Gathering Place', 102: 'Meeting Room'}
3 if not room in rooms:
4     print('The room doesn\'t exist.')
5 else:
6     print('The room name is: ' + rooms[room])
```

Why is it not working?

- A. Misnamed variable(s)
- B. Mismatched data type(s)
- C. Invalid Syntax
- D. None of the above.

Correct Answer: B
Explanation

Explanation/Reference:

QUESTION 145

You are designing a decision structure to convert a student's numeric grade to a letter grade. The program must assign a letter grade as specified as followed:

```

1 | 90 through 100 -> A
2 | 80 through 89 -> B
3 | 70 through 79 -> C
4 | 65 through 69 -> D
5 | 0 through 64 -> F

```

For example, if the user enters a 90, the output should be Your letter grade is A. Likewise, if a user enters an 89, the output should be Your letter grade is B.

```

1 | # Letter Grade Converter
2 | grade = int(input('Enter a numeric grade:'))
3 | # Line-3
4 |     letter_grade = 'A'
5 | # Line-5
6 |     letter_grade = 'B'
7 | # Line-7
8 |     letter_grade = 'C'
9 | # Line-9
10 |     letter_grade = 'D'
11 | else:
12 |     letter_grade = 'F'
13 | print('Your letter grade is:', letter_grade)

```

Which of the following should you insert on Line-3, Line-5, Line-7 and Line-9?

- A.
- ```

1 | if grade >= 90: # Line-3
2 | elif grade >= 80: # Line-5
3 | elif grade >= 70: # Line-7
4 | elif grade >= 65: # Line-9

```
- B.
- ```

1 | if grade > 90:     # Line-3
2 | elif grade > 80:   # Line-5
3 | elif grade > 70:   # Line-7
4 | elif grade > 65:   # Line-9

```
- C.
- ```

1 | if grade < 90: # Line-3
2 | elif grade < 80: # Line-5
3 | elif grade < 70: # Line-7
4 | elif grade < 65: # Line-9

```

D.

```
1 if grade <= 90: # Line-3
2 elif grade <= 80: # Line-5
3 elif grade <= 70: # Line-7
4 elif grade <= 65: # Line-9
```

**Correct Answer:** A

**Explanation**

**Explanation/Reference:**

#### QUESTION 146

The ABC company is creating a program that allows customers to log the number of miles biked. The program will send messages based on how many miles the customer logs. You create the following Python code.

```
1 ???
2 name = input('What is your name? ')
3 return name
4
5
6 ???
7 calories = miles * calories_per_mile
8 return calories
9
10
11 distance = int(input('How many miles did you bike this week? '))
12 burn_rate = 50
13 biker = get_name()
14 calories_burned = calc_calories(distance, burn_rate)
15 print(biker + ', you burned about', calories_burned, 'calories.')
```

What would you insert instead of ??? and ??? ? (Choose two.)

- A. def calc\_calories(miles, calories\_per\_mile):
- B. def calc\_calories():
- C. def calc\_calories(miles, burn\_rate):
- D. def get\_name(biker):
- E. def get\_name(name):
- F. def get\_name():

**Correct Answer:** AF

**Explanation**

**Explanation/Reference:**

**QUESTION 147**

What is the expected output of the following code?

```
1 | x = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]
2 |
3 | def func(data):
4 | res = data[0][0]
5 | for da in data:
6 | for d in da:
7 | if res < d:
8 | res = d
9 | return res
10 |
11 | print(func(x[0]))
```

- A. 8
- B. 2
- C. 4
- D. 6
- E. The code is erroneous.

**Correct Answer:** C

**Explanation**

**Explanation/Reference:**

**QUESTION 148**

- A.
- B.
- C.
- D.

**Correct Answer:**

**Explanation**

**Explanation/Reference:**

**QUESTION 149**

The ABC organics company needs a simple program that their call center will use to enter survey data for a new coffee variety. The program must accept input and return the average rating based on a five-star scale. The output must be rounded to two decimal places.

You need to complete the code to meet the requirements.

```

1 sum = count = done = 0
2 average = 0.0
3
4 while done != -1:
5 rating = XXX
6 if rating == -1:
7 break
8 sum += rating
9 count += 1
10
11 average = float(sum / count)
12
13 YYY + ZZZ

```

What should you insert instead of XXX, YYY and ZZZ?

- A.
 

```

1 XXX -> float(input('Enter next rating (1-5), -1 for done'))
2 YYY -> print('The average star rating for the new coffee is: '
3 ZZZ -> format(average, '.2f'))

```
- B.
 

```

1 XXX -> input('Enter next rating (1-5), -1 for done')
2 YYY -> print('The average star rating for the new coffee is: '
3 ZZZ -> format(average, '.2d'))

```
- C.
 

```

1 XXX -> float(input('Enter next rating (1-5), -1 for done'))
2 YYY -> printline('The average star rating for the new coffee is: '
3 ZZZ -> format(average, '.2f'))

```
- D.
 

```

1 XXX -> print(input('Enter next rating (1-5), -1 for done'))
2 YYY -> print('The average star rating for the new coffee is: '
3 ZZZ -> format(average, '.2f'))

```
- E.
 

```

1 XXX -> float(input('Enter next rating (1-5), -1 for done'))
2 YYY -> output('The average star rating for the new coffee is: '
3 ZZZ -> format(average, '.2d'))

```
- F.
 

```

1 XXX -> float(input('Enter next rating (1-5), -1 for done'))
2 YYY -> print('The average star rating for the new coffee is: '
3 ZZZ -> format(average, '.2d'))

```

**Correct Answer: A**  
**Explanation**

**Explanation/Reference:**

#### QUESTION 150

You work for a company that distributes media for all ages.

You are writing a function that assigns a rating based on a user's age. The function must meet the following requirements:

- Anyone 18 years old or older receives a rating of A.
- Anyone 13 or older, but younger than 18 receives a rating of T.
- Anyone 12 years old or younger receives a rating of C.
- If the age is unknown, the rating is set to C.

```
1 def get_rating(age):
2 rating = ''
3 if # Line-3
4 elif # Line-4
5 elif # Line-5
6 else # Line-6
7 return rating
```

Which of the following should you insert on Line-3 to Line-6?

- A.
- ```
1      : rating = 'A'          # Line-3
2      age < 18: rating = 'T'   # Line-4
3      age < 13: rating = 'C'   # Line-5
4      age == None: rating = 'C' # Line-6
```
- B.
- ```
1 age == None: rating = 'C' # Line-3
2 age < 18: rating = 'T' # Line-4
3 age < 13: rating = 'C' # Line-5
4 : rating = 'A' # Line-6
```
- C.
- ```
1      age < 13: rating = 'C'     # Line-3
2      age == None: rating = 'C'  # Line-4
3      age < 18: rating = 'T'     # Line-5
4      : rating = 'A'            # Line-6
```
- D.
- ```
1 : rating = 'A' # Line-3
2 age < 13: rating = 'C' # Line-4
3 age < 18: rating = 'T' # Line-5
4 age == None: rating = 'C' # Line-6
```

E.

```
1 | age < 18: rating = 'T' # Line-3
2 | age < 13: rating = 'C' # Line-4
3 | age == None: rating = 'C' # Line-5
4 | : rating = 'A' # Line-6
```

F.

```
1 | age == None: rating = 'C' # Line-3
2 | age < 13: rating = 'C' # Line-4
3 | age < 18: rating = 'T' # Line-5
4 | : rating = 'A' # Line-6
```

**Correct Answer:** F

**Explanation**

**Explanation/Reference:**

### QUESTION 151

Analyze the following code fragments that assign a boolean value to the variable even?

```
1 | num = 42
2 |
3 | # Code-1
4 | if num % 2 == 0:
5 | even = True
6 | else:
7 | even = False
8 |
9 | # Code-2
10 | even = True if num % 2 == 0 else False
11 |
12 | # Code-3
13 | even = num % 2 == 0
```

- A. All three are correct, but Code-2 is preferred.
- B. All three are correct, but Code-3 is preferred.
- C. Code-3 has a syntax error because you attempt to assign a number to even.
- D. All three are correct, but Code-1 is preferred.
- E. Code-2 has a syntax error because you cannot have True and False literals in the conditional expression.

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

**QUESTION 152**

How many stars will the following code send to the monitor?

```
1 | x = 0
2 | while x < 6:
3 | x += 1
4 | if x % 2 == 0:
5 | continue
6 | print('*')
```

- A. one
- B. two
- C. three
- D. zero

**Correct Answer:** C

**Explanation**

**Explanation/Reference:**

x starts at 0. The loop runs while x < 6. x is incremented by 1 on each loop. If x is even (x % 2 == 0), it hits continue, which skips the print(\*\*)

**QUESTION 153**

Consider the following code.

```
1 | nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 | x = 0
3 | while x < 10: # Line 3
4 | print(nums(x)) # Line 4
5 | if nums[x] == 7: # Line 5
6 | break # Line 6
7 | else: # Line 7
8 | x += 1 # Line 8
```

You want to print the numbers 1 to 7 to the monitor. But the code does not work. What do you have to change? (Choose two.)

- A. x = x + 1 # Line 8
- B. while (x < 10): # Line 3
- C. print(nums[x]) # Line 4
- D. if nums[x] == 7: # Line 5

**Correct Answer:** CD

**Explanation**

**Explanation/Reference:**

**QUESTION 154**

Consider the following code.

```
1 | for i in range(5, 0, ???):
2 | print(i, i, i, i, i)
```

What would you insert instead of ???, so that the program prints the following pattern to the monitor?

```
1 | 5 5 5 5 5
2 | 4 4 4 4 4
3 | 3 3 3 3 3
4 | 2 2 2 2 2
5 | 1 1 1 1 1
```

- A. None
- B. 1
- C. 0
- D. 1

**Correct Answer:** B

**Explanation**

**Explanation/Reference:**

**QUESTION 155**

Which of the following sentences correctly describes the output of the below Python code?

```
1 | data = [4, 2, 3, 2, 1]
2 | res = data[0]
3 |
4 | for d in data:
5 | if d < res:
6 | res = d
7 |
8 | print(res)
```

- A. None of the above.
- B. res is the sum of all the number in the list.
- C. res is the smallest number in the list.
- D. res is the largest number in the list.
- E. res is the average of all the number in the list.

**Correct Answer:** C

**Explanation**

**Explanation/Reference:**

**QUESTION 156**

How many stars will the following code print to the monitor?

```
1 | x = 1
2 | while x < 10:
3 | print('*')
4 | x = x << 1
```

- A. two
- B. one
- C. four
- D. eight

**Correct Answer: C**

**Explanation**

**Explanation/Reference:**

**QUESTION 157**

You are coding a math utility by using Python. You are writing a function to compute roots. The function must meet the following requirements:

- If a is non-negative, return  $a^{1/b}$
- If a is negative and even, return 'Result is an imaginary number'
- If a is negative and odd, return  $-(-a)^{1/b}$

Which of the following functions meets the requirements?

- A.
- ```
1 | def safe_root(a, b):
2 |     if a >= 0:
3 |         answer = a ** (1 / b)
4 |     elif a % 2 == 0:
5 |         answer = 'Result is an
imaginary number'
6 |     else:
7 |         answer = -(-a) ** (1 / b)
8 |     return answer
```
- B.
- ```
1 | def safe_root(a, b):
2 | if a % 2 == 0:
3 | answer = a ** (1 / b)
4 | elif a >= 0:
5 | answer = 'Result is an imaginary number'
6 | else:
7 | answer = -(-a) ** (1 / b)
8 | return answer
```

C.

```
1 def safe_root(a, b):
2 if a % 2 == 0:
3 answer = -(-a) ** (1 / b)
4 elif a >= 0:
5 answer = 'Result is an imaginary number'
6 else:
7 answer = a ** (1 / b)
8 return answer
```

D.

```
1 def safe_root(a, b):
2 if a >= 0:
3 answer = -(-a) ** (1 / b)
4 elif a % 2 == 0:
5 answer = 'Result is an imaginary number'
6 else:
7 answer = a ** (1 / b)
8 return answer
```

**Correct Answer:** A

**Explanation**

**Explanation/Reference:**

#### **QUESTION 158**

How much will the delivery cost be, if the order value is 1700 and the state is FL (Florida)?

```

1 | order = int(input('Please enter the order value: '))
2 | state = input('Please enter the state (as postal abbreviation): ')
3 | delivery = 0
4 |
5 | if state in ['NC', 'SC', 'VA']:
6 | if order <= 1000:
7 | delivery = 70
8 | elif 1000 < order < 2000:
9 | delivery = 80
10 | else:
11 | delivery = 90
12 | else:
13 | delivery = 50
14 | if state in ['GA', 'WV', 'FL']:
15 | if order > 1000:
16 | delivery += 30
17 | if order < 2000 and state in ['WV', 'FL']:
18 | delivery += 40
19 | else:
20 | delivery += 25
21 | print(delivery)

```

- A. 120
- B. 105
- C. 80
- D. 90

**Correct Answer:** A

**Explanation**

**Explanation/Reference:**

```

order = int(input('Please enter the order value: '))
state = input('Please enter the state (as postal abbreviation): ')
delivery = 0

```

```

if state in ['NC', 'SC', 'VA']:
if order <= 1000:
delivery = 70
elif 1000 < order < 2000:
delivery = 80
else:

```

```
delivery = 90

else:
 delivery = 50
 if state in ['GA', 'WV', 'FL']:
 if order > 1000:
 delivery += 30
 if order < 2000 and state in ['WV', 'FL']:
 delivery += 40
 else:
 delivery + 25

print (delivery)
```

#### QUESTION 159

How many stars will the following code print to the monitor?

```
1 | i = 0
2 | while i < i + 2:
3 | i += 1
4 | print('*')
5 | else:
6 | print('*')
```

- A. two
- B. one
- C. The snippet will enter an infinite loop.
- D. zero

**Correct Answer:** C

**Explanation**

**Explanation/Reference:**

#### QUESTION 160

What will happen when you attempt to run the following code?

```
1 | While True:
2 | print("1")
```

- A. The program will not run due to syntax error.
- B. The program will run and cause an error.
- C. The program will fall into infinite loop, printing 1 on each line.
- D. The program will print 1 on one line only.

**Correct Answer:** C

**Explanation**

**Explanation/Reference:**

**QUESTION 161**

Consider the following code.

```
1 | data = ['Peter', 'Paul', 'Mary', 'Jane']
2 | res = 0
```

Which of the following code snippets will expand the code, so that 100 will be printed to the monitor? (Choose two.)

A.

```
1 | for i in ('Peter', 'Steve', 'Jane'):
2 | if i in data:
3 | res += 100
4 | print(res)
```

B.

```
1 | for i in ('Peter', 'Steve', 'Jane'):
2 | if i in data:
3 | res += 50
4 | print(res)
```

C.

```
1 | for i in ('Peter', 'Steve', 'Jane'):
2 | if i not in data:
3 | res += 50
4 | print(res)
```

D.

```
1 | for i in ('Peter', 'Steve', 'Jane'):
2 | if i not in data:
3 | res += 100
4 | print(res)
```

**Correct Answer:** BD

**Explanation**

**Explanation/Reference:**

**QUESTION 162**

What is the expected output of the following code?

```
1 | def func(text, num):
2 | while num > 0:
3 | print(text)
4 | num = num - 1
5 |
6 | func('Hello', 3)
```

A. 1 | Hello  
2 | Hello

- B. 1 | Hello  
2 | Hello  
3 | Hello
- C. 1 | Hello  
2 | Hello  
3 | Hello  
4 | Hello
- D. An infinite loop.

**Correct Answer:** D  
**Explanation**

**Explanation/Reference:**

#### QUESTION 163

What is the expected output of the following code?

```
1 | x = True
2 | y = False
3 | z = False
4 |
5 | if not x or y:
6 | print(1)
7 | elif not x or not y and z:
8 | print(2)
9 | elif not x or y or not y and x:
10 | print(3)
11 | else:
12 | print(4)
```

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

**Correct Answer:** A  
**Explanation**

**Explanation/Reference:**

#### QUESTION 164

How many stars will the following snippet print to the monitor?

```

1 | i = 4
2 | while i > 0:
3 | i -= 2
4 | print('*')
5 | if i == 2:
6 | break
7 | else:
8 |
9 | print('*')

```

- A. The snippet will enter an infinite loop.
- B. 2
- C. 0
- D. 1

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

#### QUESTION 165

Which of the following for loops would output the below number pattern?

```

1 | 11111
2 | 22222
3 | 33333
4 | 44444
5 | 55555

```

A.

```

1 | for i in range(1, 6):
2 | print(str(i) * 5)

```

B.

```

1 | for i in range(1, 5):
2 | print(str(i) * 5)

```

C.

```

1 | for i in range(0, 5):
2 | print(str(i) * 5)

```

D.

```
1 | for i in range(1, 6):
2 | print(i, i, i, i, i)
```

**Correct Answer:** A

**Explanation**

**Explanation/Reference:**

#### QUESTION 166

What happens when the user runs the following code?

```
speed = 3
while speed < 8:
 speed += 2
 if speed == 7:
 continue
 print("*", end="")
else:
 print("*")
```

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

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

#### QUESTION 167

How many hashes ( # ) does the code output to the screen?

```
floor = 0
while floor != 0:
 floor -= 1
 print("#", end="")
else:
 print("#")
```

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

**Correct Answer:** D

**Explanation**

Explanation/Reference:

### QUESTION 168

DRAG DROP

Arrange the code boxes in the correct positions in order to obtain a loop which executes its body with the counter variable going through values 1, 3, and 5 (in the same order).

|         |    |     |
|---------|----|-----|
| range   | (  | 1,  |
| counter | )  | 2   |
| in      | 7, | for |

A.

```
for counter in range(1, 7, 2)
```

|         |    |     |
|---------|----|-----|
| range   | (  | 1,  |
| counter | )  | 2   |
| in      | 7, | for |

Correct Answer: A

Explanation

Explanation/Reference:

### QUESTION 169

What is the expected output of the following code?

```
menu = {"canistrelli": 1.12, "vol-au-vent": 2.99, "gougere": 0.99}
```

```
for value in menu.keys():
 print(str(value)[1], end="")
```

- A. aoo
- B. The code is erroneous and cannot be run.
- C. ...
- D. canistrellivol

Correct Answer: A

Explanation

**Explanation/Reference:**

**QUESTION 170**

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:
 break
print(equals)
```

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

**Correct Answer: A**

**Explanation**

**Explanation/Reference:**

**QUESTION 171**

What is the expected result of running the following code?

```
def do_the_mess(parameter):
 parameter = [variable]
 return parameter

the_list = [x for x in range(0, 1)]
variable = -2
do_the_mess(the_list)
print(the_list[0])
```

- A. The code prints 2
- B. The code prints 1
- C. The code prints 0
- D. The code raises an unhandled exception.

**Correct Answer: C**

**Explanation**

**Explanation/Reference:**

**QUESTION 172**

What is the expected output of the following code?

```
menu = {"syrniki": 12.8, "shashlik": 49.9, "borscht": 23.2}

for value in menu.items():
 print(value[1], end="")
```

- A. 293
- B. yh
- C. The code is erroneous and cannot be run.
- D. 12.849.923.2

**Correct Answer: D**  
**Explanation**

**Explanation/Reference:**

#### QUESTION 173

What is the expected result of the following code?

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

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

**Correct Answer: C**  
**Explanation**

**Explanation/Reference:**

#### QUESTION 174

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

**Correct Answer: B**

**Explanation**

**Explanation/Reference:**

**QUESTION 175**

What is the expected output of the following code?

```
print(3 * 'abc' + 'xyz')
```

- A. 3abcxyz
- B. abcabcxyzxyz
- C. abcabcabcxyz
- D. abcxyzabcxyzabcxyz

**Correct Answer: C**

**Explanation**

**Explanation/Reference:**

**QUESTION 176**

Consider the following code.

```
x = float('23.42')
```

Which of the following expressions will evaluate to 2?

- A. int(x) + False
- B. bool(x) + True
- C. str(x)
- D. bool(x)

**Correct Answer: B**

**Explanation**

**Explanation/Reference:**

bool(x) converts x (which is 23.42) to a boolean. Since x is a non-zero value, it becomes True.

True is treated as 1 in numerical operations.

True + True is 1 + 1, which evaluates to 2.

**QUESTION 177**

Which of the following statements can be used to return the length of the given string str?

- A. str.size()
- B. str.\_len\_()
- C. size(str)
- D. len(str)

**Correct Answer: D**

**Explanation**

**Explanation/Reference:**

**QUESTION 178**

What is the expected output of the following code?

```
print(float('1.3'))
```

- A. The code is erroneous.
- B. 1,3
- C. 13
- D. 1.3

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

#### **QUESTION 179**

What is the decimal value of the following binary number?

- A. 8
- B. 4
- C. 12
- D. 10

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

1010 = 10

First 1 (left) is in the 8 bit location

Second 1 is in the 2 bit location

0 bits are turned off and not counted

$8 + 2 = 10$

Bit values from right to left are 1, 2, 4, 8

#### **QUESTION 180**

The most important difference between integer and floating-point numbers lies in the fact that:

- A. they are stored differently in the computer memory
- B. they cannot be used simultaneously
- C. integers cannot be literals, while floats can

**Correct Answer:** A

**Explanation**

**Explanation/Reference:**

#### **QUESTION 181**

You want to print each name of the list on a new line.

`data = ['Peter', 'Paul', 'Mary', 'Jane']`

Which statement will you use?

- A. `print(data.concatenate("\n"))`
- B. `print("\n".join(data))`
- C. `print(data.join("\n"))`
- D. `print(data.join('%s\n', names))`

**Correct Answer:** B

### Explanation

### Explanation/Reference:

#### QUESTION 182

What is the expected output of the following code?

```
print(ord('c') - ord('a'))
```

- A. 1
- B. 2
- C. 3
- D. 0

**Correct Answer: B**

### Explanation

### Explanation/Reference:

ord('a'): The Unicode code point for the character 'a' is 97.

ord('c'): The Unicode code point for the character 'c' is 99.

ord('c') - ord('a') = 99 - 97 = 2

#### QUESTION 183

What is the expected output of the following code?

```
print('x', 'y', 'z', sep='sep')
```

- A. x y z
- B. xyz
- C. xsepysepzsep
- D. xsepysepz

**Correct Answer: D**

### Explanation

### Explanation/Reference:

'x', 'y', and 'z' are the values to print.

sep: what to put between each object (default is a space ' ')

sep='sep' means the string 'sep' will be used between them.

#### QUESTION 184

What is the expected output of the following code?

```
print(type(1J))
```

- A. <type 'dict'>
- B. <type 'unicode'>
- C. <type 'float'>
- D. <type 'complex'>

**Correct Answer: D**

### Explanation

### Explanation/Reference:

#### QUESTION 185

Which of the following operators can be used with strings?

- 1) +
- 2) \*
- 3) -
- 4) in

- A. 1, 2, 3
- B. 1, 2, 4
- C. 1, 2, 3, 4
- D. 1, 2

**Correct Answer:** B

**Explanation**

**Explanation/Reference:**

#### **QUESTION 186**

Which of the following statements is false?

- A. The None value can be compared with variables.
- B. The None value can not be used as an argument of arithmetic operators.
- C. The None value may not be used outside functions.
- D. The None value can be assigned to variables.

**Correct Answer:** C

**Explanation**

**Explanation/Reference:**

#### **QUESTION 187**

The value thirty point eleven times ten raised to the power of nine should be written as:

- A. 30.11E9.0
- B. 30E11.9
- C. 30.11\*10^9
- D. 30.11E9

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

#### **QUESTION 188**

If you want to build a string that reads:

Peter's sister's name's "Anna"

Which of the following literals would you use? (Choose two.)

- A. 'Peter's sister's name's "Anna"'
- B. 'Peter\'s sister\'s name\'s \'Anna\''
- C. "Peter's sister's name's "Anna""
- D. "Peter's sister's name's \'Anna\'"

**Correct Answer:** BD

### **Explanation**

### **Explanation/Reference:**

#### **QUESTION 189**

You want to write a program that asks the user for a value. For the rest of the program you need a whole number, even if the user enters a decimal value. What would you have to write?

- A. `num = str(input('How many do you need?'))`
- B. `num = int(float(input('How many do you need?')))`
- C. `num = float(input('How many do you need?'))`
- D. `num = int('How many do you need?')`

**Correct Answer: B**

### **Explanation**

### **Explanation/Reference:**

The `input()` function gets the user's input as a string. `float()` converts that string into a floating-point number (allowing for decimal input). `int()` then converts the float to an integer, effectively discarding the decimal part.

#### **QUESTION 190**

The `0o` prefix means that the number after it is denoted as:

- A. decimal
- B. binary
- C. hexadecimal
- D. octal

**Correct Answer: D**

### **Explanation**

### **Explanation/Reference:**

#### **QUESTION 191**

How many arguments can the `print()` function take?

- A. Any number of arguments (excluding zero).
- B. Not more than seven arguments.
- C. Any number of arguments (including zero).
- D. Just one argument.

**Correct Answer: C**

### **Explanation**

### **Explanation/Reference:**

#### **QUESTION 192**

Which of the following statements are true? (Choose two.)

- A. The `None` value can be assigned to variables
- B. The `None` value can be used as an argument of arithmetic operators

- C. The None value can be compared with variables
- D. The None value cannot be used outside functions

**Correct Answer:** AC

**Explanation**

**Explanation/Reference:**

#### **QUESTION 193**

What is the expected output of the following code?  
`print(chr(ord('z') - 2))`

- A. x
- B. y
- C. a
- D. z

**Correct Answer:** A

**Explanation**

**Explanation/Reference:**

#### **QUESTION 194**

How many elements will the following list contain?  
`data = [i for i in range(-1, 2)]`

- A. one
- B. two
- C. zero
- D. four
- E. three

**Correct Answer:** E

**Explanation**

**Explanation/Reference:**

#### **QUESTION 195**

How many stars will the following snippet print to the monitor? `x = 16 while x > 0: print('*') x //= 2`

- A. one
- B. three
- C. The code will enter an infinite loop.
- D. five

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

#### **QUESTION 196**

What is the expected output of the following code?  
`print(len([i for i in range(0, -2)]))`

- A. 1
- B. 3
- C. 2
- D. 0

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

#### QUESTION 197

Assuming that the following assignment has been successfully executed:

```
my_list = [1, 2, 4, 8]
```

Select the expressions which will not raise any exception. (Choose two.)

- A. `my_list[4]`
- B. `my_list[my_list[3]]`
- C. `my_list[-2]`
- D. `my_list[-3:-2]`

**Correct Answer:** CD

**Explanation**

**Explanation/Reference:**

#### QUESTION 198

Which of the following snippets correctly define a function which returns its only argument doubled? (Choose two.)

- A. 

```
def double(value):
 return 2 * value
```
- B. 

```
def 2_times_arg(in):
 return 2 * in
```
- C. 

```
def multiply_by_2:
 value *= 2
```
- D. 

```
def times_2(x):
 return x + x
```

**Correct Answer:** AD

**Explanation**

**Explanation/Reference:**

#### QUESTION 199

A program written in a high-level programming language is called:

- A. a source code
- B. machine code

- C. a binary code
- D. the ASCII code

**Correct Answer:** D

**Explanation**

**Explanation/Reference:**

#### **QUESTION 200**

What is true about exceptions and debugging? (Choose two.)

- A. A tool that allows you to precisely trace program execution is called a debugger.
- B. One try-except block may contain more than one except branch.
- C. The default (anonymous) except branch cannot be the last branch in the try-except block.
- D. If some Python code is executed without errors, this proves that there are no errors in it.

**Correct Answer:** AB

**Explanation**

**Explanation/Reference:**

#### **QUESTION 201**

Which of the following functions can be invoked with one argument?

- A. `def theta(None):`  
`pass`
- B. `def eta(level, size, depth = 0):`  
`pass`
- C. `def epsilon(level=100):`  
`pass`
- D. `def zeta():`  
`pass`

**Correct Answer:** C

**Explanation**

**Explanation/Reference:**

#### **QUESTION 202**

Assuming that the following assignment has been successfully executed:

```
the_list = [True, 3.1415, -1]
```

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

- A. `len(sorted(the_list)) != len(the_list)`
- B. `"True" in the_list`
- C. `the_list.index() == 2`
- D. `(len(the_list) - 3) in the_list`

**Correct Answer:** BC

**Explanation**

**Explanation/Reference:**

**QUESTION 203**

Which of the following expressions evaluate to zero? (Choose two.)

- A.  $2 / -3 * 6 + 4$
- B.  $-3 / 2 * 4 + 1$
- C.  $3 ** 2 // 3 / 3 - 1$
- D.  $2 // 2 * 2 + 3$

**Correct Answer:** AC

**Explanation**

**Explanation/Reference:**

**QUESTION 204**

Assuming that the following assignment has been successfully executed:

```
the_list = ['list', False, 3e8]
```

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

- A. `int(the_list[2]) == len(the_list)`
- B. `the_list[1] in the_list`
- C. `300 in the_list and the_list[l]`
- D. `the_list.index(False) == 1`

**Correct Answer:** BD

**Explanation**

**Explanation/Reference:**