PYTHON ASSIGNMENT SOLUTIONS

DATA STRUCTURE

LISTS:

1.fruits = ['Banana','Apple','Date','Banana','Apple','Pear','Orange']

print(fruits.count('Apple'))

Answer=2;

2. print(fruits.count('Kiwi'))

Answer=0;

3.print(fruits.index('Apple'))

Answer=1;

4.print(fruits.index('Apple',4))

Answer=4;

5.fruits.reverse()

print (fruits)

Answer=['Orange', 'Pear', 'Apple', 'Banana', 'Date', 'Apple', 'Banana'];

6.fruits.append('Peach')

print (fruits)

Answer=['Banana', 'Apple', 'Date', 'Banana', 'Apple', 'Pear', 'Orange', 'Peach'];

7. fruits.sort()

print (fruits)

Answer=['Apple', 'Apple', 'Banana', 'Banana', 'Date', 'Orange', 'Peach', 'Pear'];

8.fruits.pop()

print (fruits)

Answer=['Apple', 'Apple', 'Banana', 'Banana', 'Date', 'Orange', 'Peach'];

USING LISTS AS STACKS:

1.stack=[3,4,5]

stack.append(6)

stack.append(7)

print(stack)

Answer=[3, 4, 5, 6, 7];

2.(i)stack.pop()

print(stack)

Answer=[3, 4, 5, 6];

. (ii)stack.pop()

print(stack)

Answer=[3, 4, 5];

. (iii)stack.pop()

print(stack)

Answer=[3, 4];

USING LISTS AS QUEUES:

1.from collections import deque

queue=deque(['ARAFAT','BOSHIR','MIKE'])

queue.append('MARK')

print(queue)

Answer=deque(['ARAFAT', 'BOSHIR', 'MIKE', 'MARK'])

2.(i)queue.append('KEVIN')

print(queue)

Answer=deque(['ARAFAT', 'BOSHIR', 'MIKE', 'MARK', 'KEVIN'])

(ii)queue.popleft()

print(queue)

Answer=deque(['ARFAT', 'BOSHIR', 'MIKE', 'MARK', 'KEVIN'])

(ii)queue.popleft()

print(queue)

Answer=deque(['MIKE', 'MARK', 'KEVIN'])

LIST COMPREHENSION:

1.lists = [(x,y)for x in [1,2,3] for y in [3,1,4] if x!=y]

print(lists)

Anwer=[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)];

2.squares = []

for x in range(10):

squares.append(x\*\*2)

print(squares)

Answers=[0, 1, 4, 9, 16, 25, 36, 49, 64, 81];

3.(i)vec = [-4,-2,0,2,4]

Doubeled=[x\*2 for x in vec]

print(Doubeled)

Answer=[-8, -4, 0, 4, 8];

(ii)vec = [-4,-2,0,2,4]

filt = [x for x in vec if x>=0]

print(filt)

Answer=[0, 2, 4];

(iii)vec = [-4,-2,0,2,4]

fun = [abs(x) for x in vec]

print(fun)

Answer=[4, 2, 0, 2, 4]

4.freshfruits=['Banana','Strawberrry','Apple']

method=[weapon.strip() for weapon in freshfruits]

print(method)

Answer=['Banana', 'Strawberrry', 'Apple']

5.tup\_list=[(x,x\*\*2) for x in range(8)]

print(tup\_list)

Answer=[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49)]

6.vec=[[1,2,3],[4,5,6],[7,8,9]]

num=[num for elem in vec for num in elem]

print(num)

Answer=[1, 2, 3, 4, 5, 6, 7, 8, 9]

7.from math import pi

enu=[str(round(pi,i)) for i in range (1,6)]

print(enu)

Answer=['3.1', '3.14', '3.142', '3.1416', '3.14159'];

NESTED LOOP COMPREHENSION:

1.matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

transpose =[[row[i]for row in matrix] for i in range(4)]

print(transpose)

Answer=[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]

2.matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

T= list (zip(\*matrix))

print(T)

Answer=[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]

THE DEL STATEMENT:

1.(i)a = [-1, 1, 66.25, 333, 333, 1234.5]

del a[0]

print(a)

Answer= a = [ 1, 66.25, 333, 333, 1234.5]

(ii)a = [-1, 1, 66.25, 333, 333, 1234.5]

del a[2:4]

print(a)

Answer=[-1, 1, 333, 1234.5]

(iii)a = [-1, 1, 66.25, 333, 333, 1234.5]

del a[:]

print(a)

Answer= [];

TUPPLES AND SEQUENCES:

1.t = 12345, 54321, 'hello!'

print (t)

Answer=(12345, 54321, 'hello!')

2.t = 12345, 54321, 'hello!'

print (t[0])

Answer= 12345

3.t = 12345, 54321, 'hello!'

u=t,(1,2,3,4,5)

print (u)

Answer=((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))

4.v = ([1, 2, 3], [3, 2, 1])

print(v)

Answer=([1, 2, 3], [3, 2, 1])

5.(i)empty = ()

print(len(empty))

Answer=0;

(ii)singleton = 'hello',

print(len(singleton))

Answer=1;

(iii)singleton = 'hello',

print(singleton)

Answer=('hello',);

SETS

1.basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}

print(basket)

Answer={'orange', 'banana', 'pear', 'apple'};

2.(i)a = set('abracadabra')

b = set('alacazam')

print(a)

Answer={'a', 'r', 'b', 'c', 'd'};

(ii)print((a-b))

Answer={'d', 'b', 'r'};

(iii)print((a|b))

print((a&b))

print((a^b))

Answers={'r', 'a', 'z', 'b', 'c', 'm', 'd', 'l'}

{'c', 'a'}

{'z', 'b', 'm', 'd', 'l', 'r'}

(iv)a={x for x in 'abracadabra' if x not in 'abc'}

print (a)

Answer={'d', 'r'};

DICTIONARIES

1. (i)tel = {'jack': 4098, 'sape': 4139}

tel['guido'] = 4127

print(tel)

Answer=

{'jack': 4098, 'sape': 4139, 'guido': 4127};

(ii)print((tel['jack']))

Answer=4088;

(iii)tel = {'jack': 4098, 'sape': 4139}

tel['guido'] = 4127

del tel['sape']

tel['irv'] = 4127

print(tel)

Answer={'jack': 4098, 'guido': 4127, 'irv': 4127}

(iv) print(list(tel))

['jack', 'guido', 'irv']

print(sorted(tel))

['guido', 'irv', 'jack']

(v)print('jack' in tel)

TRUE

print('Mark' in tel)

FALSE

2.(i)a=dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])

print(a)

Answer={'sape': 4139, 'guido': 4127, 'jack': 4098};

(ii)a={x: x\*\*2 for x in (2, 4, 6)}

print(a)

Answer={2: 4, 4: 16, 6: 36};

(iii)a=dict(sape=4139, guido=4127, jack=4098)

print(a)

Answer={'sape': 4139, 'guido': 4127, 'jack': 4098};

LOOPING TECHNIQUES

1.(i)knights={'gallahad':'the pure','robin':'the brave'}

for k,v in knights.items():

print((k,v))

Answer=gallahad the pure

robin the brave

(ii)for i,v in enumerate (['tic','tac','toe']):

print(i,v)

Answer=0 tic

1 tac

2 toe

2.(i)questions=['name','quest','favouritecolour']

answers=['ARAFAT','the holy grail','Red']

for q,a in zip(questions,answers):

print('Whats your{0}?it is {1}.'.format(q,a))

Answers=Whatsyourname?it is ARAFAT.

Whatsyourquest?it is the holy grail.

Whatsyourfavouritecolour?it is Red.

(ii)for i in reversed(range(1,10,2)):

print(i)

Answers=9

7

5

3

1

3.(i)basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']

for f in sorted(set(basket)):

print(f)

Answer=apple

banana

orange

pear

(ii)import math

raw\_data = [56.2, float('NaN'), 51.7, 55.3, 52.5, float('NaN'), 47.8]

filtered\_data = []

for value in raw\_data:

if not math.isnan(value):

filtered\_data.append(value)

print(filtered\_data)

Answers=[56.2, 51.7, 55.3, 52.5, 47.8];

MORE ON CONDITIONS

1.string1, string2, string3 = '', 'Trondheim', 'Hammer Dance'

non\_null = string1 or string2 or string3

print(non\_null)

Answers=Trondheim;

COMPARING SEQUENCES AND OTHER TYPES

a=(1, 2, 3) < (1, 2, 4)

b=[1, 2, 3] < [1, 2, 4]

c='ABC' < 'C' < 'Pascal' < 'Python'

d=(1, 2, 3, 4) < (1, 2, 4)

e=(1, 2) < (1, 2, -1)

f=(1, 2, 3) == (1.0, 2.0, 3.0)

g=(1, 2, ('aa', 'ab')) < (1, 2, ('abc', 'a'), 4)

print(a,b,c,d,e,f,g)

Answers=True TrueTrueTrueTrueTrueTrue;