Python Interview Questions and Solutions

1. Scenario: You have a list of integers, and you need to find the maximum value in the list. a. Question: How would you find the maximum value from the following list?

To find the maximum value in the list, I will use the inbuilt max() function and another way to get the maximum value is by using the loop to compare the value manually from the list. Both example is given below.

In [1]:
```
#  using the in-built max() function to find the maximum value from the list
numbers = [1, 30, 50, 20, 100, 200]
max_value = max(numbers)
print(max_value)
```

```
200
```

In [2]:
```
# Use the if else statement to find the maximum value from the list
num = [1, 30, 50, 20, 100, 200]
max_value = num[0] # First assuming that the first element in the is the largest
for number in num:
    if number > max_value:
        max_value = number
print("Maximum number from the list is: ", max_value)
```

```
Maximum number from the list is:  200
```

2. Scenario: You need to concatenate a list of strings into a single string, with each string separated by a comma. a. Question: How would you combine the following list of strings into a single string?

In string there is a method called join() which helps to joins all the strings from the list. I will use join() methods to join strings from the list. Example is given below:

In [3]:
```
#Example of concatenating a list of strings into a single string using join methods
listOfString = ["Sam", "Paul", "Jordan", "Jonny", "Chris"]
concatenateString = ", ".join(listOfString)
print(concatenateString)
```

```
Sam, Paul, Jordan, Jonny, Chris
```

3. Scenario: You have a dictionary where the keys are names and the values are ages. You need to retrieve the age of a specific person. a. Question: How would you get the age of "Dev" from the following dictionary?

To retrive the age of a specific person we can use their name as a key. To get the age of Dev from the dictionary we can retrive age using Dev as a key. Example is given below.

In [4]:
```
#Dictionary where keys are names and ages are values
ages = {"Sam": 14, "Dev": 20, "Jordan": 25, "Paul": 18,"Chris": 30}
personAge = ages["Dev"] #retriving Dev age passing Dev as key
print("Dev age is: ", personAge)
```

```
Dev age is:  20
```

4. Scenario: You need to remove duplicate elements from a list while preserving the original order of elements. a. Question: How would you remove duplicates from the following list?

To remove duplicate elements from a list while preserving the original order of elements we can use a set to track the elements because duplicate is not allowed in the set and loop to iterate those lists of sets. An example is given below:

In [5]:
```
numberOfList = [1,1,2,2,3,3,3,4,5,5,6,7,8,9,9,9,9,9,10,10,1,6,5]

#Removing duplicates while preserving the original order of element
seenNumbers = set()
newNumberOfList = []
for number in numberOfList:
    if number not in seenNumbers:
        newNumberOfList.append(number)
        seenNumbers.add(number)
print("New number on the List: ", newNumberOfList)
```

```
New number on the List:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

5. Scenario: You have a NumPy array and need to compute the sum of all its elements. a. Question: How would you find the sum of elements in the following NumPy array?

In numPy library we can find the sum() function which we can use to find the sum of numpy array. Example is given nelow

In [6]:
```
import numpy as np
#Numpy array
numpyArray = np.array([10, 30, 40, 50,  80])
totalSum = np.sum(numpyArray)
print("Total Sum: ", totalSum)
```

```
Total Sum:  210
```

6. Scenario: You have a 2D NumPy array and need to get the maximum value from each column. a. Question: How would you find the maximum values for each column in the following array?

In NumPy we can use the numpy.max() function with axis parameter to find the maximum value from each column. Example is given below:

In [7]:
```
import numpy as np
#2D Numpy Array
numpyArray = np.array([[1,2,3], [3,4,5], [5,5,5]])
maximumValue = np.max(numpyArray, axis=0)
print("The maximum values for each column in the array is: ", maximumValue)
```

```
The maximum values for each column in the array is:  [5 5 5]
```

7. Scenario: You have a DataFrame and need to sort it by a specific column. a. Question: How would you sort the DataFrame by the "Age" column?

To sort a DataFrame by a specific column we can use sortValues() function that is provided by pandas library. For example:

In [14]:
```
import pandas as pd
#DataFrame
data = {"Name": ["Sam", "Paul", "Jordan", "Jonny", "Chris"],
        "Age" : [14, 25, 18, 30, 20],
        "Country": ["Denmark", "Australia", "India", "USA", "England"]}
dataFrame = pd.DataFrame(data)
sortedDataframe = dataFrame.sort_values(by="Age")
print(sortedDataframe)
```

```
    Name  Age    Country
0    Sam   14    Denmark
2  Jordan   18      India
4   Chris   20    England
1    Paul   25  Australia
3   Jonny   30        USA
```

8. Scenario: You need to calculate the total sum of a column in a DataFrame. a. Question: How would you find the sum of the "Sales" column?

In pandas library it provides the sum() function which we can use to find the sum of the sale column. For example:

In [16]:
```
import pandas as pd
#Sales Dataframe
data = {
        "Product": ["Electronics", "Groceries", "Clothings", "Stationery"],
        "Sales" : [20, 40, 80, 100]
}
dataFrame = pd.DataFrame(data)
#calculating the total sum of the sales column
totalSum = dataFrame["Sales"].sum()
print("Total Sum: ", totalSum)
```

```
Total Sum:  240
```

9. Scenario: You need to check if a variable is of type int. a. Question: How would you verify if a variable x is an integer?

We can simly use the type() function to check the type of the variable. For example:

In [10]:
```
x = 20
#checking the type of the variable
print(type(x))
```

```
<class 'float'>
```

10. Scenario: You have a list of mixed types and need to create separate lists for integers, floats, and strings. a. Question: How would you categorize the following mixed list into different types?

To categorize the mixed list into different types we can iterate through the list and check the type of each elements using type() function. Example is given below:

In [11]:
```
mixedList = [1,2,3.5, 4.1, 6, 8, "Jordan", 10, "Paul", 9.8, "Chris"]
#initilize the empty list for each type
integers = []
floats = []
strings = []
#iterate the each elements and categorize them using type function
for item in mixedList:
    if type(item) is int:
        integers.append(item)
    elif type(item) is float:
        floats.append(item)
    elif type(item) is str:
        strings.append(item)
print("Integer: ", integers)
print("floats: ", floats)
print("strings: ", strings)
```

```
Integer:  [1, 2, 6, 8, 10]
floats:  [3.5, 4.1, 9.8]
strings:  ['Jordan', 'Paul', 'Chris']
```

11. Scenario: You are given a string of comma-separated numbers. You need to convert this string into a list of integers. a. Question: How would you achieve this conversion?

We can use the split() method in the string to separate the numbers into a list of substrings using commas and After splitting them we can convert each substring into an integer. An example is given below:

In [13]:
```
strOfNum = "1,2,3,5,8,30,10,209"
#Splitting strings using comma into a list of substring
stringList = strOfNum.split(",")
print (stringList) #printing string list

#Converting each substring to integer
integerList = [int(num) for num in stringList]
print(integerList)
```

```
['1', '2', '3', '5', '8', '30', '10', '209']
[1, 2, 3, 5, 8, 30, 10, 209]
```

12. Scenario: You need to reverse the words in a sentence while keeping the order of the words intact. a. Question: How would you reverse each word in the following sentence?

To reverse each word in the sentence by keeping the order of the words first, we need to split the sentence using the split() method in the string, then reverse them using slicing, and Lastly join all the split strings together in a single string using join() function. An example is given below:

In [18]:
```
sentence = " Hello! I am Sabin"
#splitting the string
splitString = sentence.split()
#reverse each word from the split string
reverseString = [string[::-1] for string in splitString] #iterate over each words in the list and reverse it by slicing
#joining the split string together in one string
joinString = " ".join(reverseString)
print(joinString)
```

```
!olleH I ma nibaS
```

13.Scenario: You need to combine two lists into a dictionary where the elements of the first list are keys and the elements of the second list are values. a. Question: How would you combine the following lists into a dictionary?

We can use the zip() function along with dict() to combine the lists into a dictionary. zip() function helps to pair elements from two lists. For example

In [19]:
```
# Two diff lists
keys = ['name', 'age', 'city']
values = ['Jordan', 20, 'Saint Paul']

# Combine the lists into a dictionary
combinedDict = dict(zip(keys, values))

print("Combined Dictionary:", combinedDict)
```

```
Combined Dictionary: {'name': 'Jordan', 'age': 20, 'city': 'Saint Paul'}
```

14. Scenario: You are given a list of tuples where each tuple contains a name and an age. You need to sort this list by age. a. Question: How would you sort the following list of tuples by age?

We can use itemgetter opoerater and sorted function to sort the list of tuple by age. Item getter is the function that creates a callable that retrives the element at index of each tuple.

In [20]:
```
from operator import itemgetter

# list of tuples
tuple = [('Jordan', 25), ('Chris', 18), ('Paul', 30), ('David', 42)]

# Sort the list of tuples by age using itemgetter
sortedTuple = sorted(tuple, key=itemgetter(1))

print("List sorted by age:", sortedTuple)
```

```
List sorted by age: [('Chris', 18), ('Jordan', 25), ('Paul', 30), ('David', 42)]
```

15. Scenario: You have a 2D NumPy array and need to calculate the mean value of each row. a. Question: How would you compute the mean value for each row in the following array?

To compute the mean value for each row in the given array, we can use numpy.mean() function with the axis parameter set to 1. Axis parameter specify that which axis to operate along from the 2D array and keeping axis '1' indicates thar the operation should be performed accross columns. For example:

In [22]:
```
import numpy as np

#  2D NumPy array
numpyArray = np.array([
    [1, 2, 5],
    [5, 6, 8],
    [11, 12, 20]
])

# Compute the mean value for each row
rowMeans = np.mean(numpyArray, axis=1)

print("Mean value of each row:", rowMeans)
```

```
Mean value of each row: [ 2.66666667  6.33333333 14.33333333]
```

16. Scenario: You need to create a NumPy array of size 10 filled with zeros, and then set the fifth element to 5. a. Question: How would you create and modify the array as described?

We can use numpy.zeros() to create the array with zeros and assig the value at 4th index since in array it starts with 0 index. For example:

In [24]:
```
import numpy as np

#Creating a NumPy array of size 10 filled with zeros
array = np.zeros(10)
print(array) #printing the array with 10 zeros

#assigning the fifth element to 5
array[4] = 5

print("Modified array:", array)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Modified array: [0. 0. 0. 0. 5. 0. 0. 0. 0. 0.]
```

17. Scenario: You have a DataFrame and need to find the median of a numeric column. a. Question: How would you find the median of the "Scores" column in the following DataFrame?

We can use median() method provided by pandas to find the median of the scores column from the Dataframe. For example:

In [25]:
```
import pandas as pd
# Dataframe
data = {
        "Name": ["Jordan", "Paul", "Chris", "Dev"],
        "Scores" : [20, 40, 80, 100]
}
dataFrame = pd.DataFrame(data)
#calculating the median of the scores column
medianScore = dataFrame["Scores"].median()
print("Median Score: ", medianScore)
```

```
Median Score:  60.0
```

18. Scenario: You need to filter rows where the value in the "Sales" column is above 100 and then sort these rows by "Date" in descending order. a. Question: How would you filter and sort the DataFrame based on the given criteria?

We can use filtering and sorting methods provide by pandas to filter and sort the dataframe based on given criteria. An example is given below:

In [32]:
```
import pandas as pd

# DataFrame
data = {
    "Date": ["2024-08-01", "2024-08-02", "2024-08-03", "2024-08-04"],
    "Sales": [10, 200, 85, 110]
}

dataFrame = pd.DataFrame(data)

# Convert the Date column to DateTime type for accurate sorting
dataFrame['Date'] = pd.to_datetime(dataFrame['Date'])

#Filter rows where 'Sales' > 100
filteredDataframe = dataFrame[dataFrame['Sales'] > 100]

# Sort the filtered rows by 'Date' in descending order
sorted_filtered_df = filteredDataframe.sort_values(by='Date', ascending=False)

print("Filtered and sorted DataFrame: ")
print(filteredDataframe)
```

```
Filtered and sorted DataFrame:
        Date  Sales
1 2024-08-02    200
3 2024-08-04    110
```

19. Scenario: You have a string containing a number and need to convert it to an integer. Additionally, handle the case where the string might not represent a valid integer. a. Question: How would you safely convert the string "42" to an integer and handle invalid cases?

In this case we can use try-exxept block, this approach allows to attempt the conversion and catch any erriss if the strings is not valid integer. For example:

In [36]:
```
def convertToInt(x):
    try:
        # Attempt to convert the string to an integer
        return int(x)
    except ValueError:
        # Handle the case where the string is not a valid integer
        print(f"Error: The string '{x}' is not a valid integer.")
        return None

# Example strings
validString = "20"
invalidString = "Sabin"

# Convert and handle results
validInteger = convertToInt(validString)
invalidInteger = convertToInt(invalidString)

print("Valid integer:", validString)
print("Invalid integer:", invalidString)
```

```
Error: The string 'Sabin' is not a valid integer.
Valid integer: 20
Invalid integer: Sabin
```

20. Scenario: You have a list containing different data types, and you want to separate out all the strings. a. Question: How would you extract all string elements from the following list?

We can use filter function to extract all string from the list containing different data types. For example:

In [37]:
```
#  list with different data types
mixedList = [45, 'New York', 3.14, 'Food', True, 'Dev', 10, False, 200]

# Defining a function to check if an element is a string
def isString(element):
    return isinstance(element, str)

# Extract all string elements using filter and the is_string function
stringElements = list(filter(isString, mixedList))
```

```
print("String elements:", stringElements)
```

```
String elements: ['New York', 'Food', 'Dev']
```