In [ ]:
```
# Q1. L is a list defined as L= [11, 12, 13, 14].
L = [11, 12, 13, 14]
```

In [ ]:
```
# 1.WAP to add 50 and 60 to L.
L=L+[50,60]
L
```

Out[ ]:
```
[11, 12, 13, 14, 50, 60]
```

In [ ]:
```
# 2.WAP to remove 11 and 13 from L.
L.remove(11)
L.remove(13)
L
```

Out[ ]:
```
[12, 14, 50, 60]
```

In [ ]:
```
# 3.WAP to sort L in ascending order.
L.sort()
L
```

Out[ ]:
```
[11, 12, 13, 14]
```

In [ ]:
```
# 4.WAP to sort L in descending order.
L.sort(reverse=True)
L
```

Out[ ]:
```
[14, 13, 12, 11]
```

In [ ]:
```
# 5.WAP to search for 13 in L.
L.index(13)
```

Out[ ]:
```
2
```

In [ ]:
```
# 6.WAP to count the number of elements present in L.
len(L)
```

Out[ ]:
```
4
```

In [ ]:
```
# 7.WAP to sum all the elements in L.
sum(L)
```

Out[ ]:
```
50
```

In [ ]:
```
# 8. WAP to sum all ODD numbers in L.
sum([x for x in L if x%2==1])
```

```
Out[ ]: 24
```

```
In [ ]:  # 9.WAP to sum all EVEN numbers in L.
         sum([x for x in L if x%2==0])
```

```
Out[ ]: 26
```

```
In [ ]:  #10 .WAP to sum all PRIME numbers in L.
         is_prime = lambda number: all( number%i != 0 for i in range(2, int(number**
         sum([x for x in L if is_prime(x)])
```

```
Out[ ]: 24
```

```
In [ ]:  # 11.WAP to clear all the elements in L.
         L.clear()
         L
```

```
Out[ ]: []
```

```
In [ ]:  # 12 WAP to delete L.
         del L
         L
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [33], in <module>
      1 # 12 WAP to delete L.
----> 2 del L
      3 L

NameError: name 'L' is not defined
```

```
In [ ]:  # Q2. D is a dictionary defined as D= {1:5.6, 2:7.8, 3:6.6, 4:8.7, 5:7.7}.
         D = {1:5.6, 2:7.8, 3:6.6, 4:8.7, 5:7.7}
         D
```

```
Out[ ]: {1: 5.6, 2: 7.8, 3: 6.6, 4: 8.7, 5: 7.7}
```

```
In [ ]:  # i. WAP to add new entry in D; key=8 and value is 8.8
         D[8]=8.8
         D
```

```
Out[ ]: {1: 5.6, 2: 7.8, 3: 6.6, 4: 8.7, 5: 7.7, 8: 8.8}
```

```
In [ ]:  # ii. WAP to remove key=2.
         D.pop(2)
         D
```

```
Out[ ]: {1: 5.6, 3: 6.6, 4: 8.7, 5: 7.7, 8: 8.8}
```

In [ ]:
```
# iii. WAP to check weather 6 key is present in D.
6 in D
```
Out[ ]:
```
False
```

In [ ]:
```
# iv. WAP to count the number of elements present in D.
len(D)
```
Out[ ]:
```
5
```

In [ ]:
```
# v. WAP to add all the values present D.
print(D)
print("Sum {}".format(sum(D.values())))
```
```
{1: 5.6, 3: 6.6, 4: 8.7, 5: 7.7, 8: 8.8}
Sum 37.4
```

In [ ]:
```
# vi. WAP to update the value of 3 to 7.1.
print("Before Updation D was {}".format(D))
D[3]=7.1
print("After Updation D is {}".format(D))
```
```
Before Updation D was {1: 5.6, 3: 6.6, 4: 8.7, 5: 7.7, 8: 8.8}
After Updation D is {1: 5.6, 3: 7.1, 4: 8.7, 5: 7.7, 8: 8.8}
```

In [ ]:
```
# vii. WAP to clear the dictionary.
D.clear()
D
```
Out[ ]:
```
{}
```

In [ ]:
```
# Q3. S1 is a set defined as S1= [10, 20, 30, 40, 50, 60].
# S2 is a set defined as S2= [40, 50, 60, 70, 80, 90].
S1 = {10, 20, 30, 40, 50, 60}
S2 = {40, 50, 60, 70, 80, 90}
```

In [ ]:
```
# i. WAP to add 55 and 66 in Set S1.
S1.update({55,66})
S1
```
Out[ ]:
```
{10, 20, 30, 40, 50, 55, 60, 66}
```

In [ ]:
```
# ii. WAP to remove 10 and 30 from Set S1.
S1 = S1-{10,30}
S1
```
Out[ ]:
```
{20, 40, 50, 55, 60, 66}
```

In [ ]:
```
# iii. WAP to check whether 40 is present in S1.
40 in S1
```

Out[ ]:
```
True
```

In [ ]:
```
# iv. WAP to find the union between S1 and S2.
S1.union(S2)
```

Out[ ]:
```
{20, 40, 50, 55, 60, 66, 70, 80, 90}
```

In [ ]:
```
# v. WAP to find the intersection between S1 and S2.
S1.intersection(S2)
```

Out[ ]:
```
{40, 50, 60}
```

In [ ]:
```
# vi. WAP to find the S1 - S2.
S1-S2
```

Out[ ]:
```
{20, 55, 66}
```

In [ ]:
```
# Q4. Write the following program.
# i. WAP to print 100 random strings whose length between 6 and 8.
import random
import string
for i in range(100):
    print(''.join(random.choices(string.ascii_uppercase +
                                 string.digits, k = random.randint(6,8))))
```

```
BM07O0V
PKE37W
QN3EGGX5
29S8G54V
RBAZFW
ERL13UP9
7MFB04
Z2O53WO
LCFWB9AO
T91IVVHE
FFDZQ08
KXNPMZD
HYUNQ2B
D24D0L
66AML0
ZSVXGW
C941TMYC
MKVJ86V
QFMBH890
X0UP7ZL
LYOK4Z
B2WVO7J
DYDMH8
NFWS278
OXUZEWPZ
68IYCG71
RTFRBVJJ
C1N9RTO
IHGE5OM
5H5VX4T
TN32959
8UWIZHXG
U2OAOEPC
F52BKAIP
C1S2ZLI
38X78BPZ
Q624LP
1PYLU30N
JSBSAI
BU3F5GBW
4J3810J4
NZVDZWX
ICB0SS
Q5AL7I
8TL6KD5N
VC85SD
UWVK69K
1H3WBPD9
XJQQFO
LM779IO
DN8HVOVS
CIY7DM
BG316UU
7ON7TU
IX93MM
ZFCOR8
2B4WRJ5
6YW03JB
U6M2NTQY
```

```
MVXGHAC
5410OXPP
U43YB8
EBK6963C
5TBJN07W
275Q43
EHE1ZK
QUDUPQIQ
IT9LGE
3A9Y4JRV
D8WZQF4
2H5MMM8
JA2S6QP2
PW07RAIR
LC3BX6
28U57KBG
GVX6BHM
P63L5Q
XFXGHGP8
UR0VKPVT
Z19ULREL
LMJ4PYRJ
YRMFEAY
HV5OI9
DDSMY4Q5
ITBFAV
78TW2E
I9SJWQY
LKP8S08
9EHMWKZ
44JELHIH
6O5EN8
M16N2C
7YTGDDQT
4UXZEM9D
W8UFYFR
8H930QL
66UDTB
3EH70X
3ATH5E3H
H9RXA2I7
```

In [ ]:

```
# ii. WAP to print all prime numbers between 600 and 800.
is_prime = lambda number: all( number%i != 0 for i in range(2, int(number**
count =0
for i in range(600,800):
    if is_prime(i):
        print(i)
        count+=1
print("Total number of primes between 600 and 800 are {}".format(count))
```

```
601
607
613
617
619
631
641
643
647
653
659
661
673
677
683
691
701
709
719
727
733
739
743
751
757
761
769
773
787
797
Total number of primes between 600 and 800 are 30
```

In [ ]:

```
# iii. WAP to print all numbers between 100 and 1000 that are divisible by
for i in range(100,1000):
    if i%7==0 and i%9==0:
        print(i)
```

```
126
189
252
315
378
441
504
567
630
693
756
819
882
945
```

In [ ]:

```
# Q5. WAP to create two lists of 10 random numbers between 10 and 30;
L1 =random.sample(range(10, 30), 10)
L2 = random.sample(range(10, 30), 10)
print("L1 is {}".format(L1))
print("L2 is {}".format(L2))
```

```
L1 is [12, 21, 11, 27, 14, 17, 23, 18, 28, 10]
L2 is [22, 17, 19, 10, 15, 16, 26, 18, 25, 21]
```

In [ ]:

```
# i. Common numbers in the two lists
print("Common numbers in the two lists are {}".format(set(L1).intersection(
```

```
Common numbers in the two lists are {17, 10, 18, 21}
```

In [ ]:

```
# ii. Unique numbers in both the list
print("Unique numbers in both the list are {}".format(set(L1).union(set(L2)
```

```
Unique numbers in both the list are {10, 11, 12, 14, 15, 16, 17, 18, 19, 21,
22, 23, 25, 26, 27, 28}
```

In [ ]:

```
# iii. Minimum in both the list
print("Minimum in both the list are {}".format(min(L1+L2)))
```

```
Minimum in both the list are 10
```

In [ ]:

```
# iv. Maximum in both the list
print("Maximum in both the list are {}".format(max(L1+L2)))
```

```
Maximum in both the list are 28
```

In [ ]:

```
# v. Sum of both the lists
print("Sum of both the lists are {}".format(sum(L1+L2)))
```

```
Sum of both the lists are 370
```

In [ ]:

```
# Q6. WAP to create a list of 100 random numbers between 100 and 900.
L = random.sample(range(100, 900), 100)
print("L is {}".format(L))
```

```
L is [805, 414, 453, 784, 369, 563, 832, 346, 164, 835, 286, 495, 390, 242,
762, 408, 761, 550, 430, 320, 504, 176, 612, 727, 799, 221, 166, 326, 610, 2
56, 460, 425, 553, 635, 846, 841, 424, 513, 162, 434, 397, 634, 148, 589, 46
6, 590, 650, 794, 524, 778, 113, 428, 749, 503, 686, 822, 527, 486, 573, 24
6, 245, 130, 292, 310, 713, 521, 733, 677, 357, 109, 174, 106, 233, 765, 30
0, 368, 379, 337, 102, 537, 253, 315, 107, 149, 156, 319, 471, 351, 622, 66
0, 304, 876, 227, 470, 760, 126, 753, 767, 558, 312]
```

In [ ]:

```
# i. All odd numbers
print("All odd numbers are {}".format(list(filter(lambda x: x%2==1, L))))
```

```
All odd numbers are [805, 453, 369, 563, 835, 495, 761, 727, 799, 221, 425,
553, 635, 841, 513, 397, 589, 113, 749, 503, 527, 573, 245, 713, 521, 733, 6
77, 357, 109, 233, 765, 379, 337, 537, 253, 315, 107, 149, 319, 471, 351, 22
7, 753, 767]
```

In [ ]:

```
# ii. All even numbers
print("All even numbers are {}".format(list(filter(lambda x: x%2==0, L))))
```

```
All even numbers are [414, 784, 832, 346, 164, 286, 390, 242, 762, 408, 550,
430, 320, 504, 176, 612, 166, 326, 610, 256, 460, 846, 424, 162, 434, 634, 1
48, 466, 590, 650, 794, 524, 778, 428, 686, 822, 486, 246, 130, 292, 310, 17
4, 106, 300, 368, 102, 156, 622, 660, 304, 876, 470, 760, 126, 558, 312]
```

In [ ]:

```
# iii. All prime numbers
is_prime = lambda number: all( number%i != 0 for i in range(2, int(number**
print("All prime numbers are {}".format(list(filter(is_prime, L))))
```

```
All prime numbers are [563, 761, 727, 397, 113, 503, 521, 733, 677, 109, 23
3, 379, 337, 107, 149, 227]
```

In [ ]:

```
# Q7. D is a dictionary defined as D={1:"One",2:"Two",3:"Three",4:"Four", 5
D = {1:"One",2:"Two",3:"Three",4:"Four", 5:"Five"}
D
```

Out[ ]:

```
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four', 5: 'Five'}
```

In [ ]:

```
#open a file
f = open("Q7.txt", "w")
#write to file
for key,value in D.items():
    f.write(str(key)+","+str(value)+"\n")
#close file
f.close()
```

In [ ]:

```
#read the file
f = open("Q7.txt", "r")
#print the file
print(f.read())
```

```
1,One
2,Two
3,Three
4,Four
5,Five
```

In [ ]:

```
# Q8. L is a list defined as L={"One","Two","Three","Four","Five"}.
# WAP to count the length of reach element from a list and write to the fil
# format:
# One, 3
# Two, 3
# Four, 4
L = ["One","Two","Three","Four","Five"]
L
```

Out[ ]:

```
['One', 'Two', 'Three', 'Four', 'Five']
```

In [ ]:

```
#open a file
f = open("Q8.txt", "w")
#write to file
for i in L:
    f.write(str(i)+", "+str(len(i))+"\n")
#close file
f.close()
```

In [ ]:

```
#read the file
f = open("Q8.txt", "r")
#print the file
print(f.read())
```

```
One, 3
Two, 3
Three, 5
Four, 4
Five, 4
```

In [ ]:

```
# Q9. Write to the file 100 random strings whose length between 10 and 15.
import random
import string
#open a file
f = open("Q9.txt", "w")
#write to the file
for i in range(100):
    #write the strings with index

    f.write("{}. ".format(i+1)+''.join(random.choices(string.ascii_uppercase
                                     string.digits, k = random.randint(10,15)))+"\n
#close the file
f.close()
```

In [ ]:

```
#read the file
f = open("Q9.txt", "r")
#print the file
print(f.read())
```

1. UXHSPK0ALATQNLF
2. DFT52RWRB8
3. ARTQ0VF5VYYWH8D
4. JYRPLR7YE2
5. D3YLCZST3J2K
6. 7ELX8ZEBFICPG
7. 63I8M38B4L
8. VNGNHW9Q5BWNCZR
9. 47XYL3367T
10. B5HAR0I63YT
11. LMEQRMBNDS9PL
12. 8YTI8FBNOU9V
13. VSODHF95FZTOHF4
14. CTD7VMXX7BI6WK
15. 5ESHYJOAVW
16. 9UVFWYVOPDKQIA
17. LNDMX76I23
18. XY495I8MUJU2H8
19. RXVWHP9RPYH
20. SD3C1A61LS
21. 65FQX6GD7L39KM
22. 2A4LN1IEJ9MU
23. 77NJX9O5MMQYOP
24. EYTN4S3YPCYXD
25. BKPGQLR0Q68A
26. SBMVUV7NEIO24I
27. 6UZ5SIKW0H
28. WW9Z5GBFG6
29. UCEPP7I2IBN
30. KMHODE4M5BZVWD1
31. MMZ3B4QHBV
32. 7DVCF8V896JU
33. 84IEMT1YL35
34. DFTTWGDFK6F
35. 0VHR4T6IPUJWE2N
36. GV06CUAC5C
37. YWLO6FDIWC1
38. 35R08XS34L89Z
39. 3B8ABB02S4
40. F3B39RZXVHOQNM4
41. SWZ9DSVQ6CE19D9
42. JSC9H3DNNAIPTF
43. 039HXGIQLL1
44. R26CYBDBUGC
45. 67VMBSRPP3H
46. XQDMPOOIOP8AOQ9
47. W93CXOEZFXBUC
48. KSNP0NJUEAXMC
49. 8JZXP2ADVN3PFTK
50. JAGU815HPU
51. DQR70AVV4EAIRQ
52. M2RNZCEIZJ8HF
53. WYX1S136I45
54. 5XKRY86KQ542HHY
55. DKLB4RIYGR
56. XC2W28JBCBU2F7
57. ZU8QI7FTQORY4NO
58. XS7XSW2ALVXH98A
59. WWWFRRWTYO

```
60. 3F8DLE8QQA8
61. MNDNOOOLZ0T29X
62. GBMBZ3ZC12FO
63. 0O3C1R6CVQM7PY
64. ICLWOIA0I6ITFYX
65. RWJ9NI2QDHRED5
66. C0SGUD28V5
67. RXZN0Q56NIP
68. URZ7INLI2I
69. B0MIIDYXOOQPC
70. 3K2FTWMB697BC
71. E3C240XDTLALD
72. 7XUEE85L6BWD6G
73. M4OY7K0VUV
74. KL5XL779FJYYFIG
75. K9LA7ZDW1I1JDM
76. 8TUTAFIKHM
77. LQYGZ753ID
78. QS8HWN560W
79. C9AIENB5JCX7183
80. XEIMKJAPJ9AWIWI
81. GYBQNSYKVJ
82. JAW6O94RAS
83. AK7NRWD89YJ
84. ZECJAYAWCIX
85. 4WKOX4WMLFMA
86. CCFXNS14ZFM
87. 7X3B9KRKYK7Y
88. ZTN3CSEKXSMO7
89. MV6W9TKLJY3
90. N6EF5AFY8WYM4XM
91. KO5PNEU3C0VIAZ
92. CB7NOBS140ONF
93. X3LJQV6TAFDOR
94. QJ43RRHLN86
95. RI5YUWNHL2Q
96. IV2961PQ5I
97. E6XBGCT2WL34
98. M5BO3H5EYJZU53N
99. MYTHZXUNZGKCPXZ
100. 0TR5D3L4VML36E3
```

In [ ]:

```
# Q10. Write to the file all prime numbers between 600 and 800.
is_prime = lambda number: all( number%i != 0 for i in range(2, int(number**
count=0
#open a file
f = open("Q10.txt", "w")
#write to the file
for i in range(600,800):
    if is_prime(i):
        count+=1
        f.write(str(i)+"\n")
f.write("The total number of primes between 600 and 800 are {}".format(coun
#close the file
f.close()
```

In [ ]:
```
#read the file
f = open("Q10.txt", "r")
#print the file
print(f.read())
```

```
601
607
613
617
619
631
641
643
647
653
659
661
673
677
683
691
701
709
719
727
733
739
743
751
757
761
769
773
787
797
The total number of primes between 600 and 800 are 30
```

In [ ]:
```
# Q11. WAP to calculate the time taken by a program.
import time
start_time = time.time()
#your code here
#prime number between 1 and 1000000
is_prime = lambda number: all( number%i != 0 for i in range(2, int(number**
count=0
for i in range(1,1000000):
    if is_prime(i):
        count+=1
print("The total number of primes between 1 and 10000 are {}".format(count)
print("Time taken by the program is {}".format(time.time()-start_time))
```

```
The total number of primes between 1 and 10000 are 78499
Time taken by the program is 8.20159649848938
```

In [ ]:

```
#Q12. WAP to create a dictionary of student marks in five subjects and you 
# student having maximum and minimum average marks.
marks = {
    "Raj":{
        "English":80,
        "Maths":90,
        "Science":70,
                    "Social":60,
        "Computer":85
    },
    "Ravi":{
        "English":70,
        "Maths":80,
        "Science":90,
        "Social":80,
        "Computer":75
    },
    "Rajesh":{
        "English":90,
        "Maths":60,
        "Science":80,
        "Social":70,
        "Computer":70
    },
    "Rajni":{
        "English":80,
        "Maths":70,
        "Science":80,
        "Social":70,
        "Computer":80
    },
    "Raju":{
        "English":70,
        "Maths":80,
        "Science":70,
        "Social":90,
        "Computer":70
    }
}
```

In [ ]:

```
# find the student having maximum and minimum average marks.
avg_marks = {}
for key, value in marks.items():
    avg_marks[key] = sum(value.values())/len(value)
print("Average Marks of all students are {}".format(avg_marks))
print("The student having maximum average marks is {}".format(max(avg_marks
print("The student having minimum average marks is {}".format(min(avg_marks
```

```
Average Marks of all students are {'Raj': 77.0, 'Ravi': 79.0, 'Rajesh': 74.
0, 'Rajni': 76.0, 'Raju': 76.0}
The student having maximum average marks is Ravi
The student having minimum average marks is Rajesh
```

In [ ]:

```
# Q13. WAP to sort the following number of elements in a list and calculate
# number of elements being 5k,10k,15k,20k,25k
L = [5000,10000,15000,20000,25000]
for i in L:
    L = random.sample(range(1, 1000000), i)
    start_time = time.time()
    L.sort()
    print("Time taken to sort {} elements is {}".format(i, time.time()-star
```

```
Time taken to sort 5000 elements is 0.0009968280792236328
Time taken to sort 10000 elements is 0.0019855499267578125
Time taken to sort 15000 elements is 0.0029804706573486328
Time taken to sort 20000 elements is 0.00498652458190918
Time taken to sort 25000 elements is 0.00799250602722168
```

In [ ]: