Theory Questions - Python Programming: Possible questions

Prepared By: Ali Al Housseini – S287634

1

Explain the concept of alias in Python and the potential problems related to it during code writing.

In Python, an alias is a second name that refers to the same object as another name. This can make the code more difficult to read and understand and lead to unexpected behavior and bugs in the code. When working with mutable objects, it's important to use the copy method or the "deepcopy" method of the copy module instead of the assignment operator to create a new object with the same data.

2

Why are dictionaries called that in Python?

Dictionaries in Python are called so because they are a data structure that allows you to map keys to values, similar to a real-world dictionary which allows you to look up words and their meanings. Each key in a Python dictionary is mapped to a specific value, and you can use the keys to access the corresponding values. The key-value mapping in a Python dictionary is similar to the way words and their meanings are related in a real-world dictionary, which is why the data structure is called a "dictionary" in Python.

3

What is the main purpose of indentation in the Python language? Give an example where it is of vital importance.

The main purpose of indentation in the Python language is to indicate the structure and organization of the code. In Python, indentation is used to indicate blocks of code that belong together, such as the block of code inside a loop or the block of code inside a function or class.

Indentation is of vital importance in Python because it is used to indicate the scope of variables and control flow structures, such as loops and conditional statements. For example, in a for loop, the code inside the loop must be indented to indicate that it should be executed multiple times. If the code inside the loop is not indented correctly, the loop will not work as intended.

Here is an example:

```
for i in range(10):
    print(i)
```

In this example, the print(i) statement is indented, indicating that it should be executed during each iteration of the loop. If the print(i) statement were not indented, it would only be executed once, outside of the loop.

In summary, indentation is an important aspect of Python programming because it is used to indicate the structure and organization of the code and it is used to indicate the scope of variables and control flow structures.

4

Provide an example where the order of elif statements affects the final result.

In Python, the elif (short for "else if") statement is used to check multiple conditions in a single if-elif-else block. The order in which these conditions are checked can affect the final result of the code.

Here is an example:

```
x = 5
if x > 10:
    print("x is greater than 10")
elif x > 0:
    print("x is greater than 0")
else:
```

```
    print("x is less than or equal to 0")
```

In this example, if the elif x > 0 statement is placed before the elif x > 10 statement, then the final output will be "x is greater than 0" because the first elif statement that is true will be executed and the rest of the statements will be ignored. But if the order was elif x > 10 before elif x > 0 the output will be "x is greater than 10"

Here is another example:

```
x = 15

if x > 10:

    print("x is greater than 10")

elif x > 20:

    print("x is greater than 20")

else:

    print("x is less than or equal to 10")
```

In this example, if the elif x > 20 statement is placed before the elif x > 10, the final output will be "x is less than or equal to 10" because x is not greater than 20, however, if the order is elif x > 10 before elif x > 20 the output will be "x is greater than 10"

In conclusion, the order of elif statements can affect the final result of the code because the first elif statement that is true will be executed and the rest of the statements will be ignored. so it's important to pay attention to the order of the conditions when using elif statements.

5

A programmer wants to represent matrices of numbers in their application. For example, they want to represent the matrix

| **1 2** |

| **3 4** |

The programmer has thought of 3 possible representations of this matrix. Discuss briefly the advantages and disadvantages of the 3 options, indicating if any of them should be considered incorrect.

- `[ 1, 2, 3, 4 ]`
- `[[1,2], [3,4]]`
- `[[1,3], [2,4]]`

There are a few different ways a programmer might choose to represent a matrix of numbers in their application, but here are a few possibilities and the advantages and disadvantages of each:

[1, 2, 3, 4]

Advantage: Simple and easy to implement, doesn't take up much memory.

Disadvantage: Not very readable, it is not clear that this is a matrix and it's hard to access a specific element in it.

[[1,2], [3,4]]

Advantage: It is easy to access a specific element in the matrix by using the indices, it's more readable than the first representation.

Disadvantage: It takes up more memory than the first representation.

[[1,3], [2,4]]

Advantage: It's easy to access a specific element in the matrix by using the indices

Disadvantage: It's not the correct representation of the matrix, the values are not in their correct positions.

In conclusion, The first representation is not recommended as it is not very readable, the second representation is a good option if memory usage is not a concern, and the third representation is incorrect.

6

Explain the concept of variable visibility and indicate the difference between local and global variables.

In programming, variable visibility refers to the scope of a variable, or the part of the code where a variable can be accessed. In Python, there are two types of variable visibility: local and global.

A local variable is a variable that is defined within a function or block of code, and it can only be accessed within that function or block of code. Local variables are created when a function is called and they are destroyed when the function returns. For example:

```
def my_function():
    x = 10 # x is a local variable
    print(x)

my_function()
print(x) # raises an exception
```

In this example, the variable x is a local variable that is defined within the my_function function. It can only be accessed within the function and trying to access it outside the function will raise an exception.

On the other hand, a global variable is a variable that is defined outside of any function or block of code, and it can be accessed from anywhere in the code. Global variables are created when the program starts and they are destroyed when the program ends. For example:

```
x = 10 # x is a global variable

def my_function():
    print(x)

my_function()
```

```
print(x)
```

In this example, the variable x is a global variable that is defined outside the function and it can be accessed from anywhere in the code, both inside and outside the function.

It's important to note that if you want to modify the value of a global variable inside a function you have to use the global keyword to specify that you are referring to the global variable and not creating a new local variable with the same name:

```
x = 10

def my_function():
    global x
    x = x + 1
    print(x)

my_function()
print(x)
```

In summary, the main difference between local and global variables is their scope, a local variable can only be accessed within the function or block of code it is defined in, while a global variable can be accessed from anywhere in the code.

7

What advantages and disadvantages does a set provide compared to a list?

In Python, a set is a collection of unique elements, while a list is an ordered collection of elements. Both sets and lists have their own advantages and disadvantages.

Advantages of sets:

Sets are unordered collections of unique elements, which means that they don't have duplicate values. This makes them particularly useful for removing duplicates from a collection of items.

Sets have a very fast lookup time for membership testing (i.e., checking if an element is in the set or not) due to their use of hash tables. This makes them more efficient than lists for certain operations, such as removing duplicates or checking for the existence of an element.

Sets are also useful for performing set operations such as union, intersection, and difference.

Disadvantages of sets:

Sets are unordered, which means that elements have no index and cannot be accessed by indexing.

Because sets use a hash table to store elements, they can take up more memory than lists.

Advantages of lists:

Lists are ordered collections of elements, which means that elements have an index and can be accessed by indexing.

Lists can store duplicate values

Lists are useful for performing list operations such as indexing, slicing, and iteration.

Disadvantages of lists:

Lists are ordered collections, which means that they are less efficient than sets for certain operations, such as removing duplicates or checking for the existence of an element.

In conclusion, sets and lists have their own advantages and disadvantages. Sets are useful for removing duplicates and set operations, while lists are useful for ordered collections and list operations. Depending on the requirements of the

specific use case, a programmer may choose to use one data structure over the other.

8

Consider Lists and Sets:

list the differences between these two data structures

highlight their advantages and disadvantages

provide an example of data for which representation is more appropriate using a list and data for which representation is more appropriate using a set.

Lists and sets are both data structures in Python, but they have several differences:

Lists are ordered collections of elements, while sets are unordered collections of unique elements. This means that elements in a list have an index and can be accessed by indexing, while elements in a set have no index and cannot be accessed by indexing.

Lists can store duplicate values, while sets do not allow duplicate values. This means that if you insert the same element multiple times in a set, it will only be stored once.

Lists support slicing, indexing, and iteration operations, while sets support set operations such as union, intersection, and difference.

Advantages of Lists:

Lists are ordered collections of elements, which means that they are useful for storing data that needs to be in a specific order.

Lists can store duplicate values which can be useful in some scenarios.

Lists are useful for performing list operations such as indexing, slicing, and iteration

Disadvantages of Lists:

Lists are less efficient than sets for certain operations such as removing duplicates or checking for the existence of an element.

Advantages of Sets:

Sets are unordered collections of unique elements, which means they are useful for removing duplicates from a collection of items.

Sets have a very fast lookup time for membership testing (i.e., checking if an element is in the set or not) due to their use of hash tables.

Sets are also useful for performing set operations such as union, intersection, and difference.

Disadvantages of Sets:

Sets are unordered, which means that elements have no index and cannot be accessed by indexing.

Because sets use a hash table to store elements, they can take up more memory than lists.

An example of data for which representation is more appropriate using a list is a list of students in a class, where the order of the students matters, such as sorted by alphabetical order. An example of data for which representation is more appropriate using a set is a list of unique words in a document, where the order of the words is not important but the number of occurrences are.

9

Which of the following statements allows you to create a variable d2 that contains a copy of the dictionary d1?

d2 = d1

d2 = dict(d1.values())

d2 = dict(d1.keys())

d2 = dict(d1)

The statement that allows you to create a variable d2 that contains a copy of the dictionary d1 is:

d2 = dict(d1)

This creates a new dictionary object and copies the key-value pairs from d1 into d2.

d2 = d1 creates an alias, it creates a reference to the same dictionary object.

d2 = dict(d1.values()) creates a new dictionary object with the values of d1 as its keys and with None as values.

d2 = dict(d1.keys()) creates a new dictionary object with the keys of d1 as its keys and with None as values.

It's important to note that if the original dictionary contains mutable objects such as lists or dictionaries, changes made to the copy will also be reflected in the original and vice versa. To create a deep copy of the dictionary, you can use the copy module's deepcopy method.

10

The Python language provides dictionaries (dict) and sets (set) for programmers. Provide a brief description of these two complex data structures, indicating at least one common feature.

The Python language provides two complex data structures for programmers, dictionaries (dict) and sets (set).

Dictionaries, also known as associative arrays or hash maps, are a collection of key-value pairs. Each key is unique, and it is associated with a value. Dictionaries are used to store data in a structured way, and they are optimized for fast lookups. They can be created using curly braces {} or the built-in dict() function. For example:

Copy code

my_dict = {"name": "John Smith", "age": 30}

Sets, on the other hand, are a collection of unique elements. Sets are used to store unique values and perform set operations such as union, intersection and difference. Sets are implemented as hash tables and are optimized for fast membership testing. They can be created using the built-in set() function or by using curly braces {} with elements separated by commas. For example:

Copy code

my_set = {1, 2, 3, 4}

A common feature between both data structures is that they are implemented as hash tables. This means that they both have fast lookup times for checking the existence of an element, and also both data structures do not allow duplicate values.

11

Which of these structures can not be implemented in Python and why?

dictionary of lists

set of dictionaries

dictionary of sets

list of lists.

All of the structures listed can be implemented in Python.

A dictionary of lists can be implemented by creating a dictionary with keys and values that are lists. For example:

my_dict = {"list1": [1, 2, 3], "list2": [4, 5, 6]}

A set of dictionaries can be implemented by creating a set with elements that are dictionaries. For example:

my_set = {{"name": "John Smith", "age": 30}, {"name": "Jane Doe", "age": 25}}

A dictionary of sets can be implemented by creating a dictionary with keys and values that are sets. For example:

my_dict = {"set1": {1, 2, 3}, "set2": {4, 5, 6}}

A list of lists can be implemented by creating a list with elements that are lists. For example:

my_list = [[1, 2, 3], [4, 5, 6]]

All the above structures can be implemented in Python, none of them are impossible to implement.

12

Suppose you read data from the keyboard. What happens if you use this data directly in an arithmetic operation? How should you proceed to handle numerical data entered from the keyboard?

If you read data from the keyboard and use it directly in an arithmetic operation, Python will raise a TypeError if the data is not a numerical data type (int or float) because only numerical data types can be used in arithmetic operations.

For example, if you read a string from the keyboard and try to use it in an arithmetic operation, Python will raise a TypeError:

```
x = input("Enter a number: ")
y = x + 2  # raises TypeError
```

To handle numerical data entered from the keyboard, you should first convert the input data to a numerical data type (int or float) before performing arithmetic operations. You can use the int() or float() functions to convert the input data to the desired data type.

For example:

```
x = input("Enter a number: ")
x = int(x)  # convert input data to int
y = x + 2  # valid arithmetic operation
```

It's also important to validate the input and check that the input is a number, otherwise you may get a ValueError exception. You can use the try-except block to handle this kind of errors.

```
try:
    x = int(input("Enter a number: "))
    y = x + 2
except ValueError:
    print("Invalid input, please enter a number.")
```

In this way, if the input is not a number, it will raise a ValueError and the program will catch it, print a message and continue execution.

13

Is it possible that in a dictionary there are two keys associated with equal values? And two values associated with the same key?

It is not possible to have two keys in a dictionary that are associated with the same value. Dictionaries in Python are implemented as hash tables, and the keys must be unique. If you try to insert a key-value pair into a dictionary where the key already exists, the value associated with that key will be overwritten with the new value.

However, it is possible to have two values in a dictionary associated with the same key. The values in a dictionary can be any type of data, including lists, sets, and even other dictionaries. When two values associated with the same key, the last value will be the one associated with the key, this is because the keys are unique.

For example:

```
my_dict = {'key1': 'value1', 'key1': 'value2'}
print(my_dict)
```

The output of the above code will be:

```
{'key1': 'value2'}
```

As we can see, the key 'key1' has been overwritten with the value 'value2', instead of having two keys associated with different values, the dictionary has only one key associated with the last value 'value2'

14

In a Python function, what is the difference between arguments and return values? How many arguments can a function invocation have? And how many values can a function return?

In a Python function, the difference between arguments and return values is as follows:

Arguments: These are the values that are passed to a function when it is called. Functions can have one or more arguments, and they are used to provide input to the function. The function uses these arguments to perform a specific task and produce a result.

Return values: These are the values that a function produces after it has been executed. A function can return one or more values. The return statement is used to specify the value(s) that a function should return.

A function can take any number of arguments, there is no limit on the number of arguments that a function can take. A function can take no arguments at all or any number of arguments.

A function can return any number of values, there is no limit on the number of return values that a function can have. A function can return no values at all or any number of values. Python allows a function to return multiple values by returning a tuple or a list, also, you can use the yield statement in a function to return multiple values.

15

How can you convert a string to lowercase in Python?

You can convert a string to lowercase in Python by using the lower() method. The lower() method returns a new string with all the characters in lowercase. This method does not modify the original string.

For example:

```
original_string = "HELLO WORLD"

lowercase_string = original_string.lower()

print(lowercase_string)
```

Output:

```
hello world
```