X

(https://swayam-uat-central.appspot.com) (https://swayam-uat-central.appspot.com/nc_details/AICTE)

suchetajjw47@gmail.com

AICTE (https://swayam-uat-central.appspot.com/explorer?ncCode=AICTE) » Programming and Data Structures with Python (course)

Course outline

Practice Assignments

● **Practice Assignment 1 (/programming_2021/progassignment?name=4)**

○ Practice Assignment 2 (/programming_2021/progassignment?name=13)

◐ Practice Assignment 3 (/programming_2021/progassignment?name=18)

Practice Quiz 1

Quiz 1, Mon 25 Oct 2021

PDSP Assignment 1, due Tue 2 Nov 2021

PDSP Assignment 2, due Fri 12 Nov 2021

Practice Assignment 1

Due on 2021-12-31, 23:59 IST

Write four Python functions as specified below. Copy and paste the text for all four functions together into the submission window. Your function will be called automatically with various inputs and should return values as specified. Do not write commands to read any input or print any output.

- You may define additional auxiliary functions as needed.
- In all cases you may assume that the value passed to the function is of the expected type, so your function does not have to check for malformed inputs.
- For each function, there are normally some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases. There are 20 private test cases, with equal weightage. You will get feedback about which private test cases pass or fail, though you cannot see the actual test cases.
- Ignore warnings about "Presentation errors".
- You can submit as many times as you like. Your final submission will be used for scoring.

1. Function: `sumofsquares(n)`

   A positive integer `n` is a sum of squares if `n == i`^{2} `+ j`^{2} for integers `i,j` such that `i ≥ 1` and `j ≥ 1`. For instance, `10` is a sum of squares because `10 == 1`^{2} `+ 3`^{2}, and so is `25` (`3`^{2} `+ 4`^{2}). On the other hand, `11` and `3` are not sums of squares.

   Write a function `sumofsquares(n)` that takes a positive integer argument and returns `True` if the integer is a sum of squares, and `False` otherwise.

   Here are some examples to show how your function should work.

```
>>> sumofsquares(2)
True

>>> sumofsquares(11)
False

>>> sumofsquares(3218)
True

>>> sumofsquares(3219)
False
```

2. Function: `shuffle(l1,l2)`

   Write a function `shuffle(l1,l2)` that takes as input two lists, `11` and `l2`, and returns a list consisting of the first element in `l1`, then the first element in `l2`, then the second element in `l1`, then the second element in `l2`, and so on. If the two lists are not of equal length, the remaining elements of the longer list are appended at the end of the shuffled output.

   Here are some examples to show how your function should work.

```
>>> shuffle([0,2,4],[1,3,5])
[0, 1, 2, 3, 4, 5]

>>> shuffle([0,2,4],[1])
[0, 1, 2, 4]

>>> shuffle([0],[1,3,5])
[0, 1, 3, 5]
```

3. Function: `removeduplicates(l)`

   Write a function `duplicates(l)` that finds all duplicate values in `l`. Your function should retain the first copy of each duplicate value — in other words, you should preserve the order in which values appear in `l`.

   Duplicates could be simple values such as numbers or complex values such as strings and lists. You need to find duplicates only at the top level of the list – if a list contains another list or string as an element with duplicate values, the nested list or string can remain as it is.

   Your function should return a new list of duplicates. It should not modify the input list `l`.

   Here are some examples to show how your function should work.

```
>>> duplicates([7,2,5,7,2,9])
[7, 2]

>>> duplicates([7,2,"hello",2,[5,5],"hello",9,[5,5]])
[2, 'hello', [5, 5]]
```

4. Function: `splitwith(l,x)`

   Write a function `splitwith(l,x)` that uses `x` as a separator to split the list `l` into sublists. Your function should return the list of sublists generated by splitting `l` in this way: all occurrences of `x` should be omitted. In other words, if your function returns the list of lists `[l1,l2,…,lk]` then it should be the case that

   ```
   l == l1 + [x] + l2 + [x] + ... + [x] + lk
   ```

   - p If there is no occurrence of `x` in `l`, the value returned should be list consisting of a single inner list which is the entire input list `l`.
   - If `x` occurs at the first/last position of the list, one of the sublists (before/after `x`, respectively) will be the empty list.
   - Likewise, if there are two consecutive occurrences of `x` in `l`, they will be separated by the empty list.

   Here are some examples to show how your function should work.

   ```
   >>> splitwith([7,2,5,7,7,[2,7],9],7)
   [[], [2, 5], [], [[2, 7], 9]]

   >>> splitwith([7,2,5,7,[2,7],9],8)
   [[7, 2, 5, 7, [2, 7], 9]]

   >>> splitwith([7,2,[8],8,[8],[[8],7],2,9],[8])
   [[7, 2], [8], [[[8], 7], 2, 9]]

   >>> splitwith([1,2,1],1)
   [[], [2], []]
   ```

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status

Test Case 1	`sumofsquares(15379)`	`False\n`	`NA`	Time Limit Exceeded
Test Case 2	`sumofsquares(29684)`	`True\n`	`True\n`	Passed
Test Case 3	`sumofsquares(26650)`	`True\n`	`True\n`	Passed
Test Case 4	`sumofsquares(29371)`	`False\n`	`NA`	Time Limit Exceeded
Test Case 5	`sumofsquares(29682)`	`True\n`	`True\n`	Passed
Test Case 6	`shuffle([100,200,300],[])`	`[100, 200, 300]\n`	`[100, 200, 300]\n`	Passed
Test Case 7	`shuffle([],[10,20,30])`	`[10, 20, 30]\n`	`[10, 20, 30]\n`	Passed
Test Case 8	`shuffle([[1],2,[3]],[[4,[5]],6])`	`[[1], [4, [5]], 2, 6, [3]]\n`	`[[1], [4, [5]], 2, 6, [3]]\n`	Passed
Test Case 9	`shuffle([[1,2,3]],[[4,5,6]])`	`[[1, 2, 3], [4, 5, 6]]\n`	`[[1, 2, 3], [4, 5, 6]]\n`	Passed
Test Case 10	`shuffle([[[1,2,3]]],[[4,5,6]])`	`[[[1, 2, 3]], [4, 5, 6]]\n`	`[[[1, 2, 3]], [4, 5, 6]]\n`	Passed
Test Case 11	`duplicates([71,22,57,22,71,99])`	`[71, 22]\n`	`[71, 22]\n`	Passed
Test Case 12	`duplicates([30,20,10,10,20,30])`	`[30, 20, 10]\n`	`[30, 20, 10]\n`	Passed
Test Case 13	`duplicates([301,201,101,101,301,201])`	`[301, 201, 101]\n`	`[301, 201, 101]\n`	Passed
Test Case 14	`duplicates([[301],201,301,[201]])`	`[]\n`	`[]\n`	Passed
Test Case 15	`duplicates(["[301]",[201],201,"301","[201]",301,201])`	`[201]\n`	`[201]\n`	Passed

Test Case 16	splitwith([22,75,"hello",11,[51],"hello",9,[51]],[51])	[[22, 75, 'hello', 11], ['hello', 9], []]\n	[[22, 75, 'hello', 11], ['hello', 9], []]\n	Passed
Test Case 17	splitwith([7,"2",5,7,7,["2",7],9],"2")	[[7], [5, 7, 7, ['2', 7], 9]]\n	[[7], [5, 7, 7, ['2', 7], 9]]\n	Passed
Test Case 18	splitwith([71,2,5,7,[2,7],9],83)	[[71, 2, 5, 7, [2, 7], 9]]\n	[[71, 2, 5, 7, [2, 7], 9]]\n	Passed
Test Case 19	splitwith([7,2,[],8,[8],[],[[],7],2,9],[])	[[7, 2], [8, [8]], [[[], 7], 2, 9]]\n	[[7, 2], [8, [8]], [[[], 7], 2, 9]]\n	Passed
Test Case 20	splitwith([11,22,11,11,22,11],11)	[[], [22], [], [22], []]\n	[[], [22], [], [22], []]\n	Passed

The due date for submitting this assignment has passed.
18 out of 20 tests passed.
You scored 90.0/100.

Assignment submitted on 2021-10-16, 08:24 IST

Your last recorded submission was :

```python
def sumofsquares(n):
    for i in range(1,n):
        for j in range(1,n):
            sqSum = i**2 + j**2
            if sqSum == n:
                return True
    return False

def shuffle(l1,l2):
    newList = []
    for i in range(min(len(l1),len(l2))):
        newList.append(l1[i])
        newList.append(l2[i])
    if (len(l1)<len(l2)):
        newList.extend(l2[len(l1):])
    elif (len(l1)==len(l2)):
        return newList
    else:
        newList.extend(l1[len(l2):])
    return newList

def duplicates(l):
    dup = []
    for i in range(len(l)):
        if l[i] in l[i+1:]:
            if l[i] not in dup:
                dup.append(l[i])
    return dup

```

```python

def splitwith(l,x):
    splitList1 = []
    splitList2 = l[:]
    i = 0
    while x in splitList2:
        while i<len(splitList2):
            if splitList2[i] != x:
                i = i+1
            else:
                splitList1.append(splitList2[:i])
                splitList2 = splitList2[(i+1):]
                i = 0
                break
        splitList2 = splitList2[:]
    if len(splitList2) !=0:
        splitList1.append(splitList2)
    if l[(len(l)-1)] == x:
        splitList1.append([])
    return splitList1
import ast

def tolist(inp):
  inp = "["+inp+"]"
  inp = ast.literal_eval(inp)
  return (inp[0],inp[1])

def parse(inp):
  inp = ast.literal_eval(inp)
  return (inp)

fncall = input().strip()
lparen = fncall.find("(")
rparen = fncall.rfind(")")
fname = fncall[:lparen]
farg = fncall[lparen+1:rparen]

if fname == "sumofsquares":
  arg = parse(farg)
  print(sumofsquares(arg))
elif fname == "shuffle":
  (l1,l2) = parse(farg)
  print(shuffle(l1,l2))
elif fname == "duplicates":
  l = parse(farg)
  l1 = l[:]
  dl = duplicates(l)
  if (l1 == l):
    print(dl)
  else:
    print("Side effect")
elif fname == "splitwith":
  (l,x) = parse(farg)
  print(splitwith(l,x))
else:
  print("Function", fname, "unknown")

```

Sample solutions (Provided by instructor)

```python
### Question 1

from math import *

def sumofsquares(n):
    for i in range(1,n//2+1):
        if issquare(i) and issquare(n-i):
            return(True)
    return(False)

def issquare(n):
    m = int(sqrt(n))
    return (n == m*m)

### Question 2

def shuffle(l1,l2):
    myl1 = l1[:]
    myl2 = l2[:]
    newlist = []
    while myl1 != [] and myl2 != []:
        newlist.append(myl1[0])
        newlist.append(myl2[0])
        myl1 = myl1[1:]
        myl2 = myl2[1:]

    if myl1 != []:
        newlist.extend(myl1)
    elif myl2 != []:
        newlist.extend(myl2)

    return(newlist)

### Question 3

def duplicates(l):
    duplist = []
    for i in range(len(l)):
        if l[i] not in duplist:
            if l[i] in l[i+1:]:
                duplist.append(l[i])
    return(duplist)

### Question 4

def splitwith(l,x):
    newlist = []
    nextblock = []
    for y in l:
        if y == x:
            newlist.append(nextblock)
            nextblock = []
        else:
            nextblock.append(y)
    newlist.append(nextblock)
    return(newlist)


import ast

def tolist(inp):
  inp = "["+inp+"]"
  inp = ast.literal_eval(inp)
  return (inp[0],inp[1])

def parse(inp):
  inp = ast.literal_eval(inp)
  return (inp)

fncall = input().strip()
lparen = fncall.find("(")
rparen = fncall.rfind(")")
fname = fncall[:lparen]
farg = fncall[lparen+1:rparen]

if fname == "sumofsquares":
  arg = parse(farg)
  print(sumofsquares(arg))
elif fname == "shuffle":
  (l1,l2) = parse(farg)
  print(shuffle(l1,l2))
elif fname == "duplicates":
  l = parse(farg)
  l1 = l[:]
  dl = duplicates(l)
  if (l1 == l):
    print(dl)
  else:
    print("Side effect")
elif fname == "splitwith":
  (l,x) = parse(farg)
  print(splitwith(l,x))
else:
  print("Function", fname, "unknown")

```