20200114_solution

February 3, 2020

1 Solutions of Exam 2020-01-14

1.1 Part A

1.1.1 A1 multiple choice

1. B
2. C
3. E, [1, 2, 1, 2, 1, 2, 3, 4, 5, 3, 4, 5]
4. A
5. B
6. A
7. B
8. C

1.1.2 Some examples are shown below

[1]:
```
# (a) and (b)
def func(y):
    if y == 2:
        print(y)
    elif y > 2:
        print(y)
    else:
        print(y)

func(2.0)
func(3)
```

```
2.0
3
```

[2]:
```
# (c)
L1 = [1, 2]
L2 = [3, 4, 5]
print(L1*3+L2*2)
```

```
[1, 2, 1, 2, 1, 2, 3, 4, 5, 3, 4, 5]
```

[3]:
```python
# (d)
with open('exam.dat', 'r') as file:
    read_data = file.read()
    print(read_data)
    read_data = file.read()
    print(read_data)
```

```
1
```

[4]:
```python
# (e)
x = [3, 7, 8]
y = x
y.append(10)
print(x)
```

```
[3, 7, 8, 10]
```

[5]:
```python
# (f)
a = list(range(10))
print(a[1::3])
```

```
[1, 4, 7]
```

[6]:
```python
# (h)
print('i' in 'This is a string', 'i' in ['This', 'is', 'a', 'string'])
```

```
True False
```

1.1.3 A2

[7]:
```python
y = 0
while y < 30:
    # Remove the next line to get rid of the infinite loop
    # y = 1
    print(f'{y}', end=',')
    y = y + 2
```

```
0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,
```

1.1.4 A3

[8]:
```
print('First row')
def printRows(text):
    print('Second row')
    print(text)
print('Third row')
printRows('Last row')
```

```
First row
Third row
Second row
Last row
```

1.1.5 A4

[9]:
```
a = list(range(10))
print(f'The first element in the list is {a[0]}')
print(f'The sum of the list is {sum(a)}')
print(f'The mean of the list is {sum(a)/10}')
print(f'The last element in the list is {a[-1]}')
# The list has 10 elements and its indices go from 0 to 9. Hence a[10] asks
# for an element outside the list. Using a[-1] returns the last element␣
 ↪regardless
# the size of the list.
```

```
The first element in the list is 0
The sum of the list is 45
The mean of the list is 4.5
The last element in the list is 9
```

1.1.6 A5

There are many possible solutions:

[10]:
```
def rep(lst, x):
    for i in range(len(lst)):
        lst[i] = x

def rep_2(lst, x):
    for i,k in enumerate(lst):
        lst[i] = x

# This function will work, but not a good approach
def rep_3(lst, x):
    for i in range(len(lst)):
```

```
        lst.pop(0)
        lst.append(x)
```

[11]:
```
a = [1, 3, 7]
rep(a, 3)
print(a)
```

```
[3, 3, 3]
```

[12]:
```
a = [1, 3, 7]
rep_2(a, 3)
print(a)
```

```
[3, 3, 3]
```

[13]:
```
a = [1, 3, 7]
rep_3(a, 3)
print(a)
```

```
[3, 3, 3]
```

Some wrong examples:

[14]:
```
def rep_wrong(lst, x):
    lst = [x for i in lst]
    return lst

a = [1, 3, 7]
rep_wrong(a, 3)
print(a)
```

```
[1, 3, 7]
```

[15]:
```
def rep_wrong(lst, x):
    for i in range(len(lst)):
        lst[i-1] = x
    return lst


a = [1, 3, 7]
rep_wrong(a, 3)
print(a)
```

```
[3, 3, 3]
```

1.1.7 A6

[16]:
```
# (a)
with open('exam.txt') as f:
    wordList = f.read().split()

wordCount = len(wordList)
```

[17]:
```
# (b)
wordDict = {}
for word in wordList:
    if word in wordDict:
        wordDict[word] += 1
    else:
        wordDict[word] = 1
```

[18]:
```
# (c)
def filterWords(wordDict, n):
    return {word: count
            for word, count in wordDict.items()
            if count < n}
```

1.1.8 A7

An example solution:

[19]:
```
class City:
    def __init__(self, name='', pop=0, zCode='00', x=0, y=0):
        self.name = name
        self.pop = pop
        self.zCode = zCode
        self.x = x
        self.y = y
    def __str__(self):
        info = f'The city {self.name} has {self.pop} population, the zip code is␣
↪{self.zCode}XXX and the GPS coordinates is ({self.x}, {self.y}).'
        return info
```

To test the class `City`:

[20]:
```
Uppsala = City('Uppsala', 180000, '75', 59, 17)
print(Uppsala)
```

```
The city Uppsala has 180000 population, the zip code is 75XXX and the GPS
coordinates is (59, 17).
```

1.2 Part B

1.2.1 B8

An example:

[21]:
```
def Bsort(lst):
    for i in range(len(lst)):
        minx = lst[i]
        tempJ = i
        for j, x in enumerate(lst[i:]):
            if x < minx:
                minx = x
                tempJ = j + i
        lst[i], lst[tempJ] = lst[tempJ], lst[i]
    return lst
arr = [64, 25, 12, 22, 11]
Bsort(arr)
print(arr)
```

```
[11, 12, 22, 25, 64]
```

1.2.2 B9

(a) Convert to 1D array

[22]:
```
def convertTo1D(matA):
    return [x for rows in matA for x in rows]

matA = [[1, 3, 5],[2, 7, 8]]
matA_1D = convertTo1D(matA)
print(matA_1D)
```

```
[1, 3, 5, 2, 7, 8]
```

(b) Convert to column representation

[23]:
```
def convertToCol(matA):
    return [[matA[j][i] for j in range(len(matA))] for i in range(len(matA[0]))]

matA_CW = convertToCol(matA)
print(matA_CW)
```

```
[[1, 2], [3, 7], [5, 8]]
```

1.2.3 B10

An example:

[24]:
```
maplist = [('Stockholm', 'Uppsala'), ('Uppsala', 'Gävle'), ('Stockholm',␣
↪'Vasteras')]

def convert(oldList):
    mapdict = {}
    for pair in oldList:
        for i in range(2):
            if pair[i] in mapdict.keys():
                mapdict[pair[i]].append(pair[1-i])
            else:
                mapdict[pair[i]] = [pair[1-i]]
    return mapdict

mapd = convert(maplist)
print(mapd)
```

```
{'Stockholm': ['Uppsala', 'Vasteras'], 'Uppsala': ['Stockholm', 'Gävle'],
'Gävle': ['Uppsala'], 'Vasteras': ['Stockholm']}
```

1.2.4 B11

An example:

[25]:
```
def convole(f, g):
    return [sum([f[n-m]*g[m] for m in range((len(f))) if n-m<len(f) and n-m>=0])␣
↪for n in range(len(g))]
```