### **Question 21:**

What will the following code print?

x = [1, 2, 3]

y = x.copy()

y.append(4)

print(x)

A) [1, 2, 3]  
 B) [1, 2, 3, 4]  
 C) [4, 1, 2, 3]  
 D) Error

The correct answer is **A) [1, 2, 3]**.

Explanation:

* The method x.copy() creates a **shallow copy** of the list x. This means that y is a new list, independent of x.
* When you append 4 to y, it only affects y, not x.
* Therefore, after appending 4 to y, x remains unchanged as [1, 2, 3], while y becomes [1, 2, 3, 4].

### **Question 24:**

What does the following code output?

x = {1, 2, 3}

y = {3, 4, 5}

z = x & y

print(z)

A) {1, 2, 3}  
B) {3, 4, 5}  
C) {3}  
D) {1, 2, 4, 5}

The correct answer is **C) {3}**.

Explanation:

* The & operator is used to find the **intersection** of two sets in Python. It returns a new set containing only the elements that are common to both sets.
* In this case, the common element between x = {1, 2, 3} and y = {3, 4, 5} is 3, so z will be {3}.

### **Question 38:**

What will be the output of the following code?

x = [1, 2, 3]

y = x[:]

y[0] = 10

print(x)

A) [1, 2, 3]  
B) [10, 2, 3]  
C) [10, 2, 3] (new reference)  
D) [1, 2, 10]

The correct answer is **A) [1, 2, 3]**.

Explanation:

* The expression x[:] creates a **shallow copy** of the list x. This means y is a new list that contains the same elements as x, but x and y are different objects.
* When you modify y[0] = 10, only y is affected, not x, because they are separate lists.
* Therefore, x remains [1, 2, 3], while y becomes [10, 2, 3].

### 

### 

### 

### 

### **Question 39:**

What is the output of the following code?

x = [1, 2, 3]

y = x

y = y + [4, 5]

print(x)

A) [1, 2, 3]  
B) [1, 2, 3, 4, 5]  
C) [4, 5]  
D) [1, 2, 4, 5]

The correct answer is **A) [1, 2, 3]**.

Explanation:

* Initially, x = [1, 2, 3] and y = x. So, y is a reference to the same list as x.
* When you do y = y + [4, 5], it creates a **new list** and reassigns y to that new list, which is [1, 2, 3, 4, 5]. However, this does not modify the original list x, since y is now referencing a new list.
* Therefore, x remains [1, 2, 3].

**Set and Boolean Operations in Python**

### **1. Set Operations:**

**Union (OR) - |**: Combines all unique elements from two sets.  
  
 x = {1, 2, 3}

y = {3, 4, 5}

print(x | y) # Output: {1, 2, 3, 4, 5}

* **Intersection (AND) - &**: Returns only the elements common to both sets.

x = {1, 2, 3}

y = {3, 4, 5}

print(x & y) # Output: {3}

### **2. Boolean Operations:**

**OR - or**: Returns True if at least one condition is True.  
  
 a = False

b = True

print(a or b) # Output: True

* **AND - and**: Returns True only if both conditions are True.  
    
   a = True

b = False

print(a and b) # Output: False

—---------------------------------FALTU-----------------------------------------------------------------

Log in

Sign up

##### **You said:**

make a test series type mcq for python askk one by one with option then i give answer then u verify its correct or not

###### **ChatGPT said:**

ChatGPT

Sure! Let’s start with the first question.

### **Question 1:**

What is the output of the following Python code?

python

Copy

x = 10

y = 5

x, y = y, x

print(x, y)

A) 10 5  
B) 5 10  
C) 10 10  
D) 5 5

What is your answer?

##### **You said:**

a

###### **ChatGPT said:**

ChatGPT

The correct answer is **B) 5 10**.

Explanation: The line x, y = y, x swaps the values of x and y. After the swap:

* x becomes 5 (value of y)
* y becomes 10 (value of x)

Let's move on to the next question!

### **Question 2:**

Which of the following is the correct way to create a set in Python?

A) my\_set = {1, 2, 3}  
B) my\_set = [1, 2, 3]  
C) my\_set = (1, 2, 3)  
D) my\_set = set(1, 2, 3)

What is your answer?

##### **You said:**

c

###### **ChatGPT said:**

ChatGPT

The correct answer is **A) my\_set = {1, 2, 3}**.

Explanation:

* **A)** {1, 2, 3} is the correct syntax for creating a set in Python.
* **B)** [1, 2, 3] creates a list, not a set.
* **C)** (1, 2, 3) creates a tuple, not a set.
* **D)** set(1, 2, 3) is incorrect syntax for creating a set.

Next question coming up!

### **Question 3:**

What is the output of the following Python code?

python

Copy

def func(a, b=5, c=10):

return a + b + c

print(func(2))

A) 17  
B) 12  
C) 10  
D) Error: missing argument for c

What is your answer?

##### **You said:**

a

###### **ChatGPT said:**

ChatGPT

The correct answer is **A) 17**.

Explanation:

* The function func has three parameters: a, b (with a default value of 5), and c (with a default value of 10).
* When calling func(2), the argument a is set to 2, and the default values for b and c (5 and 10, respectively) are used.
* So, the calculation becomes 2 + 5 + 10 = 17.

You're doing great! Let's move to the next question.

### **Question 4:**

Which of the following statements is true about Python dictionaries?

A) Dictionaries can have duplicate keys.  
B) Dictionaries maintain the order of elements in all versions of Python.  
C) A dictionary key can be any data type, including lists.  
D) A dictionary is an unordered collection of key-value pairs (prior to Python 3.7).

What is your answer?

##### **You said:**

D

###### **ChatGPT said:**

ChatGPT

The correct answer is **D) A dictionary is an unordered collection of key-value pairs (prior to Python 3.7).**

Explanation:

* **D)** In Python versions prior to 3.7, dictionaries do not maintain the insertion order of elements. Starting from Python 3.7, dictionaries do maintain insertion order, but this was not the case in earlier versions.
* **A)** Dictionaries do not allow duplicate keys. Each key must be unique.
* **B)** As mentioned, dictionaries only started maintaining order in Python 3.7+, so this is not true for all versions.
* **C)** Dictionary keys must be immutable (e.g., strings, numbers, or tuples), so lists cannot be used as dictionary keys.

Let's continue!

### **Question 5:**

What will be the output of the following code?

python

Copy

x = [1, 2, 3]

y = x

y[0] = 99

print(x)

A) [99, 2, 3]  
B) [1, 2, 3]  
C) [99, 99, 99]  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

The correct answer is **A) [99, 2, 3]**.

Explanation:

* In the line y = x, both x and y refer to the same list object in memory. So, modifying y[0] also modifies x[0] because they are pointing to the same list.
* Therefore, when you change y[0] to 99, x[0] also becomes 99, and the output is [99, 2, 3].

Let's move on!

### **Question 6:**

What will the following code output?

python

Copy

print(type(3.14))

A) <class 'float'>  
B) <class 'int'>  
C) <class 'str'>  
D) <class 'complex'>

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) <class 'float'>**.

Explanation:

* 3.14 is a floating-point number, so type(3.14) will return <class 'float'>.

You're on a roll! Here's the next question.

### **Question 7:**

What does the zip() function do in Python?

A) It joins two strings into one.  
B) It merges two or more iterables into a single iterable of tuples.  
C) It sorts the elements of a list.  
D) It extracts the first element from multiple iterables.

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) It merges two or more iterables into a single iterable of tuples**.

Explanation:

* The zip() function takes two or more iterables (like lists or tuples) and aggregates them into a single iterable of tuples, where the first tuple contains the first element from each iterable, the second tuple contains the second element, and so on.

For example:

python

Copy

a = [1, 2, 3]

b = ['a', 'b', 'c']

zipped = zip(a, b)

print(list(zipped)) # Output: [(1, 'a'), (2, 'b'), (3, 'c')]

Let's keep going!

### **Question 8:**

What is the result of the following expression?

python

Copy

"Hello" \* 3

A) "HelloHelloHello"  
B) "Hello 3"  
C) 3Hello  
D) Error

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) "HelloHelloHello"**.

Explanation:

* In Python, multiplying a string by an integer repeats the string that many times. So, "Hello" \* 3 results in the string "HelloHelloHello".

You're doing great! Let's move on to the next question.

### **Question 9:**

Which of the following is true about Python lists?

A) Lists are immutable.  
B) Lists can contain elements of different data types.  
C) Lists cannot be nested.  
D) Lists can only store integers.

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) Lists can contain elements of different data types**.

Explanation:

**B)** Python lists are versatile and can contain elements of different data types, including integers, strings, floats, and even other lists.  
Example:  
python  
Copy  
my\_list = [1, "Hello", 3.14, [2, 3]]

* **A)** Lists are mutable, meaning you can modify them after they are created.
* **C)** Lists can be nested (i.e., a list can contain other lists).
* **D)** Lists can store any type of object, not just integers.

Let's go to the next one!

### **Question 10:**

What is the output of the following code?

python

Copy

x = [1, 2, 3]

x.append([4, 5])

print(x)

A) [1, 2, 3, 4, 5]  
B) [1, 2, 3, [4, 5]]  
C) [[1, 2, 3], 4, 5]  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) [1, 2, 3, [4, 5]]**.

Explanation:

* The append() method adds its argument as a single element to the end of the list. So when you do x.append([4, 5]), it adds the list [4, 5] as a single element to x.
* The resulting list is [1, 2, 3, [4, 5]], where the last element is itself a list.

Great job! Here comes the next question.

### **Question 11:**

Which of the following is the correct way to create a tuple in Python?

A) my\_tuple = [1, 2, 3]  
B) my\_tuple = (1, 2, 3)  
C) my\_tuple = {1, 2, 3}  
D) my\_tuple = 1, 2, 3

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) my\_tuple = (1, 2, 3)**.

Explanation:

* **B)** This is the correct syntax to create a tuple in Python. Tuples are enclosed in parentheses ().

Other options:

* **A)** [1, 2, 3] creates a list, not a tuple.
* **C)** {1, 2, 3} creates a set, not a tuple.
* **D)** 1, 2, 3 is also a valid way to create a tuple (without parentheses), as Python implicitly interprets this as a tuple when there are multiple values separated by commas.

Let's move on!

### **Question 12:**

What does the following code print?

python

Copy

x = 7

def test():

global x

x = 10

test()

print(x)

A) 7  
B) 10  
C) Error  
D) None

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) 10**.

Explanation:

* The global keyword in the function test() allows you to modify the global variable x. Without global, Python would treat x as a local variable inside the function, but by using global, you modify the global x directly.
* Inside the function test(), x is set to 10, and this change is reflected when you print x outside the function.

You're doing excellent! Here's the next question.

### **Question 13:**

What is the purpose of the self keyword in Python?

A) To refer to the instance of a class.  
B) To refer to the class itself.  
C) To define a method in a class.  
D) To access global variables.

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) To refer to the instance of a class**.

Explanation:

* In Python, self refers to the instance of the class, allowing you to access instance variables and methods from within the class. It is the first parameter in instance methods and is automatically passed when you call a method on an object.

Example:

python

Copy

class MyClass:

def \_\_init\_\_(self, value):

self.value = value # 'self' refers to the instance

def display(self):

print(self.value)

obj = MyClass(10)

obj.display() # Output: 10

Let’s keep going!

### **Question 14:**

What will be the output of this code?

python

Copy

x = 4

y = 5

z = x == y

print(z)

A) True  
B) False  
C) 4 == 5  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) False**.

Explanation:

* The expression x == y checks if x is equal to y. Since x is 4 and y is 5, the comparison 4 == 5 is False.
* Therefore, z is assigned the value False, and print(z) outputs False.

Nice work! Here's the next question.

### **Question 15:**

What is the output of the following code?

python

Copy

x = [1, 2, 3]

y = x

y[1] = 10

print(x)

A) [1, 10, 3]  
B) [1, 2, 3]  
C) [10, 2, 3]  
D) [1, 2, 10]

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) [1, 10, 3]**.

Explanation:

* In the line y = x, both x and y point to the same list in memory. So, when you modify y[1] = 10, it also modifies x[1], since both refer to the same list.
* Therefore, the list x becomes [1, 10, 3].

You're doing great! Let's move to the next one.

### **Question 16:**

Which of the following statements about Python's range() function is true?

A) range() generates a list of numbers.  
B) range() generates an iterable object, not a list.  
C) range() only works with integers.  
D) range() can only be used in loops.

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) range() generates an iterable object, not a list**.

Explanation:

* The range() function returns a range object, which is an iterable, not a list. However, you can convert it to a list by using list(range(...)).

Example:

python

Copy

r = range(5) # This creates a range object, not a list.

print(type(r)) # Output: <class 'range'>

print(list(r)) # Output: [0, 1, 2, 3, 4]

Other points:

* **A)** range() doesn't generate a list directly, it returns an iterable object that can be looped over.
* **C)** range() works with integers, but not just integers; you can specify a start, stop, and step value (which are integers).
* **D)** range() is commonly used in loops, but it's not restricted to loops. You can use it in other places where an iterable is needed.

Let's go on!

### **Question 17:**

What will the following code output?

python

Copy

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

del x['b']

print(x)

A) {'a': 1, 'b': 2, 'c': 3}  
B) {'a': 1, 'c': 3}  
C) Error: key 'b' not found  
D) {'a': 1, 'b': 2}

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) {'a': 1, 'c': 3}**.

Explanation:

* The del statement removes a key-value pair from the dictionary. In this case, del x['b'] removes the key 'b' and its corresponding value 2.
* After the deletion, the dictionary becomes {'a': 1, 'c': 3}.

Great job! Here's the next question.

### **Question 18:**

Which of the following methods is used to add an item to a Python set?

A) set.add(item)  
B) set.append(item)  
C) set.insert(item)  
D) set.put(item)

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

The correct answer is **A) set.add(item)**.

Explanation:

* The add() method is used to add a single item to a set in Python.

Example:

python

Copy

my\_set = {1, 2, 3}

my\_set.add(4)

print(my\_set) # Output: {1, 2, 3, 4}

* **B)** append() is used for lists, not sets.
* **C)** insert() is used for inserting elements at a specific position in a list, not a set.
* **D)** put() is not a valid method for sets in Python.

Let's keep going!

### **Question 19:**

Which of the following will throw a TypeError?

A) 1 + 1  
B) 'Hello' + ' ' + 'World'  
C) [] + []  
D) 1 + '1'

What is your answer?

##### **You said:**

C

###### **ChatGPT said:**

ChatGPT

The correct answer is **D) 1 + '1'**.

Explanation:

* **D)** Adding an integer (1) and a string ('1') will throw a TypeError because Python cannot implicitly combine these two different data types.

Example:

python

Copy

1 + '1' # This will raise a TypeError

Other options:

* **A)** 1 + 1 results in 2 and does not raise an error.
* **B)** 'Hello' + ' ' + 'World' results in the string 'Hello World' and is valid.
* **C)** [] + [] results in an empty list [] and is valid.

You're doing well! Here's the next question.

### **Question 20:**

Which of the following is the correct way to open a file for reading in Python?

A) open('file.txt', 'r')  
B) open('file.txt', 'w')  
C) open('file.txt', 'rw')  
D) open('file.txt', 'read')

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) open('file.txt', 'r')**.

Explanation:

* **A)** The correct mode for opening a file for reading is 'r'. This allows you to read the contents of the file.

Example:

python

Copy

file = open('file.txt', 'r')

content = file.read()

file.close()

Other options:

* **B)** 'w' mode is for writing to a file (it will overwrite the file if it exists).
* **C)** 'rw' is not a valid mode in Python.
* **D)** 'read' is not a valid mode; 'r' should be used for reading.

Great work! Here's the next question.

### **Question 21:**

What will the following code print?

python

Copy

x = [1, 2, 3]

y = x.copy()

y.append(4)

print(x)

A) [1, 2, 3]  
B) [1, 2, 3, 4]  
C) [4, 1, 2, 3]  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

The correct answer is **A) [1, 2, 3]**.

Explanation:

* The method x.copy() creates a **shallow copy** of the list x. This means that y is a new list, independent of x.
* When you append 4 to y, it only affects y, not x.
* Therefore, after appending 4 to y, x remains unchanged as [1, 2, 3], while y becomes [1, 2, 3, 4].

Good job! Here comes the next question.

### **Question 22:**

What does the following code output?

python

Copy

def my\_func(\*args):

return len(args)

print(my\_func(1, 2, 3))

A) 1  
B) 3  
C) Error  
D) None

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) 3**.

Explanation:

* The \*args syntax allows a function to accept a variable number of positional arguments. Inside the function, args is treated as a tuple containing all the arguments passed.
* In the code, the function my\_func(1, 2, 3) receives three arguments, so args will be (1, 2, 3), and len(args) will return 3.

Great work! Here's the next question.

### **Question 23:**

Which of the following is used to check if a key exists in a dictionary?

A) key() in dict  
B) key in dict  
C) dict.has\_key(key)  
D) key.exists(dict)

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) key in dict**.

Explanation:

* The correct way to check if a key exists in a dictionary is using the in keyword: key in dict.

Example:

python

Copy

my\_dict = {'a': 1, 'b': 2, 'c': 3}

print('a' in my\_dict) # Output: True

print('d' in my\_dict) # Output: False

Other options:

* **A)** key() in dict is not valid syntax.
* **C)** dict.has\_key(key) was used in Python 2 but is not available in Python 3.
* **D)** key.exists(dict) is not a valid way to check for a key in a dictionary.

You're doing awesome! Here’s the next question.

### **Question 24:**

What does the following code output?

python

Copy

x = {1, 2, 3}

y = {3, 4, 5}

z = x & y

print(z)

A) {1, 2, 3}  
B) {3, 4, 5}  
C) {3}  
D) {1, 2, 4, 5}

What is your answer?

##### **You said:**

D

###### **ChatGPT said:**

ChatGPT

The correct answer is **C) {3}**.

Explanation:

* The & operator is used to find the **intersection** of two sets in Python. It returns a new set containing only the elements that are common to both sets.
* In this case, the common element between x = {1, 2, 3} and y = {3, 4, 5} is 3, so z will be {3}.

Other options:

* **D)** {1, 2, 4, 5} would be the union of the two sets, which can be obtained with the | operator, not &.

You're doing great! Here's the next question.

### **Question 25:**

Which of the following is the correct way to create a generator in Python?

A) gen = (x \* 2 for x in range(5))  
B) gen = [x \* 2 for x in range(5)]  
C) gen = {x \* 2 for x in range(5)}  
D) gen = <x \* 2 for x in range(5)>

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

The correct answer is **A) gen = (x \* 2 for x in range(5))**.

Explanation:

* **A)** This is the correct syntax for creating a generator in Python. A generator expression is written inside parentheses (). It yields values one by one as you iterate over it, which is more memory-efficient than creating a list.

Example:

python

Copy

gen = (x \* 2 for x in range(5))

for value in gen:

print(value) # Output: 0, 2, 4, 6, 8

Other options:

* **B)** [x \* 2 for x in range(5)] creates a **list**, not a generator. This creates a list containing the values [0, 2, 4, 6, 8].
* **C)** {x \* 2 for x in range(5)} creates a **set** comprehension, not a generator.
* **D)** <x \* 2 for x in range(5)> is invalid syntax for a generator.

Great job! Let's move on.

### **Question 26:**

What is the purpose of the \_\_init\_\_ method in a Python class?

A) It initializes class-level variables.  
B) It initializes the instance variables when an object is created.  
C) It is used to define class methods.  
D) It is used for inheritance.

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) It initializes the instance variables when an object is created**.

Explanation:

* The \_\_init\_\_ method in Python is the **constructor** method. It is automatically called when a new object of a class is created. This method initializes the instance variables (attributes) of the class.

Example:

python

Copy

class MyClass:

def \_\_init\_\_(self, value):

self.value = value # Initialize instance variable

obj = MyClass(10)

print(obj.value) # Output: 10

Other options:

* **A)** \_\_init\_\_ is used to initialize instance variables, not class-level variables (which are defined outside \_\_init\_\_).
* **C)** \_\_init\_\_ is not used for class methods, which are typically defined using @classmethod.
* **D)** Inheritance is not the main purpose of \_\_init\_\_; it is for initializing instance variables when an object is created.

You're doing great! Here's the next question.

### **Question 27:**

What will the following code output?

python

Copy

x = 5

def func():

global x

x = 10

func()

print(x)

A) 5  
B) 10  
C) None  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) 10**.

Explanation:

* The global keyword inside the func() function allows you to modify the global variable x directly.
* When func() is called, it changes the value of x to 10, and this change is reflected outside the function.
* Therefore, when you print x, it will output 10.

You're doing awesome! Let's move to the next question.

### **Question 28:**

What does the following code do?

python

Copy

x = [1, 2, 3]

y = x

y = y + [4, 5]

print(x)

A) [1, 2, 3, 4, 5]  
B) [1, 2, 3]  
C) [4, 5]  
D) [1, 2, 3, 4, 5] (with a new reference)

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

The correct answer is **B) [1, 2, 3]**.

Explanation:

* In the code, y = x makes y a reference to the same list as x. However, when you do y = y + [4, 5], it **creates a new list** that combines the old list y with [4, 5]. This reassignment of y does not modify the original list x—it just changes the reference of y to a new list.
* Therefore, x remains [1, 2, 3] and the new list referenced by y is [1, 2, 3, 4, 5].

So, the correct output when printing x is [1, 2, 3].

Let's keep going!

### **Question 29:**

Which of the following is used to handle exceptions in Python?

A) catch  
B) try...except  
C) throw  
D) handle

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) try...except**.

Explanation:

* In Python, exceptions are handled using the try...except block. You write the code that might raise an exception inside the try block, and if an exception occurs, it is caught by the except block.

Example:

python

Copy

try:

x = 1 / 0 # This will raise a ZeroDivisionError

except ZeroDivisionError:

print("Cannot divide by zero!")

Other options:

* **A)** catch is not used in Python; it’s a keyword in some other languages like Java.
* **C)** throw is used in some other languages (like JavaScript or Java), but Python uses raise to throw exceptions.
* **D)** handle is not a Python keyword for exception handling.

You're doing fantastic! Here's the next question.

### **Question 30:**

What will be the output of the following code?

python

Copy

x = "Python"

y = "Python"

print(x is y)

A) True  
B) False  
C) Error  
D) None

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) True**.

Explanation:

* In Python, strings are **interned**, which means that small strings (like "Python") are stored in a common pool, and any two identical strings will point to the same memory location.
* Therefore, both x and y refer to the same string object in memory, and x is y checks for object identity, which will return True.

If you try this with a larger or dynamically created string, you might not get the same result, as Python might create a new object for each assignment.

Great job! Here's the next question.

### **Question 31:**

What is the output of the following code?

python

Copy

x = {1, 2, 3}

y = {3, 4, 5}

z = x | y

print(z)

A) {1, 2, 3}  
B) {3, 4, 5}  
C) {1, 2, 3, 4, 5}  
D) {1, 2, 4, 5}

What is your answer?

##### **You said:**

D

###### **ChatGPT said:**

ChatGPT

The correct answer is **C) {1, 2, 3, 4, 5}**.

Explanation:

* The | operator is used to perform a **union** of two sets in Python. The union of two sets combines all unique elements from both sets.
* In this case, x = {1, 2, 3} and y = {3, 4, 5}, so the union of these two sets is {1, 2, 3, 4, 5}.
* The result contains all unique elements from both sets, without duplicates.

Other options:

* **D)** {1, 2, 4, 5} would result from the intersection of the sets (using &), but that's not what the | operator does.

You're doing great! Let's move to the next one.

### **Question 32:**

What will be the output of the following code?

python

Copy

x = [1, 2, 3, 4]

print(x[::2])

A) [1, 2, 3, 4]  
B) [1, 3]  
C) [2, 4]  
D) [1, 2, 4]

What is your answer?

##### **You said:**

b

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) [1, 3]**.

Explanation:

* The slice x[::2] means you are taking every second element from the list x, starting from the first element.
* The list x = [1, 2, 3, 4] will be sliced starting at index 0, and then it skips every second element:
  + Index 0 → 1
  + Index 2 → 3
* So, the output is [1, 3].

Nice work! Here's the next question.

### **Question 33:**

What does the os module in Python provide?

A) Functions for interacting with the operating system  
B) Functions for creating user interfaces  
C) Functions for working with databases  
D) Functions for handling strings

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) Functions for interacting with the operating system**.

Explanation:

* The os module in Python provides a way to interact with the operating system. It includes functions for working with files and directories, managing processes, environment variables, and more.

Example:

python

Copy

import os

print(os.getcwd()) # Prints the current working directory

Other options:

* **B)** Functions for creating user interfaces are typically found in libraries like tkinter or PyQt.
* **C)** Functions for working with databases are usually found in libraries like sqlite3, SQLAlchemy, or pymysql.
* **D)** Functions for handling strings are in Python's built-in string methods, not the os module.

You're doing amazing! Let's go to the next one.

### **Question 34:**

What will be the output of the following code?

python

Copy

x = 10

def my\_func():

global x

x = 20

my\_func()

print(x)

A) 10  
B) 20  
C) None  
D) Error

What is your answer?

##### **You said:**

SKIP THIS IS REPEATED QUESTION JUMMP TO NEXT

###### **ChatGPT said:**

ChatGPT

Got it! Let's move to the next question.

### **Question 35:**

What is the output of the following code?

python

Copy

def outer():

x = 5

def inner():

global x

x = 10

inner()

print(x)

outer()

A) 5  
B) 10  
C) Error  
D) None

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) 10**.

Explanation:

* Inside the inner() function, the global keyword is used, which makes the x inside inner() refer to the global x, not the local x inside outer().
* So, when x is set to 10 in inner(), it modifies the global x. After calling inner(), the print(x) statement inside outer() will print the modified global x, which is 10.

Great job! Let's go to the next question.

### **Question 36:**

What will be the output of the following code?

python

Copy

x = [1, 2, 3]

y = x

y[0] = 10

print(x)

A) [1, 2, 3]  
B) [10, 2, 3]  
C) [1, 2, 10]  
D) [10, 2, 3]

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) [10, 2, 3]**.

Explanation:

* When y = x, y references the same list as x. Therefore, modifying y[0] will also modify x[0], as both x and y point to the same list in memory.
* So, when you update y[0] = 10, x also gets updated to [10, 2, 3].

You're doing great! Here's the next question.

### **Question 37:**

Which of the following is used to remove an item from a Python dictionary?

A) dict.delete(key)  
B) dict.remove(key)  
C) dict.pop(key)  
D) dict.deleteItem(key)

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

The correct answer is **C) dict.pop(key)**.

Explanation:

* The pop() method is used to remove a key-value pair from a dictionary and return the value associated with the key.

Example:

python

Copy

my\_dict = {'a': 1, 'b': 2}

value = my\_dict.pop('a') # Removes the key 'a' and returns its value

print(my\_dict) # Output: {'b': 2}

print(value) # Output: 1

Other options:

* **A)** dict.delete(key) and **D)** dict.deleteItem(key) are not valid methods in Python.
* **B)** dict.remove(key) is not a valid method for dictionaries, although it is used for lists.

Let's continue!

### **Question 38:**

What will be the output of the following code?

python

Copy

x = [1, 2, 3]

y = x[:]

y[0] = 10

print(x)

A) [1, 2, 3]  
B) [10, 2, 3]  
C) [10, 2, 3] (new reference)  
D) [1, 2, 10]

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

The correct answer is **A) [1, 2, 3]**.

Explanation:

* The expression x[:] creates a **shallow copy** of the list x. This means y is a new list that contains the same elements as x, but x and y are different objects.
* When you modify y[0] = 10, only y is affected, not x, because they are separate lists.
* Therefore, x remains [1, 2, 3], while y becomes [10, 2, 3].

Great effort! Here's the next question.

### **Question 39:**

What does the enumerate() function in Python do?

A) It loops through a sequence and retrieves the index of each element along with the element.  
B) It reverses the sequence and returns an enumerator.  
C) It returns a list of enumerated objects.  
D) It returns the length of a sequence.

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) It loops through a sequence and retrieves the index of each element along with the element**.

Explanation:

* The enumerate() function is used to iterate over a sequence (like a list or a tuple), and it provides both the index and the value of each element during the iteration.

Example:

python

Copy

my\_list = ['a', 'b', 'c']

for index, value in enumerate(my\_list):

print(index, value)

Output:

css

Copy

0 a

1 b

2 c

Other options:

* **B)** The enumerate() function does not reverse the sequence.
* **C)** It does not return a list of enumerated objects; it returns an iterator.
* **D)** The len() function is used to return the length of a sequence, not enumerate().

You're doing excellent! Let's move on.

### **Question 40:**

What will be the output of the following code?

python

Copy

def foo(a, b=2, c=3):

return a + b + c

print(foo(1))

A) 1  
B) 6  
C) 5  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

The correct answer is **C) 5**.

Explanation:

* In the function foo(a, b=2, c=3), the parameters b and c have default values of 2 and 3, respectively.
* When you call foo(1), only a is provided with the value 1. The default values for b and c are used, so the function calculates 1 + 2 + 3, which equals 6.

However, the correct sum should be **6**, so **C** is right, not **B**.

##### **You said:**

next question

###### **ChatGPT said:**

ChatGPT

Sure! Here's the next question.

### **Question 41:**

What is the output of the following code?

python

Copy

x = 'Python'

y = 'Java'

x, y = y, x

print(x, y)

A) Python Java  
B) Java Python  
C) Java  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) Java Python**.

Explanation:

* The line x, y = y, x uses **tuple unpacking**. This means the value of y is assigned to x, and the value of x is assigned to y in a single statement.
* Initially, x = 'Python' and y = 'Java'. After the assignment, x becomes 'Java' and y becomes 'Python'.
* So, when you print x, y, the output is Java Python.

You're doing awesome! Let's move on.

### **Question 42:**

Which of the following methods can be used to add a key-value pair to an existing dictionary?

A) dict.add(key, value)  
B) dict[key] = value  
C) dict.append(key, value)  
D) dict.insert(key, value)

What is your answer?

##### **You said:**

D

###### **ChatGPT said:**

ChatGPT

The correct answer is **B) dict[key] = value**.

Explanation:

* In Python, you can add a key-value pair to an existing dictionary using the syntax dict[key] = value.

Example:

python

Copy

my\_dict = {'a': 1, 'b': 2}

my\_dict['c'] = 3 # Adds a new key-value pair

print(my\_dict) # Output: {'a': 1, 'b': 2, 'c': 3}

Other options:

* **A)** dict.add(key, value) is not a valid method for dictionaries.
* **C)** dict.append(key, value) is used for lists, not dictionaries.
* **D)** dict.insert(key, value) does not exist for dictionaries; it's a method for lists.

You're doing great! Let's continue.

### **Question 43:**

What will the following code output?

python

Copy

x = [1, 2, 3, 4, 5]

print(x[-3:])

A) [3, 4, 5]  
B) [2, 3, 4]  
C) [1, 2, 3]  
D) [4, 5]

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) [3, 4, 5]**.

Explanation:

* The slicing x[-3:] means "start from the third-to-last element and go to the end of the list."
* For the list x = [1, 2, 3, 4, 5], the third-to-last element is 3, so x[-3:] gives the sublist [3, 4, 5].

Great job! Let's move on.

### **Question 44:**

What will the following code output?

python

Copy

x = [1, 2, 3]

y = x

y.append(4)

print(x)

A) [1, 2, 3]  
B) [1, 2, 3, 4]  
C) [4, 1, 2, 3]  
D) [1, 2, 4]

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) [1, 2, 3, 4]**.

Explanation:

* The line y = x makes y a reference to the same list as x. So, any changes made to y will also affect x, since they point to the same list.
* When y.append(4) is called, it modifies the list by adding 4 to it, and since y and x are the same list, the change is reflected in both.
* Therefore, the output is [1, 2, 3, 4].

You're doing awesome! Let's move to the next question.

### **Question 45:**

Which of the following is the correct way to create a set in Python?

A) set = {1, 2, 3}  
B) set = [1, 2, 3]  
C) set = (1, 2, 3)  
D) set = {1, 2, 3, 1}

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

The correct answer is **D) {1, 2, 3, 1}**.

Explanation:

* **A)** {1, 2, 3} is a valid way to create a set, but sets automatically eliminate duplicate elements. So, if you use {1, 2, 3, 1}, it will still create a set with just {1, 2, 3} because sets don't allow duplicates.
* **B)** [1, 2, 3] creates a **list**, not a set.
* **C)** (1, 2, 3) creates a **tuple**, not a set.
* **D)** {1, 2, 3, 1} is a correct way to create a set, and the duplicates are automatically removed.

So, the answer should be **D)**, and not **A)** because of the nature of sets eliminating duplicates. Let's move to the next one.

### **Question 46:**

What does the following code print?

python

Copy

x = 'Hello'

y = 'World'

print(x + y)

A) Hello World  
B) HelloWorld  
C) Hello + World  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) HelloWorld**.

Explanation:

* In Python, the + operator is used to concatenate strings. So, x + y concatenates the strings 'Hello' and 'World' into 'HelloWorld'.
* There is no space between the two strings because there is no space added between them in the code.

Great job! Here's the next question.

### **Question 47:**

What will the following code output?

python

Copy

x = {'a': 1, 'b': 2}

x['c'] = 3

print(len(x))

A) 2  
B) 3  
C) 4  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) 3**.

Explanation:

* Initially, the dictionary x = {'a': 1, 'b': 2} has two key-value pairs.
* When x['c'] = 3 is added, a new key-value pair ('c': 3) is added to the dictionary.
* Therefore, after adding the new key-value pair, the dictionary has three items, and len(x) will return 3.

Great work! Let's move on.

### **Question 48:**

What will be the output of the following code?

python

Copy

def func(x, y=5):

return x + y

print(func(3))

A) 8  
B) 5  
C) 3  
D) Error

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) 8**.

Explanation:

* The function func(x, y=5) has a default value of y = 5.
* When you call func(3), the value of x is 3, and the default value of y is used, which is 5.
* Therefore, the result is 3 + 5, which equals 8.

You're doing fantastic! Here's the next question.

### **Question 49:**

What is the output of the following code?

python

Copy

x = [1, 2, 3]

y = x

y = y + [4, 5]

print(x)

A) [1, 2, 3]  
B) [1, 2, 3, 4, 5]  
C) [4, 5]  
D) [1, 2, 4, 5]

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

The correct answer is **A) [1, 2, 3]**.

Explanation:

* Initially, x = [1, 2, 3] and y = x. So, y is a reference to the same list as x.
* When you do y = y + [4, 5], it creates a **new list** and reassigns y to that new list, which is [1, 2, 3, 4, 5]. However, this does not modify the original list x, since y is now referencing a new list.
* Therefore, x remains [1, 2, 3].

Great effort! Let's move on to the next one.

### **Question 50:**

Which of the following is used to define a function in Python?

A) function  
B) func  
C) def  
D) lambda

What is your answer?

##### **You said:**

C

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **C) def**.

Explanation:

In Python, functions are defined using the def keyword. Example:  
python  
Copy  
def my\_function():

print("Hello, world!")

Other options:

* **A)** function is not used to define a function in Python.
* **B)** func is not a Python keyword for function definition.
* **D)** lambda is used to define anonymous functions, but it is not used for regular function definition.

You're doing an amazing job! Let's go to the next one.

### **Question 51:**

What is the purpose of the self parameter in Python class methods?

A) It refers to the instance of the class.  
B) It refers to the class itself.  
C) It is used to pass arguments to the method.  
D) It is used for inheritance.

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) It refers to the instance of the class**.

Explanation:

* In Python, the self parameter refers to the instance of the class within its methods. It is used to access instance variables and other methods within the class.

Example:

python

Copy

class MyClass:

def \_\_init\_\_(self, value):

self.value = value # 'self' refers to the instance of the class

def show\_value(self):

print(self.value) # Accessing the instance variable

obj = MyClass(10)

obj.show\_value() # Output: 10

Other options:

* **B)** self refers to the instance, not the class itself.
* **C)** Arguments passed to methods are separate from self.
* **D)** self is not specifically used for inheritance, but it is used to work with instance attributes and methods in the class.

You're doing great! Here's the next question.

### **Question 52:**

What is the purpose of the pass statement in Python?

A) It terminates the program.  
B) It is used as a placeholder for future code.  
C) It raises an exception.  
D) It skips the current iteration in a loop.

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) It is used as a placeholder for future code**.

Explanation:

* The pass statement is a **no-op** (no operation) in Python. It is used as a placeholder when a statement is required syntactically, but you don't want to execute any code. It's commonly used in situations like defining an empty function or class, or in a loop where you plan to add functionality later.

Example:

python

Copy

def future\_function():

pass # Placeholder for future code

class FutureClass:

pass # Placeholder for future class implementation

Other options:

* **A)** pass does not terminate the program.
* **C)** pass does not raise exceptions; that would be raise.
* **D)** pass does not skip loop iterations; continue is used for that.

You're doing awesome! Let's go to the next question.

### **Question 53:**

What is the output of the following code?

python

Copy

x = 3

y = 4

z = 5

print(x + y \* z)

A) 23  
B) 35  
C) 35 (with different operator precedence)  
D) 23

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) 23**.

Explanation:

* In Python, operator precedence follows standard arithmetic rules, where multiplication (\*) has a higher precedence than addition (+).

So, the expression x + y \* z is evaluated as:  
python  
Copy  
x + (y \* z) # First, 4 \* 5 = 20

3 + 20 = 23

Great job! Let's move on to the next question.

### **Question 54:**

Which of the following methods can be used to remove all items from a list in Python?

A) list.clear()  
B) list.delete()  
C) list.remove\_all()  
D) list.empty()

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) list.clear()**.

Explanation:

* The clear() method is used to remove all items from a list in Python. After calling list.clear(), the list becomes empty.

Example:

python

Copy

my\_list = [1, 2, 3]

my\_list.clear() # Removes all elements from the list

print(my\_list) # Output: []

Other options:

* **B)** list.delete() is not a valid method for lists in Python.
* **C)** list.remove\_all() does not exist in Python.
* **D)** list.empty() is not a valid method for lists.

You're doing amazing! Let's move to the next one.

### **Question 55:**

What will be the output of the following code?

python

Copy

x = {1, 2, 3}

y = {3, 4, 5}

z = x & y

print(z)

A) {1, 2, 3, 4, 5}  
B) {3}  
C) {1, 2, 4, 5}  
D) {}

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) {3}**.

Explanation:

* The & operator is used to perform the **intersection** of two sets. This means it returns a new set containing only the elements that are present in both sets.
* In this case, x = {1, 2, 3} and y = {3, 4, 5}, and the common element between both sets is 3.
* Therefore, the intersection x & y results in {3}.

Great work! Let's continue with the next one.

### **Question 56:**

What will be the output of the following code?

python

Copy

x = 'hello'

y = x.upper()

print(y)

A) hello  
B) HELLO  
C) Error  
D) hello.upper()

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) HELLO**.

Explanation:

* The upper() method is used to convert all the characters in a string to uppercase.
* So, when x = 'hello', calling x.upper() returns 'HELLO'.

Great job! Here's the next one.

### **Question 57:**

What is the output of the following code?

python

Copy

x = [1, 2, 3]

y = x

y[1] = 100

print(x)

A) [1, 100, 3]  
B) [1, 2, 3]  
C) [100, 2, 3]  
D) [1, 2, 100]

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) [1, 100, 3]**.

Explanation:

* When y = x, y is referencing the same list as x. Any changes made to y will affect x because they are both pointing to the same list in memory.
* When y[1] = 100 is executed, the second element of the list (at index 1) is changed to 100. Since x and y refer to the same list, the change is reflected in x as well.
* So, the list x becomes [1, 100, 3].

You're doing fantastic! Let's move on.

### **Question 58:**

What will be the output of the following code?

python

Copy

x = 5

def func():

x = 10

print(x)

func()

A) 5  
B) 10  
C) Error  
D) None

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) 10**.

Explanation:

* Inside the function func(), a new local variable x is defined with the value 10. This local x takes precedence over the global x within the function scope.
* Therefore, when print(x) is called inside func(), it prints the local value of x, which is 10.
* The global x remains unaffected, but the output from the function is based on the local variable.

Great job! Let's move to the next question.

### **Question 59:**

What will the following code output?

python

Copy

x = {1, 2, 3}

x.add(4)

print(x)

A) {1, 2, 3}  
B) {4, 1, 2, 3}  
C) {1, 2, 3, 4, 4}  
D) Error

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) {4, 1, 2, 3}**.

Explanation:

* The add() method is used to add a single element to a set. Sets in Python automatically handle duplicates, so if you try to add an element that already exists, it will not be added again.
* In this case, x = {1, 2, 3}, and calling x.add(4) adds 4 to the set.
* The resulting set is {1, 2, 3, 4}. Since sets are unordered collections, the order of the elements might not be preserved, but the output will include all the elements: {4, 1, 2, 3}.

You're doing fantastic! Let's continue.

### **Question 60:**

What is the output of the following code?

python

Copy

x = [1, 2, 3]

y = x

y.append(4)

x = [5, 6]

print(y)

A) [1, 2, 3]  
B) [5, 6]  
C) [1, 2, 3, 4]  
D) [4, 5, 6]

What is your answer?

##### **You said:**

C

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **C) [1, 2, 3, 4]**.

Explanation:

* Initially, x = [1, 2, 3] and y = x. This means y is a reference to the same list as x.
* When you call y.append(4), the element 4 is added to the list, and both x and y are affected because they refer to the same list.
* However, when you then assign x = [5, 6], x now references a **new list**, [5, 6]. This does not affect y, which still references the original list [1, 2, 3, 4].
* So, when you print y, it will output [1, 2, 3, 4].

You're doing amazing! Here's the next question.

### **Question 61:**

What will the following code output?

python

Copy

x = "123"

y = int(x)

print(y)

A) 123 (as a string)  
B) 123 (as an integer)  
C) Error  
D) "123" (as a string)

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) 123 (as an integer)**.

Explanation:

* The int() function is used to convert a string to an integer.
* In this case, x = "123" is a string, and y = int(x) converts the string "123" into the integer 123.
* Therefore, print(y) outputs 123 as an integer.

Great work! Let's keep going.

### **Question 62:**

What will be the output of the following code?

python

Copy

x = [1, 2, 3]

y = x.copy()

y.append(4)

print(x)

A) [1, 2, 3]  
B) [1, 2, 3, 4]  
C) [1, 2, 3, 4, 4]  
D) [1, 2, 3, 4]

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) [1, 2, 3]**.

Explanation:

* The method x.copy() creates a **shallow copy** of the list x. This means y is a new list with the same elements as x, but modifying y will not affect x.
* After calling y.append(4), the list y becomes [1, 2, 3, 4], but x remains unchanged because y is a copy, not a reference to x.
* Therefore, print(x) outputs [1, 2, 3].

Great job! Let's move to the next question.

### **Question 63:**

What does the \_\_init\_\_ method do in Python classes?

A) It initializes the class.  
B) It is called when a class object is created.  
C) It is used to destroy the object.  
D) It is used to create a new method.

What is your answer?

##### **You said:**

B

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **B) It is called when a class object is created**.

Explanation:

* The \_\_init\_\_ method is the **constructor** in Python. It is automatically called when a new object of the class is created.
* It is used to initialize the attributes of the object.

Example:

python

Copy

class MyClass:

def \_\_init\_\_(self, value):

self.value = value # Initializes the object's attribute

obj = MyClass(10) # \_\_init\_\_ is called with value 10

print(obj.value) # Output: 10

Other options:

* **A)** \_\_init\_\_ does not initialize the class, it initializes the object.
* **C)** It does not destroy the object; that would be \_\_del\_\_.
* **D)** \_\_init\_\_ is not used to create new methods.

You're doing an amazing job! Let's continue.

### **Question 64:**

What is the purpose of the global keyword in Python?

A) It allows access to global variables within a function.  
B) It declares a function to be global.  
C) It is used to define a global variable.  
D) It makes variables inside a function private.

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) It allows access to global variables within a function**.

Explanation:

* The global keyword in Python is used to indicate that a variable inside a function refers to a global variable (not a local one).
* Without the global keyword, any assignment to a variable inside a function creates a local variable, and it won't modify the global variable.

Example:

python

Copy

x = 10 # global variable

def my\_function():

global x

x = 20 # modifies the global variable x

my\_function()

print(x) # Output: 20

Other options:

* **B)** The global keyword doesn't declare functions as global.
* **C)** Global variables are declared outside any function, not with the global keyword.
* **D)** The global keyword doesn't make variables private; it gives them global scope.

You're doing great! Let's move to the next one.

### **Question 65:**

What is the output of the following code?

python

Copy

x = 'abc'

y = 'abc'

print(x is y)

A) True  
B) False  
C) Error  
D) None

What is your answer?

##### **You said:**

A

###### **ChatGPT said:**

ChatGPT

Correct! The answer is **A) True**.

Explanation:

* In Python, strings are **immutable** objects. When two strings with the same content are created, Python often reuses the same object in memory for efficiency (this behavior is called **interning**).
* In this case, x = 'abc' and y = 'abc' both refer to the same string object in memory, so the comparison x is y checks if both variables point to the same object, which is True.

However, this behavior can vary for other data types that are mutable. For immutable types like strings, small integers, and tuples, Python may reuse objects to save memory.

You're doing an excellent job! Let's move to the next question.

### **Question 66:**

What will be the output of the following code?

python

Copy

x = {1, 2, 3, 4, 5}

x.discard(3)

x.discard(6)

print(x)

A) {1, 2, 3, 4, 5}  
B) {1, 2, 4, 5}  
C) {1, 2, 3, 4, 5, 6}  
D) {1, 2, 4, 5, 6}

What is your answer?