Assignment # 5

SER-501

Abdul Samad Khan(akhan51)

Question1(Required)

```
def longest_ordered_subsequence(L):
    n = len(L)
    # if the array is empty
    if(n == 0):
        return 0
    # set an array of len n to 1
    T = [1 for i in range(n)]
    # iterating over every value
    for i in range(1, n):
        for j in range(0, i):
            # check if the previous value is greater or not
            if(L[i] > L[j]):
                if(T[j] + 1 > T[i]):
                    T[i] = T[j] + 1
    check = 0
    # check the values of the new array
    # will only take those values that are greater then the previous
    for k in range(0, len(T)):
        if(T[k] > T[check]):
            check = k
    return T[check]
```

Question2(Required)

```
def partition_set_solver(S):
    total = sum(S)

    if total & 1 == 1:
        return False

    total >>= 1
    n = len(S) + 1

    dp = [[False for i in range(total + 1)] for j in range(n)]

    for i in range(n):
        dp[i][0] = True

    for i in range(1, n):
        for j in range(1, total + 1):
            dp[i][j] = dp[i - 1][j]
            if j >= S[i - 1]:
```

```
            dp[i][j] = dp[i][j] or dp[i - 1][j - S[i - 1]]
    return dp[n - 1][total]
```

```
def subset_sum_solver(S, n):
    # if target value is greater
    if(n < 0):
        return False
    # if S is empty
    if(len(S) == 0):
        return False
    s = sum(S)
    # if sum is less then the target value
    if(s < n):
        return False
    # if 2*target is less then sum
    if(2 * n < s):
        value = s - 2 * n
    # if 2*target greater then the sum
    if(2 * n >= s):
        value = 2 * n - s
    # append value to S
    S.append(value)
    # run partition_set_solver and return value
    return partition_set_solver(S)
```

Question3(Optional)

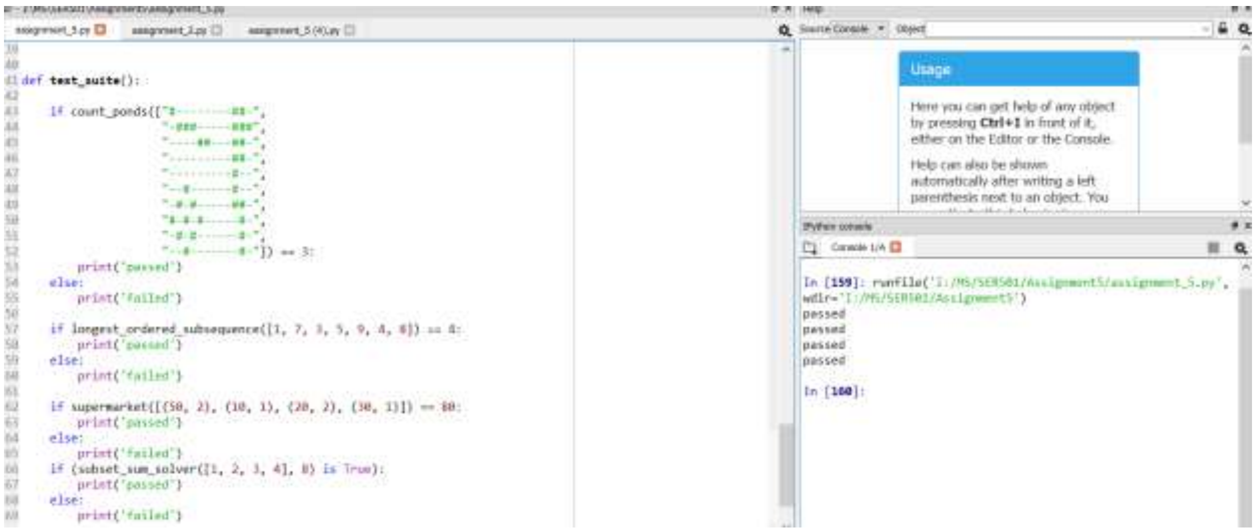
```
def count_ponds(G):
    # check if the string is empty or not
    if(len(G) == 0):
        return 0
    m = len(G)
    n = len(G[0])
    total = 0
    # making a 2d array to keep track of visited positions
    visited = [[0 for j in range(len(G[0]))]for i in range(len(G))]
    # traversing each point in our 2d array
    for i in range(0, m):
        for j in range(0, n):
            # if the node has not been alreaddy visited
            if(visited[i][j] == 0 and G[i][j] == '#'):
                total = total + 1
                # apply DFS to the node
                DFS(G, i, j, visited)
    return total
```

```
def DFS(G, i, j, visited):
    # x y would help us to access all the 8 neighbors
    x = [-1, -1, -1, 0, 0, 1, 1, 1]
    y = [-1, 0, 1, -1, 1, -1, 0, 1]
    # setting the node to true
    visited[i][j] = 1
    # loop for all the 8 neighbors
    for d in range(0, 8):
        a = x[d] + i
        b = y[d] + j
        # check if the node is not in the corner and check if it is not out of
        # range and if it has already been visited or not
        if(a >= 0 and b >= 0 and a < len(G) and b < len(G[0]) and
           visited[a][b] == 0 and G[a][b] == '#'):
              DFS(G, a, b, visited)
```

Question4(Optional)

```
def supermarket(Items):
    n = len(Items)
    # if array is empty
    if(n == 0):
        return 0
    # sort items according to their deadline
    Items = sorted(Items, key=lambda x: x[1])
    # make an array to keep track of values
    T = [0 for i in range(n)]
    # setting prices of the Items to the array
    for i in range(0, n):
        T[i] = Items[i][0]
    # iterating over each elements
    for i in range(1, n):
        for j in range(0, i):
            # if deadlines are not same
            if(Items[j][1] != Items[i][1]):
                # if the value of index at T is
                # greater after adding with Item value
                if(T[j] + Items[i][0] > T[i]):
                    T[i] = Items[i][0] + T[j]
    # return max value of the array T
    return np.amax(T)
```

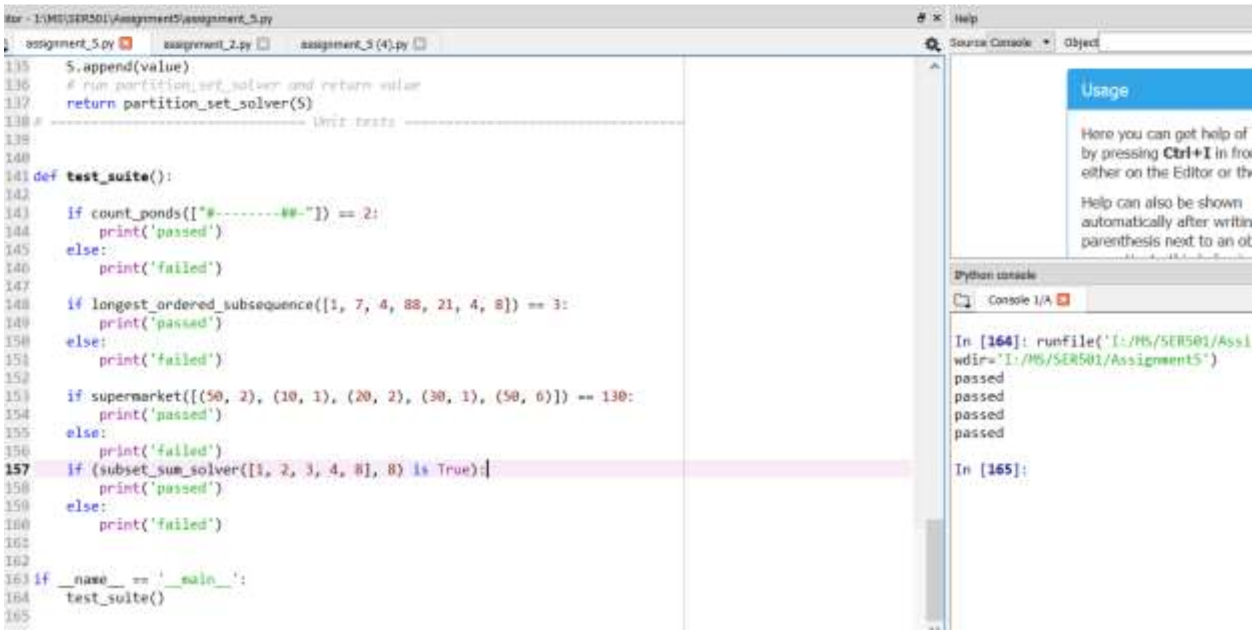
Given test case



```
def test_suite():

    if count_ponds(["#--------##-",
                    "-###-----###",
                    "-----##---##-",
                    "----------##-",
                    "----------#--",
                    "--#-------#--",
                    "-#-#------##-",
                    "#-#-#-------#-",
                    "-#-#-------#-",
                    "--#--------#-"]) == 3:
        print('passed')
    else:
        print('failed')

    if longest_ordered_subsequence([1, 7, 3, 5, 9, 4, 8]) == 4:
        print('passed')
    else:
        print('failed')

    if supermarket([(50, 2), (10, 1), (20, 2), (30, 1)]) == 80:
        print('passed')
    else:
        print('failed')
    if (subset_sum_solver([1, 2, 3, 4], 8) is True):
        print('passed')
    else:
        print('failed')
```

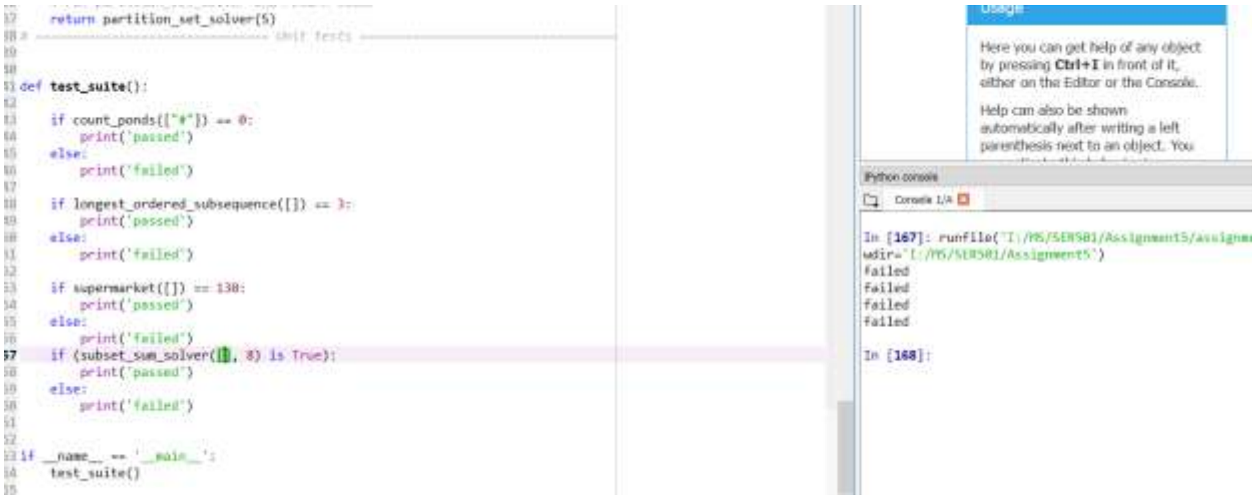
```
In [159]: runfile('I:/MS/SER501/Assignment5/assignment_5.py',
wdir='I:/MS/SER501/Assignment5')
passed
passed
passed
passed

In [160]:
```

Random test case



```
    S.append(value)
    # run partition_set_solver and return value
    return partition_set_solver(S)
# ---------------------------------- Unit tests ----------------------------------


def test_suite():

    if count_ponds(["#--------##-"]) == 2:
        print('passed')
    else:
        print('failed')

    if longest_ordered_subsequence([1, 7, 4, 88, 21, 4, 8]) == 3:
        print('passed')
    else:
        print('failed')

    if supermarket([(50, 2), (10, 1), (20, 2), (30, 1), (50, 6)]) == 130:
        print('passed')
    else:
        print('failed')
    if (subset_sum_solver([1, 2, 3, 4, 8], 8) is True):
        print('passed')
    else:
        print('failed')


if __name__ == '__main__':
    test_suite()
```

```
In [164]: runfile('I:/MS/SER501/Assi
wdir='I:/MS/SER501/Assignment5')
passed
passed
passed
passed

In [165]:
```



```
    return partition_set_solver(5)
# ------------------------------ Unit tests ------------------------------


def test_suite():

    if count_ponds(["#"]) == 0:
        print('passed')
    else:
        print('failed')

    if longest_ordered_subsequence([]) == 3:
        print('passed')
    else:
        print('failed')

    if supermarket([]) == 130:
        print('passed')
    else:
        print('failed')
    if (subset_sum_solver([], 8) is True):
        print('passed')
    else:
        print('failed')


if __name__ == '__main__':
    test_suite()
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

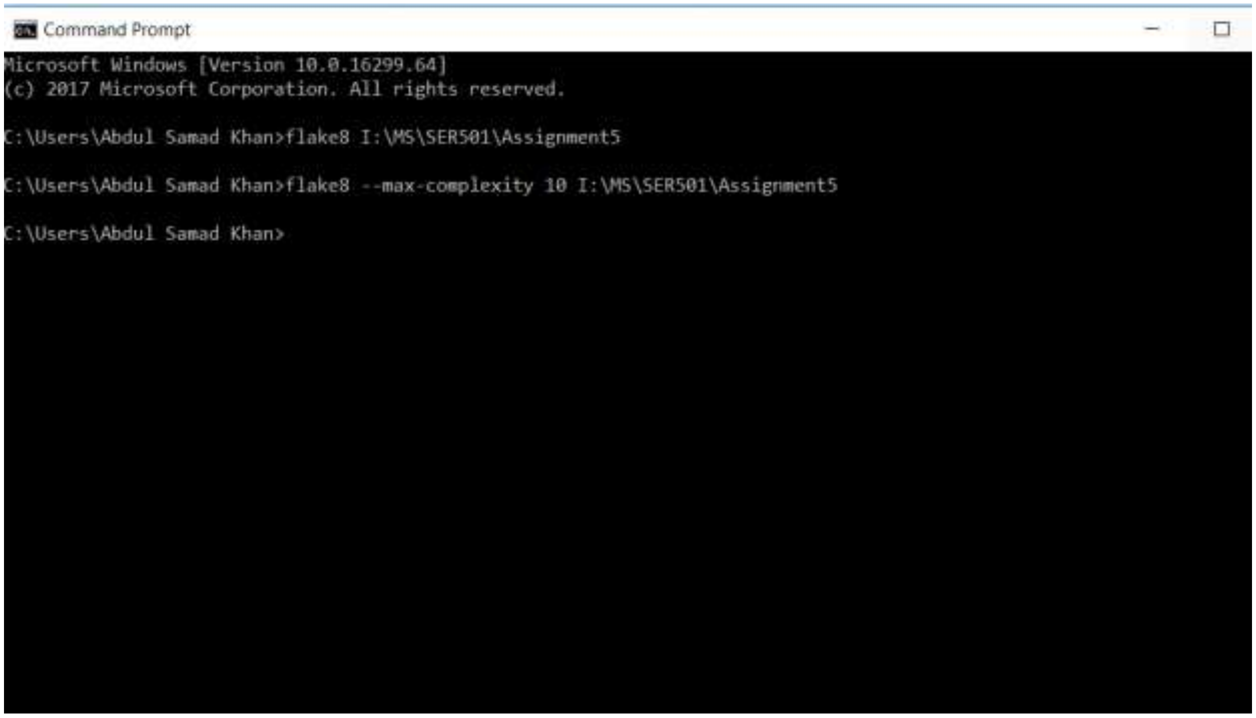
Help can also be shown automatically after writing a left parenthesis next to an object. You

IPython console

Console 1/A

```
In [167]: runfile('I:/MS/SER501/Assignment5/assignm
wdir='I:/MS/SER501/Assignment5')
failed
failed
failed
failed

In [168]:
```

<u>Complexity check and flake8 check</u>



Command Prompt

```
Microsoft Windows [Version 10.0.16299.64]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Abdul Samad Khan>flake8 I:\MS\SER501\Assignment5

C:\Users\Abdul Samad Khan>flake8 --max-complexity 10 I:\MS\SER501\Assignment5

C:\Users\Abdul Samad Khan>
```