Topic 10: Question 5

Construct a class named 'Point', which takes in the x and y coordinates as parameters. It should include the documentation string and the methods '\_\_init\_\_' and '\_\_str\_\_'.

>>> P1 = Point(2,4)

>>> P1.\_\_doc\_\_

'A class implementation of a 2-dimensional point.'

>>> str(P1)

'(2, 4)'

>>> **print** P1

(2, 4)

class Point:

def \_\_init\_\_(self, ):

def \_\_str\_\_(self):

Topic 10: Question 3

To create an object, it is important to look at the 'constructor' method: \_\_init\_\_() to find out the arguments that are required. The example below shows you how to create a Laptop object. Create a Person object based on the class definition given in the textbox below.

**Examples**

>>> **class** **Laptop**:

**def** \_\_init\_\_(self, weight, color):

self.weight = weight

self.color = color

>>> laptop = Laptop(1.2, "Silver") *#Create a Laptop object*

>>> laptop = Laptop()

Traceback (most recent call last):

File "<pyshell#6>", line 1, **in** <module>

laptop = Laptop()

**TypeError**: \_\_init\_\_() takes exactly 3 arguments (1 given)

class Person:

def \_\_init\_\_(self, weight, height):

self.weight = weight

self.height = height

# Create a Person object with weight = 60, height = 1.7

p =Person(60,1.7)

Topic 10: Question 7

It is possible to change the behaviour of built-in opertators with special methtods. For example, the '+' operator can be implemented with the \_\_add\_\_ method. Define a Point class that supports operator overloading for the '+' and '-' operators.

**Examples**

>>> a = Point(1,3)

>>> b = Point(7,2)

>>> **print** a+b

'(8, 5)'

>>> **print** a-b

'(-6, 1)'

class Point:

"A class implementation of 2-Dimensional point."

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def \_\_str\_\_(self):

return '(%d, %d)' % (self.x, self.y)

def \_\_add\_\_(self, other):

x=self.x + other.x

y=self.y+ other.y

return Point(x,y)

def \_\_sub\_\_(self, other):

x=self.x - other.x

y=self.y- other.y

return Point(x,y)

Topic 10: Question 8

Inheritance allows the reuse of code, by implementing a parent-child relationship between classes. Create 2 derived classes Student and WorkingAdult from the base class Person.

**Examples**

>>> s = Student('Peter', 9, 'ABC Primary School')

>>> s.introduce()

'My name is Peter. I am 9 years old. I am studying at ABC Primary School.'

>>> a = WorkingAdult('John', 23, 'Waiter')

>>> a.introduce()

'My name is John. I am 23 years old. I am a waiter.'

class Person:

"""A base class"""

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

class Student(Person):

"""A derived class for Student"""

def \_\_init\_\_(self, name, age, school):

Person.\_\_init\_\_(self, name, age)

def introduce(self):

class WorkingAdult(Person):

"""A derived class for WorkingAdult"""

def \_\_init\_\_(self, name, age, job):

def introduce(self):

Topic 14: Question 1

**range(***[start,] stop[, step]***)** Function to create lists containing arithmetic progressions. *start* is zero if omitted. *step* has a default value of 1 and must be non-zero. If *step* is positive, the upper limit is *stop*-1. If *step* is negative, the lower limit is *stop*+1.

**Examples**

>>> range(10) *# only 'stop' is specified.*

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> range(5, 10) *# 'start', 'stop' are specified.*

[5, 6, 7, 8, 9 ]

>>> range(1, 10, 2) *# 'start', 'stop', 'step' are specified.*

[1, 3, 5, 7, 9]

>>> range(10, 1, -2) *# with negative 'step'.*

[10, 8, 6, 4, 2]

>>> createLists(5)

([1, 2, 3, 4, 5], [5, 4, 3, 2, 1], [ -5, -4, -3, -2, -1], [-1, -2, -3, -4, -5])

# Write a function that returns 4 lists given a postive number.

def createLists(num):

Topic 14: Question 2

**abs(***x***)** Return the absolute value of a number or the magnitude of a complex number.

**Examples**

>>> abs(-1) *# negative integer*

1

>>> abs(2.0) *# floating-point number*

2.0

>>> abs(3+4j) *# complex number*

5.0

>>> diff(3+4j, -6)

1.0

>>> diff(-5, 6)

1

# Write a function that returns the output as shown in the above examples.

def diff(a, b):

return -abs(a)+abs(b)

Topic 14: Question 3

**chr(***i***)** Return a character whose ASCII code is the integer *i*. *i* is between 0 and 255 inclusive.

**Examples**

>>> chr(97) *# ASCII code 97-122 corresponds to 'a-z'*

'a'

>>> chr(90) *# ASCII code 65-90 corresponds to 'A-Z'*

'Z'

>>> chr(48) *# ASCII code 48-57 corresponds to '0-9'*

'0'

>>> chr(256)

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

**ValueError**: chr() arg **not** **in** range(256)

>>> toString([112, 121, 115, 99, 104, 111, 111, 108, 115])

'pyschools'

# Write a function that returns a string of characters based on a list of ASCII codes.

def toString(alist):

a=""

for i in alist:

b=chr(i)

a+=b

return a

Topic 14: Question 4

**ord(***c***)** Return the ASCII code of a character. **ord** is the opposite of **chr**.

**Examples**

>>> ord('a')

97

>>> chr(ord('a)-32) # convert 'a' to 'A' by subtracting 32 from ASCII code

'A'

>>> ord('2')

50

>>> ord(1)

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

**TypeError**: ord() expected string of length 1, but int found

>>> capitalize('how are you?')

'How Are You?'

# Write a function that capitalizes the first character of each word.

def capitalize(phrase):

Topic 14: Question 6

**min(***iterable[, args...][, key]***)** Return the minimum item of a non-empty *iterable* or *args*. Optional *key* is a function to convert each item before doing comparision.  
**max** works like **min** but returns the item with maximum value.

**Examples**

>>> aList = [4, -5, 7 ]

>>> min(aList) *# items contained in iterable*

-5

>>> min(3, 2, 5) *# items are the arguments*

2

>>> min(-2, -1, 0)

-2

>>> min(-2, -1, 0, key=abs) *# key function is specified*

0

>>> mixedList([0, '-2', '4', '13', 3])

(-2, 13)

# Write a function that returns minimum and maximum values of a list containing numbers in integer and string formats.

def mixedList(mlist):

mlist = [int(x) for x in mlist]

return min(mlist), max(mlist)

Topic 14: Question 8

**hex(***x***)** Convert an integer number to a hexadecimal string.  
**oct(***x***)** Convert an integer number to a octal string.

**Examples**

>>> a = 100

>>> hex(a)

'0x64'

>>> oct(a)

'0144'

>>> int(hex(a), 16)

100

>>> int(oct(a), 8)

100

>>> dec2hexoct(64)

('0x40', '0100')

# Write a function that converts a decimal integer to both hexadecimal and octal format.

def dec2hexoct(x):

return (hex(x),oct(x))

Topic 14: Question 9

**map(***function, iterable, ...***)** Apply *function* to every item of *iterable* and return a list.

**Examples**

>>> map(ord, 'abcde') *# convert each character to ASCII code using 'ord'*

[97, 98, 99, 100, 101]

>>> map(float, ['1.3', '1', '2') *# convert to floating point numbers*

[1.3, 1.0, 2.0 ]

>>> **def** add(x, y):

... **return** x+y

...

>>> map(add, [1, 2, 3], [4, 5, 6]) *# function is 'add'*

[5, 7, 9]

>>> map(**lambda** x,y:x+y, [1, 2, 3], [4, 5, 6]) *# using lambda*

[5, 7, 9]

>>> mapfn1(range(10, 12)] *# convert to hex*

['0xa', '0xb', '0xc']

>>> mapfn2(range(10)] *# modulo 2*

[0, 1, 0, 1, 0, 1, 0, 1, 0, 1]

>>> mapfn3(['pyschools']) *# convert to uppercase*

['P', 'Y', 'S', 'C', 'H', 'O', 'O', 'L', 'S']

# Complete the code below so that the outputs are as shown in the examples above.

def mapfn1(alist):

return list(map( hex , alist))

def mapfn2(alist):

return list(map(lambda x: x%2 , alist))

def mapfn3(word):

return list(map(lambda x: x.upper() , word) )

Topic 14: Question 10

**filter(***function, iterable***)** Construct a list from items of *iterable* for which *function* returns True.

**Examples**

>>> filter(**lambda** x:x%2==0, range(10)) *# generate a list of even integers.*

[0, 2, 4, 6, 8]

>>> filter(**lambda** x:x>0, [5, -2, 8, 1, -1]) *# return a list of positive integers*

[5, 8, 1]

>>> **def** ones(x):

... **if** '1' **in** str(x):

... **return** True

... **else**:

... **return** False

>>> filter(ones, range(30)) *# return a list of integers with digit '1'.*

[1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21 ]

>>> fn1('aBCDefG')

'aef'

>>> fn2('13a@b24&z')

'1324'

# Complete the code below so that the outputs are as shown in the examples above.

def lowercase(x):

if x.islower():

return True

else:

return False

def asf(x):

if x.isnumeric():

return True

else:

return False

def fn1(word):

return filter(lowercase, word)

def fn2(word):

return filter(lambda x: ord(x)>47 and ord(x)<58 , word)

Topic 14: Question 11

**reduce(***function, iterable[, initializer]***)** Apply function of two arguments cumulatively to the items of *iterable*, reducing the *iterable* to a single value.

**Examples**

>>> reduce(**lambda** x,y:x+y, range(5)) *# sum up numbers from 0 to 4 inclusive i.e. (0+1+2+3+4)*

10

>>> reduce(**lambda** x,y:x+y, range(5), 10) *# with initializer set to 10.*

20

>>> factorial(4)

24

>>> factorial(0)

1

# Write a factorial function using the 'reduce' function

def factorial(num):

start = num

if num==0:

return 1

return reduce(lambda x,y:x\*y ,range( 1,num ), start)

Topic 14: Question 14

**list(***[iterable]***)** Return a list whose items are the same and in the same order as *iterrable*'s items.

**Examples**

>>> list() *# an emtpy list*

[]

>>> list('xyz')

['x', 'y', 'z']

>>> list((1, 2, 3)) *# convert tuple to list*

[1, 2, 3]

>>> list(dict(zip('123', 'abc'))) *# generate a list from keys of dictionary.*

['1', '3', '2' ]

>>> map(list, zip(list(b), list(c)))

[ ['g','e'], ['o', 'n'], ['o', 'g'], ['g', 'i'], ['l', 'n'], ['e', 'e']]

# Write down the values of variables b and c, with the output as shown in the above example.

b ='google'

c = ('e','n','g','i','n','e')

Topic 14: Question 16

**len(***s***)** Return the length (the number of items) of a sequnce (string, tuple or list) or a mapping (dictionary).

**Examples**

>>> len('python') *# string*

6

>>> len([]) *# empty list*

0

>>> len((1, 2, 3, 4)) *# tuple*

4

>>> len({'a':1, 'b':2, 'c':3}) *# dictionary*

3

>>> totSize('abc', (1,), [1,2,3])

7

# Write a function that returns the total size of the arguments.

# Note: \*args denotes a variable argument list, represented by a tuple.

def totSize(\*args):

counter=0

for i in args:

a=len(i)

counter+=a

return counter

Topic 14: Question 17

**sum(***[iterable[, start]***)** Return the total of the items in an *iterable*. *start* is the initial value, and is zero by default.

**Examples**

>>> sum(range(10)) *# total from 0 to 9*

45

>>> sum(range(10), 5) *# total from 0 to 9, with initial value of 5*

50

>>> totalSum(range(10), range(5))

55

>>> totalSum([1, 2, 3], (3, 4))

13

>>> totalSum([1, 2, 3], {1:3, 2:4})

9

# Write a function that returns the total of two sequences.

def totalSum(a, b):

Topic 14: Question 18

**divmode(***a, b***)** Return the quotient and remainder of a/b.  
For integers, result is the same as (*a* // *b*, *a* % *b*).

**Examples**

>>> divmod(3, 2)

(1, 1)

>>> divmod(8, -2)

(-4, 0)

>>> exponent(8, 2)

3

>>> exponent(25, 5)

2

# Write a function that returns the exponent of a positive number given its base.

def exponent(num, base):

exp = 0

while divmod(num, )[ ]: # loop while the quotient is non-zero

return exp

Topic 14: Question 19

**all(***iterable***)** Return True if all elements of the *iterable* are true.  
**any(***iterable***)** Return True if any element of the *iterable* is true. If the *iterable* is empty, return False.

**Examples**

>>> all([0, 1, 2]) *# 0 is False*

False

>>> any([0, 1, 2]) *# 1 and 2 are True*

True

>>> checkItems([1, 2, 3])

'All are true.'

>>> checkItems([{'a':1}, 'hello', 0])

'Some are true.'

>>> checkItems([0, (), ''])

'All are false.'

# Write a function that checks if some or all of the items are true or false.

def checkItems(items):

Topic 14: Question 20

**sorted(***iterable[, cmp[, key[, reverse]]]***)** Return a new sorted list from the elements in *iterable*.  
The *key* is a function that specifies the item in an element to be used for comparision.  
*reverse* sorts the elements in the opposite order.

**Examples**

>>> sorted('python')

['h', 'n', 'o', 'p', 't', 'y']

>>> sorted('python', reverse=True) *# sort in descending order*

['y', 't', 'p', 'o', 'n', 'h']

>>> sorted([('x', 4), ('z', 2), ('y', 1)])

[('x', 4), ('y', 1), ('z', 2)]

>>> sorted([('x', 4), ('z', 2), ('y', 1)], key=**lambda** x:x[1]) *# use of 'key' function to sort based on 2nd item in tuple.*

[('y', 1), ('z', 2), ('x', 4)]

>>> sorted(['1', '3', '20', '2']) *# '20' comes before '3'*

['1', '2', '20', '3']

>>> sorted(['1', '3', '20', '2'], key=int) *# use of 'key' function*

['1', '2', '3', '20']

>>> **def** cmpint(a, b):

... **if** int(a)>int(b):

... **return** 1

... **elif** int(a)<int(b):

... **return** -1

... **else** int(a)==int(b):

... **return** 0

...

>>> sorted(['1', '3', '20', '2'], cmp=cmpint) *# use of 'cmp' function*

['1', '2', '3', '20']

>>> sorted(['s1q1', 's10q1', 's1q2', 's10q10', 's10q2']) *# sort lexicographically*

['s10q1', 's10q10', 's10q2', 's1q1', 's1q2']

>>> sortqns(['s1q1', 's10q1', 's1q2', 's10q10', 's10q2']) *# sort by stage, followed by question.*

['s1q1', 's1q2', 's10q1', 's10q2', 's10q10']

# Write a function that sorts a list of questions that are organized by 's'tage and

# 'q'uestion numbers.

def cmpqn(a, b):

def sortqns(qnlist):

return sorted(qnlist, cmp=cmpqn)