```
#1 1d
#Which of the following functions can be used to eliminate multiple
occurrences of a character or characters which are mistakenly added?
#a) remove() b) strip() c) pop() d replace()
s="  ababab  "
#no s.remove() s.pop() functions in python for strings
print(s.strip()) #returns "ababab"
#s.replace('a','') replaces all a's with '' therefore removing it
s.replace('a','')
```

```
ababab
```

```
'  bbb  '
```

```
#2 2b
#What will be the output of the following code?
A=['Abe', 'pq', 'abe', '2020', '111', 'pal']
#max(A) returns 'pq' since pq is last in dictionary order
#min(A) returns '111' since 111 is first in dictionary order
print(max(A)+min(A))
#a) AttributeError b)pq111 c)2020 d)pal111
```

```
pq111
```

```
#3 3C
#What is the output of the following?
s='Welcome to Grade 12'.lower()
#index=-19 -18 -17 -16 -15 -14 -13 -12 -11 -10 -9  -8  -7  -6  -5  -4
-3  -2   -1
#s=    'w' 'e' 'l' 'c' 'o' 'm' 'e' ' ' 't' 'o' ' ' 'g' 'r' 'a' 'd' 'e'
' ' '1' '2'
#index= 0   1   2   3   4   5   6   7   8   9  10   11 12   13 14  15
16  17  18
#[start: end : step]
print(s[-2:2:-1]) # returns 's[-2/17],s[-3/16],s[-4/15],s[-5/14],s[-
6/13],s[-7/12],s[-8/11],s[-9/10],s[-10/9],s[-11/8],s[-12/7],s[-
13/6],s[-14/5],s[-15/4],s[-16/3]'
print(s[-2:2:-2]) # returns 's[-2/17],s[-4/15],s[-6/13],s[-8/11],s[-
10/9],s[-12/7],s[-14/5],s[-16/3]'
#s[-2:2] returns '1 edarg ot emocl'
#a) 'leaGo mc' b)[] c) 'leago mc' d) '2 dr teo'
```

```
1eago mc
1 edarg ot emoc
```

```
#4 4d
#Given
Lst=[30,10,90,80]
#. Which of the foiiowing will remove all the elements in the list and
make it an empty list?
```

```
#a) del Lst[:] #Lst[:] is same as Lst so a option is del Lst which
removes all elements
#b) Lst[:]=[]  #Lst[:] is same as Lst so b option is Lst=[] which
assigns an empty list to Lst
#c) del Lst[0:len(Lst)] #Lst[0:len(Lst)] is same as Lst so c option is
del Lst[] which removes all elements
#d) All of the above
del Lst[:]
print(Lst)
Lst = [30,10,90,80]
Lst[:]=[]
print(Lst)
Lst=[30,10,90,80]
del Lst[0:len(Lst)]
print(Lst)
#so all options are correct
```

```
[]
[]
[]
```

```
#5
#Write a python statement to sort the keys of a dictionary named "emp"
in alphabetical order and store the result in sorted keys.
emp={'c':3,'d':4,'a':1,'b':2}
sorted_keys=sorted(emp.keys())
print(sorted_keys)
```

```
['a', 'b', 'c', 'd']
```

```
#6 6b
#Write the output of the following code:
print(len('programming'.upper().split("M")))
#print(len('PROGRAMMING'.split("M")))
#split cuts the string into list of 2 strings based on the delimiter
#1 step ['PROGRA', 'MING']
#2 step ['PROGRA', '', 'ING']
#print(len(['PROGRA','','ING']))
#ans 3
#a) 2 b) 3 c) 4 d) 0
```

```
3
```

```
#7 7C
#Which of the following statement(s) would give an error during the
execution of the following code?
#R={'Pno':52, 'Pname': Virat, 'Expert': ['Badminton', 'Tennis'],
'Score':(77,44)}
R={'Pno':52, 'Expert': ['Badminton', 'Tennis'], 'Score':(77,44)}
print(R)    #Statement 1
R['Expert'][0]='Cricket' #Statement 2
```

```
#R['Score'][0]=50    #Statement 3
R['Pno']=50 #Statement 4
print(R)

#a) Statement 1 b)Statement 2 c) Statement 3 d) Statement 4
#first Virat must be in quotes or must be a defined variable
#then in statement 3 R['Score'] return tuple  R['Score'][0] return 1st
tuple elmenet which is immutable which returns error
```

```
{'Pno': 52, 'Expert': ['Badminton', 'Tennis'], 'Score': (77, 44)}
{'Pno': 50, 'Expert': ['Cricket', 'Tennis'], 'Score': (77, 44)}
```

```
#8 8d
#Which of the following statements will remove the last item from a
dictionary named peron.
#a) person.pop(-1) b) person.pepitem(-1)
#c) person.pop() d) person.popitem()

person={'name':'virat','age':32,'country':'india'}
#person.pop(-1) and person.pop() are invalid as pop() takes key as
argument
#person.popitem(-1) is invalid as popitem() takes no argument
person.popitem()
print(person)
```

```
{'name': 'virat', 'age': 32}
```

```
#9 9a,b,c
#What are the possible outcome(s) executed from the following code?
import random
DIR=['North', 'South', 'East', 'West']
NUM=random.randint(1,3)
N=''
for i in range(NUM,1,-1):
    N+=DIR[i]
print(N)
#a) WestEast b) East c) No output d) WestEastSouth
#possible outcome of random.randint(1,3) is 1,2,3
#possible outcome of range(NUM,1,-1) is (3,1,-1),(2,1,-1),(1,1,-1) =>
[3,2],[2],[]
#possible outcome of N+=DIR[i] is {N+=DIR[3] N+=DIR[2]},{N+=DIR[2]},{}
=> WestEast,East,''A
#ans a,b,c
```

```
East
2
```

```
#10 10C
#Predict the output
numbers=[1,2,3,4]
numbers.append([5,6,7,8]) #append takes the whole argument and adds it
```

```
to the numbers list at end as an element
#numbers = [1,2,3,4,[5,6,7,8]]
print(len(numbers)) # no of elements in numbers is 5
#a) 8 b) Error as we cannot append a list to a list
#c) 5 d) 7
#ans c
```

```
5
```

```
#11 11a,c
#Which of the following statements is used to reverse a list in-place
in Python?
#a) myList.reverse() b) myList =myList.reverse()
#c) myList= myList[::-1] d) d.reversed(myList)
myList = [1,2,3]
myList.reverse() # reverse() reverses the list on which it is being
called
print(myList)
#myList = myList.resverse(), reverse() does not return any values
# there is no d.reversed()myList function
myList =myList[::-1] # => myList[-1,-4,-1] or myList[-1,-(len(myList)
+1),-1] which reverses the list
print(myList)
```

```
[3, 2, 1]
[1, 2, 3]
```

```
#12
#Assertion(A):Python code in one module gains access to the code in
another module by the process of importing it.
#Reason(R): The import statement binds the module to the local scope.

# Ans both assertion and reason are true
# Python code in one module gains access to the code in another module
by the process of importing it.
# The import statement combines two operations; it searches for the
named module, then it binds the results of that search to a name in
the local scope.


#13
#Assertion (A): 'python'.partition('P') returns ('python', '', '')
#Reason(R): Partition method splits the ring at the first occurrence
of sep, and returns a tuple containing 3 elements.
# Reason is true and if separation('P') is not present as in this
case('python') it returns the  string and 2 empty strings so Assertion
is also true
print('python'.partition('P'))
```

```
('python', '', '')
```

```
#14
#Observe the following code carefully and rewrite it after removing
all the errors. Underline all the corrections made.
#s=ToGGle
#for i in s:
#    if s[i].isalpha:
#        if s[i].isupper:print(i,s[i].lower')
#            else:print(i,i.upper)

s='ToGGle' #string must be in quoats
for i in range(len(s)): #for i in s iterates over the elements of s,
since we need i as index we can change it to range(len(s))
    if s[i].isalpha: # here function isaplpha with paranthesis is used
it return function reference, then if evaluates that reference to
truth value
        if s[i].isupper:print(i,s[i].lower())#.lower() and .upper()
needs to have brackets()
        else:print(i,s[i].upper())#else must be indented as if level
and we must use paranthesis to get function return value
```

```
0 t
1 o
2 g
3 g
4 l
5 e
```

```
#15
#Predict the output of the following code:
S='AIM'
L=[10,21,33,4]
for i in range(len(S)): #range(4) => 0,1,2
    if i%20==0:         #0%20=>0, 1%20=>1, 2%20=>2
        L.insert(i,S[i])#L.insert(0,'A') => L=['A',10,21,33,4]
    else:
        L.pop(i)        #L.pop(1),L.pop(2)
    print(L)            #['A', 10, 21, 33, 4], ['A', 21, 33, 4], ['A',
21, 4]
print(L.clear(),len(L))
```

```
['A', 10, 21, 33, 4]
['A', 21, 33, 4]
['A', 21, 4]
None 0
```

```
#16
#Complete the following python script by filling in the blank

questions=['name', 'location', 'favorite langage']
answers=['XXX', 'Chennai', 'Python']
```

```
#QA = ________
#for q,a in ________ (questions, answers):
#    QA[________] = ________

QA = {} # assign QA as empty dictionary
for q,a in zip(questions, answers): # zip is used to iterate over two
or more iterables
   QA[q] = a    # assign q as key and a as value

print(QA)
```

```
{'name': 'XXX', 'location': 'Chennai', 'favorite langage': 'Python'}
```

```
#17
#a) Differentiate between sort() and sorted().
# sort() sorts the list in-place and returns None, sorted() takes an
iterable and returns a sorted list
#or
#b) Differentiate between del statement and clear method with respect
to list.
#del statement delets the given range of elements from the list but
clear method makes the list empty
```

```
[]
```

```
#18
#Find the output of the following code?
T=((1,(2,(3,(4,)))), 'e',3) # T is a tuple with 3 elements (1,(2,(3,
(4,)))), 'e', 3 so len(T)=3
print(len(T),T[0][1][1]) #T[0] is 1st element (1,(2,(3,(4,)))) which
is a tuple with 2 element 1, (2,(3,(4,)))
                         #T[0][1] is 2nd element (2,(3,(4,))) which is
a tuple with 2 elements 2, (3,(4,))
                         #T[0][1][1] is 3rd element (3,(4,)) so ans is
3 (3,(4,))
```

```
3 (3, (4,))
```

```
#19
#a) Given the following dictionaries
dict_exam={"Exam":"SSCE", "Year":2024}
dict_result={"Total": 500, "Pass_Marks":165}
#Write a statement to merge the contents of both dictionaries?
dict_exam.update(dict_result)
print(dict_exam)
#b) Explain the difference between get() and pop() methods of python
dictionaries.
#get() is used to get the value of the key and pop() is used to delete
the key and value it also returns the value
print(dict_exam.get("Exam"),dict_exam.pop("Year"))
print(dict_exam)
```

```
{'Exam': 'SSCE', 'Year': 2024, 'Total': 500, 'Pass_Marks': 165}
SSCE 2024
{'Exam': 'SSCE', 'Total': 500, 'Pass_Marks': 165}
```

```python
#20
#Predict the output of the following code:
m=""
s='Fun@Python3.0'
for i in range(0,len(s)):#range(0,len(s)) is range(13) 0..12
    if(s[i].isupper()): #FP
        m=m+s[i].lower()#fp
    elif s[i].islower():#unython
        m=m+s[i].upper()#UNYTHON
    else:               #@3.0 #index of @3.0 is [3,10,11,12]
        if i%2==0:      #[3,10,11,12] => [false,true,false,true]
            m=m+s[i-1]  #[10,12](30) => n.
        else:
            m=m+"#"     #[3,11](@.) => ##
print(m) # m => fUN#pYTHONn#.
```

```
fUN#pYTHONn#.
```

```python
#21
#Write a program to accept a string and count the number of words
without vowel characters.
#Sample input:"My Rhythms fly" has no vowels in it
#Sample output: 3

def count_words_without_vowels(str):
    str = str.lower() # convert the string to lowercase so that we
don't need to worry about case sensitiveness
    count = 0         # set initial value of count to 0
    for w in str.split(): # split the string into list of words
        if(('a' in w) or ('e' in w) or ('i' in w) or ('o' in w) or
('u' in w)): # check if any vowel is present in the word
            pass # if present, pass
        else:
            count += 1 #else increment the count
    return count # return the count at end

count_words_without_vowels("My Rhythms fly")
```

```
3
```

```python
#22
#Consider:
Str,Lst="Global warming",['E', 'Y', 'A', 'M', '2024']
#Write Python statements (single statements only) for each of the
following tasks using BUILT-IN functions/methods only:
#a) To replace all occurrences of letter 'a' with '*'.
#b) To return the highest and the least value in the string.
```

```
#c) To check if the string is in the title case.
#d) To arrange the list elements in its alphabetical order.

print(Str.replace('a','*')) # replace all occurrences of letter 'a'
with '*'
print(max(Str),min(Str)) # max char is w and min char is ' ' space
print(Str.istitle())
Lst.sort()
print(Lst)
```

```
Glob*l w*rming
w
False
['2024', 'A', 'E', 'M', 'Y']
```

```
#23
#Amit attempted to write a program to print all unique substrings of a
given string. Help him to complete the program.
# S=________("Any String:")
# res=[]
# for i in range(len(s)):
#     for j in range(________,len(s) + 1):
#         res.________(s[________])
# print(res)

s=list("Any String:") #s is list of characters
res=[]
for i in range(len(s)): #iterate i 0 to len(s)-1
    for j in range(i,len(s) + 1): # iterate j from i to len(s)+1
        res.append(s[i:j]) # use i,j to create substrings
print(res)
```

```
[[], ['A'], ['A', 'n'], ['A', 'n', 'y'], ['A', 'n', 'y', ' '], ['A',
'n', 'y', ' ', 'S'], ['A', 'n', 'y', ' ', 'S', 't'], ['A', 'n', 'y', '
', 'S', 't', 'r'], ['A', 'n', 'y', ' ', 'S', 't', 'r', 'i'], ['A',
'n', 'y', ' ', 'S', 't', 'r', 'i', 'n'], ['A', 'n', 'y', ' ', 'S',
't', 'r', 'i', 'n', 'g'], ['A', 'n', 'y', ' ', 'S', 't', 'r', 'i',
'n', 'g', ':'], [], ['n'], ['n', 'y'], ['n', 'y', ' '], ['n', 'y', '
', 'S'], ['n', 'y', ' ', 'S', 't'], ['n', 'y', ' ', 'S', 't', 'r'],
['n', 'y', ' ', 'S', 't', 'r', 'i'], ['n', 'y', ' ', 'S', 't', 'r',
'i', 'n'], ['n', 'y', ' ', 'S', 't', 'r', 'i', 'n', 'g'], ['n', 'y', '
', 'S', 't', 'r', 'i', 'n', 'g', ':'], [], ['y'], ['y', ' '], ['y', '
', 'S'], ['y', ' ', 'S', 't'], ['y', ' ', 'S', 't', 'r'], ['y', ' ',
'S', 't', 'r', 'i'], ['y', ' ', 'S', 't', 'r', 'i', 'n'], ['y', ' ',
'S', 't', 'r', 'i', 'n', 'g'], ['y', ' ', 'S', 't', 'r', 'i', 'n',
'g', ':'], [], [' '], [' ', 'S'], [' ', 'S', 't'], [' ', 'S', 't',
'r'], [' ', 'S', 't', 'r', 'i'], [' ', 'S', 't', 'r', 'i', 'n'], [' ',
'S', 't', 'r', 'i', 'n', 'g'], [' ', 'S', 't', 'r', 'i', 'n', 'g',
':'], [], ['S'], ['S', 't'], ['S', 't', 'r'], ['S', 't', 'r', 'i'],
['S', 't', 'r', 'i', 'n'], ['S', 't', 'r', 'i', 'n', 'g'], ['S', 't',
'r', 'i', 'n', 'g', ':'], [], ['t'], ['t', 'r'], ['t', 'r', 'i'],
['t', 'r', 'i', 'n'], ['t', 'r', 'i', 'n', 'g'], ['t', 'r', 'i', 'n',
'g', ':'], [], ['r'], ['r', 'i'], ['r', 'i', 'n'], ['r', 'i', 'n',
'g'], ['r', 'i', 'n', 'g', ':'], [], ['i'], ['i', 'n'], ['i', 'n',
'g'], ['i', 'n', 'g', ':'], [], ['n'], ['n', 'g'], ['n', 'g', ':'],
[], ['g'], ['g', ':'], [], [':']]
```

```python
#24
#Write a menu driven program to perform the following operations on a
dictionary named SCOREBOARD which stores the player name and runs
scored as key value pairs.
# a) add a new player
# b) update score
# c) search score by player name.
# d) print the overall score.
# e) print the score board.

SCOREBOARD={}

def add_new_player():
    name=input("Enter player name: ")
    score=int(input("Enter player score: "))
    if name in SCOREBOARD:
        print("Player already exists")
    else:
        SCOREBOARD[name]=score
        print("Player added successfully")

def update_score():
    name=input("Enter player name to update score: ")
    if name in SCOREBOARD:
        score=int(input("Enter new score: "))
        SCOREBOARD[name]=score
        print("Score updated successfully")
    else:
        print("Player not found")

def search_score():
    name=input("Enter player name to search score: ")
    if name in SCOREBOARD:
        print("Score: ",SCOREBOARD[name])
    else:
        print("Player not found")

def print_overall_score():
    total_score = 0
    for name, score in SCOREBOARD.items():
        total_score += score
    print("Overall score: ", total_score)
```

```
def print_scoreboard():
    print("Scoreboard: ")
    for name, score in SCOREBOARD.items():
        print(name, ":", score)

while True:
    print("1. Add new player")
    print("2. Update score")
    print("3. Search score")
    print("4. Print overall score")
    print("5. Print scoreboard")
    print("6. Exit")
    choice=int(input("Enter your choice: "))
    if choice==1:
        add_new_player()
    elif choice==2:
        update_score()
    elif choice==3:
        search_score()
    elif choice==4:
        print_overall_score()
    elif choice==5:
        print_scoreboard()
    elif choice==6:
        break
    else:
        print("Invalid choice")
```

```
1. Add new player
2. Update score
3. Search score
4. Print overall score
5. Print scoreboard
6. Exit
```

```
#25
#a) Write a program to reverse a string.
#sample input: I am a star
#Sample output: star a am I

def reverse_string(string):
    return string[::-1]

#b) Suppose we have an array nums. Here a pair (i,j) is said to be a
good pair if nums[i] is the same as nums[j] and i<j. Write a program
to print the indices of good pairs.
#So, if the input is like nums = [5,6,7,5,5,7], then the output will
be 4 as there are 4 good pairs the indices are (0, 3), (0, 4) (3, 4),
(2, 5)
```

```python
def good_pairs(nums):
    count = 0
    for i in range(len(nums)-1): # iterate i from 0 to len(nums)-2
        for j in range(i+1, len(nums)): # iterate j from i+1 to
len(nums) so i<j
            if nums[i] == nums[j]:
                count += 1
    return count

good_pairs([5,6,7,5,5,7])
```

```
4
```