**Q89. How to get items of series A that are not available in another series B?**

**Ans.**  The below program will help you in identifying items in series A but no in series B:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pandas as pd    # Creating 2 pandas Series  ps1 = pd.Series([2, 4, 8, 20, 10, 47, 99])  ps2 = pd.Series([1, 3, 6, 4, 10, 99, 50])    print("Series 1:")  print(ps1)  print("nSeries 2:")  print(ps2)    # Using Bitwise NOT operator along  # with pandas.isin()  print("nItems of Series 1 not present in Series 2:")  res = ps1[~ps1.isin(ps2)]  print(res) |

**Output:**

Series 1:

0     2

1     4

2     8

3    20

4    10

5    47

6    99

Series 2:

0     1

1     3

2     6

3     4

4    10

5    99

6    50

Items of Series 1 not present in Series 2:

0     2

2     8

3    20

5    47

**Q90. How to get the items that are not common to both the given series A and B?**

**Ans.**  The below program is to identify the elements which are not common in both series:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import pandas as pd  import numpy as np  sr1 = pd.Series([1, 2, 3, 4, 5])  sr2 = pd.Series([2, 4, 6, 8, 10])  print("Original Series:")  print("sr1:")  print(sr1)  print("sr2:")  print(sr2)  print("nItems of a given series not present in another given series:")  sr11 = pd.Series(np.union1d(sr1, sr2))  sr22 = pd.Series(np.intersect1d(sr1, sr2))  result = sr11[~sr11.isin(sr22)]  print(result) |

**Output:**

Original Series:

sr1:

0    1

1    2

2    3

3    4

4    5

sr2:

0     2

1     4

2     6

3     8

4    10

Items of a given series not present in another given series:

0     1

2     3

4     5

5     6

6     8

7    10

Next, let us have a look at some Basic Python Programs in these Python Interview Questions.

**Basic Python Programs – Interview Questions**

**Q91. Write a program in Python to execute the Bubble sort algorithm.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | def bs(a):  # a = name of list     b=len(a)-1nbsp;  # minus 1 because we always compare 2 adjacent values     for x in range(b):          for y in range(b-x):                a[y]=a[y+1]       a=[32,5,3,6,7,54,87]     bs(a) |

**Output:**[3, 5, 6, 7, 32, 54, 87]

**Q92. Write a program in Python to produce Star triangle.**

|  |  |
| --- | --- |
| 1  2  3  4 | def pyfunc(r):      for x in range(r):          print(' '\*(r-x-1)+'\*'\*(2\*x+1))  pyfunc(9) |

**Output:**

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Q93. Write a program to produce Fibonacci series in Python.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | # Enter number of terms needednbsp;#0,1,1,2,3,5....  a=int(input("Enter the terms"))  f=0;#first element of series  s=1#second element of series  if a=0:     print("The requested series is",f)  else:    print(f,s,end=" ")     for x in range(2,a):           print(next,end=" ")           f=s           s=next |

**Output:** Enter the terms 5 0 1 1 2 3

**Q94. Write a program in Python to check if a number is prime.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | a=int(input("enter number"))  if a=1:     for x in range(2,a):           if(a%x)==0:            print("not prime")     break     else:        print("Prime")  else:     print("not prime") |

**Output:**

enter number 3

Prime

**Q95. Write a program in Python to check if a sequence is a Palindrome.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | a=input("enter sequence")  b=a[::-1]  if a==b:    print("palindrome")  else:    print("Not a Palindrome") |

**Output:**

enter sequence 323 palindrome

**Q96. Write a one-liner that will count the number of capital letters in a file. Your code should work even if the file is too big to fit in memory.**

**Ans:**Let us first write a multiple line solution and then convert it to one-liner code.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | with open(SOME\_LARGE\_FILE) as fh:  count = 0  text = fh.read()  for character in text:      if character.isupper():  count += 1 |

We will now try to transform this into a single line.

|  |  |
| --- | --- |
| 1 | count sum(1 for line in fh for character in line if character.isupper()) |

**Q97. Write a sorting algorithm for a numerical dataset in Python.**

**Ans:**The following code can be used to sort a list in Python:

|  |  |
| --- | --- |
| 1  2  3  4 | list = ["1", "4", "0", "6", "9"]  list = [int(i) for i in list]  list.sort()  print (list) |

**Q98. Looking at the below code, write down the final values of A0, A1, …An.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))  A1 = range(10)A2 = sorted([i for i in A1 if i in A0])  A3 = sorted([A0[s] for s in A0])  A4 = [i for i in A1 if i in A3]  A5 = {i:i\*i for i in A1}  A6 = [[i,i\*i] for i in A1]  print(A0,A1,A2,A3,A4,A5,A6) |

**Ans:**The following will be the final outputs of A0, A1, … A6

A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # the order may vary

A1 = range(0, 10)

A2 = []

A3 = [1, 2, 3, 4, 5]

A4 = [1, 2, 3, 4, 5]

A5 = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

**Q99. Write a program in Python to print the given number is odd or even.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | </pre>  def check\_odd\_even(number):  if number % 2 == 0:  print(f"{number} is even.")  else:  print(f"{number} is odd.")    # Input from the user  num = int(input("Enter a number: "))    # Checking if the number is odd or even  check\_odd\_even(num)  <pre> |

In this program a function named check\_odd\_even() is defined which takes a number as input and prints whether it’s odd or even. After the program is executed, it will prompt the user to enter a number and calls this function to determine if the entered number is odd or even.

**Q100. Write a program to check if the given number is Armstrong or not.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | </pre>  def is\_armstrong(num):  # Calculate the number of digits in the number  num\_digits = len(str(num))    # Initialize sum to store the result  sum = 0    # Temporary variable to store the original number  temp = num    # Calculate Armstrong sum  while temp > 0:  digit = temp % 10  sum += digit \*\* num\_digits  temp //= 10    # Check if the number is Armstrong or not  if num == sum:  return True  else:  return False    # Input from the user  number = int(input("Enter a number: "))    # Check if the number is Armstrong or not  if is\_armstrong(number):  print(f"{number} is an Armstrong number.")  else:  print(f"{number} is not an Armstrong number.")  <pre> |

In this program the function is\_armstrong() which takes a number as input and returns **True** if it’s an Armstrong number, otherwise **False**. It will prompt the user to enter a number and calls this function to determine if the entered number is an Armstrong number or not when executed.

**Q101. Write a Python Program for calculating simple interest.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | def calculate\_simple\_interest(principal, rate, time):  # Simple interest formula: SI = (P \* R \* T) / 100  simple\_interest = (principal \* rate \* time) / 100  return simple\_interest    # Input from the user  principal = float(input("Enter the principal amount: "))  rate = float(input("Enter the annual interest rate (in percentage): "))  time = float(input("Enter the time period (in years): "))    # Calculate simple interest  interest = calculate\_simple\_interest(principal, rate, time)    # Display the result  print(f"The simple interest for the principal amount ${principal}, annual interest rate of {rate}%, and time period of {time} years is ${interest}.") |

The function calculate\_simple\_interest() takes the principal amount, annual interest rate, and time period as input and returns the simple interest. Then, it prompts the user to enter these values and calls the function to calculate the simple interest, finally displaying the result.

**Q102. Write a Python program to check whether the string is Symmetrical or Palindrome.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | </div>  <div>    def is\_symmetrical(input\_string):  # Check if the string is symmetrical  return input\_string == input\_string[::-1]    def is\_palindrome(input\_string):  # Remove spaces and convert to lowercase  input\_string = input\_string.replace(" ", "").lower()  # Check if the string is a palindrome  return input\_string == input\_string[::-1]    # Input from the user  string = input("Enter a string: ")    # Check if the string is symmetrical  if is\_symmetrical(string):  print(f"{string} is symmetrical.")  else:  print(f"{string} is not symmetrical.")    # Check if the string is a palindrome  if is\_palindrome(string):  print(f"{string} is a palindrome.")  else:  print(f"{string} is not a palindrome.")    </div>  <div> |

This program defines two functions is\_symmetrical() and is\_palindrome(). The is\_symmetrical() function checks if the string is symmetrical, and the is\_palindrome() function checks if the string is a palindrome. Then, it prompts the user to enter a string and calls these functions to determine whether the entered string is symmetrical or a palindrome, and prints the result accordingly.

**Q104. Write a Python program to find yesterday’s, today’s and tomorrow’s date.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | from datetime import datetime, timedelta    def get\_dates():  # Get today's date  today = datetime.now().date()    # Calculate yesterday's and tomorrow's date  yesterday = today - timedelta(days=1)  tomorrow = today + timedelta(days=1)    return yesterday, today, tomorrow    # Get the dates  yesterday\_date, today\_date, tomorrow\_date = get\_dates()    # Display the dates  print("Yesterday's date:", yesterday\_date)  print("Today's date:", today\_date)  print("Tomorrow's date:", tomorrow\_date) |

This program uses the datetime module to work with dates. It defines a function get\_dates() to calculate yesterday’s, today’s, and tomorrow’s dates using **timedelta** objects. Then, it calls this function and prints the dates accordingly.