

**An**

**Assignment on**

**Data Programming**

**(Problem Set 2)**

**Submitted by**

**Rohan Hasnani (200533434)**

**To**

**Prof. Les King**

1. Consider the following Python module:

a = 0

def b():

global a

a = c(a)

def c(a):

return a + 2

After importing the module into the interpreter, you execute:

>>> b()

>>> b()

>>> b()

>>> a

What value is displayed when the last expression (a) is evaluated? Explain your answer by indicating what happens in every executed statement.

Text

Description automatically generated with medium confidence

The value displayed is 6 after executing the last expression.

In the following code we have declared a as global which will allow us to modify a outside the function.

Therefore, in function b() we assign the returned value from function c(a)

In function c(a), we are incrementing the value of a by two and returning it.

Finally when we execute b() three times, every time it increments a by 2 and assign that value to a therefore we got the final output as 6.

1. Function fileLength(), given to you, takes the name of a file as input and returns the length of the file:

>>> fileLength('midterm.py')

284

>>> fileLength('idterm.py')

Traceback (most recent call last):

File "<pyshell#34>", line 1, in <module>

fileLength('idterm.py')

File "/Users/me/midterm.py", line 3, in fileLength

infile = open(filename)

FileNotFoundError: [Errno 2] No such file or directory: 'idterm.py'

As shown above, if the file cannot be found by the interpreter or if it cannot be read as a text file, an exception will be raised. Modify function fileLength() so that a friendly message is printed instead:

>>> fileLength('midterm.py')

358

>>> fileLength('idterm.py')

File idterm.py not found.

Text, letter

Description automatically generated

Using try and catch clause we can print a user-friendly message in case of an error occurred.

1. Write a class named Marsupial that can be used as shown below:

>>> m = Marsupial()

>>> m.put\_in\_pouch('doll')

>>> m.put\_in\_pouch('firetruck')

>>> m.put\_in\_pouch('kitten')

>>> m.pouch\_contents()

['doll', 'firetruck', 'kitten']

Now write a class named Kangaroo as a subclass of Marsupial that inherits all the attributes of Marsupial and also:

a. extends the Marsupial \_\_init\_\_ constructor to take, as input, the coordinates x and y of the Kangaroo object,

b. supports method jump that takes number values dx and dy as input and moves the kangaroo by dx units along the x-axis and by dy units along the yaxis, and

c. overloads the \_\_str\_\_ operator so it behaves as shown below.

>>> k = Kangaroo(0,0)

>>> print(k)

I am a Kangaroo located at coordinates (0,0)

>>> k.put\_in\_pouch('doll')

>>> k.put\_in\_pouch('firetruck')

>>> k.put\_in\_pouch('kitten')

>>> k.pouch\_contents()

['doll', 'firetruck', 'kitten']

>>> k.jump(1,0)

>>> k.jump(1,0)

>>> k.jump(1,0)

>>> print(k)

I am a Kangaroo located at coordinates (3,0)

Text

Description automatically generated

Text

Description automatically generated

1. Write function collatz() that takes a positive integer x as input and prints the Collatz sequence starting at x. A Collatz sequence is obtained by repeatedly applying this rule to the previous number x in the sequence:

X = x/2, if x is even

X = 3x + 1, if x is odd

Your function should stop when the sequence gets to number 1. Your implementation must be recursive, without any loops.

>>> collatz(1)

1

>>> collatz(10)

10

5

16

8

4

2

1

Graphical user interface, application

Description automatically generated

1. Write a recursive method binary() that takes a non-negative integer n and prints the binary representation of integer n.

>>> binary(0)

0

>>> binary(1)

1

>>> binary(1)

11

>>> binary(9)

1001

Text

Description automatically generated

1. Implement a class named HeadingParser that can be used to parse an HTML document, and retrieve and print all the headings in the document. You should implement your class as a subclass of HTMLParser, defined in Standard Library module html.parser. When fed a string containing HTML code, your class should print the headings, one per line and in the order in which they appear in the document. Each heading should be indented as follows: an h1 heading should have indentation 0, and h2 heading should have indentation 1, etc. Test your implementation using w3c.html.

Text

Description automatically generated

Text

Description automatically generated

1. Implement recursive function webdir() that takes as input: a URL (as a string) and non-negative integers depth and indent. Your function should visit every web page reachable from the starting URL web page in depth clicks or less, and print each web page's URL. As shown below, indentation, specified by indent, should be used to indicate the depth of a URL.

>>> webdir('http://reed.cs.depaul.edu/lperkovic/csc242/test1.html' , 2, 0) http://reed.cs.depaul.edu/lperkovic/csc242/test1.html http://reed.cs.depaul.edu/lperkovic/csc242/test2.html http://reed.cs.depaul.edu/lperkovic/csc242/test4.html http://reed.cs.depaul.edu/lperkovic/csc242/test3.html <http://reed.cs.depaul.edu/lperkovic/csc242/test4.html>

1. Write SQL queries on the below database table that return:
2. All the temperature data.

SELECT temperature FROM table\_name;

1. All the cities, but without repetition.

SELECT DISTINCT cities FROM table\_name;

1. All the records for India.

SELECT \* FROM table\_name WHERE country = ‘India’.

1. All the Fall records.

SELECT \* FROM table\_name WHERE season = ‘Fall’;

1. The city, country, and season for which the average rainfall is between 200 and 400 millimeters.

SELECT city, country, season FROM table\_name WHERE rainfall >= 200 AND rainfall <= 400;

1. The city and country for which the average Fall temperature is above 20 degrees, in increasing temperature order.

SELECT city, countru FROM table\_name WHERE temperature > 20 ORDER BY temperature ASC;

1. The total annual rainfall for Cairo.

SELECT COUNT(rainfall) FROM table\_name WHERE city = ‘cairo’;

1. The total rainfall for each season.

SELECT season, count(rainfall) FROM table\_name GROUP BY season;

1. Suppose list words is defined as follows:

>>> words = ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']

Write list comprehension expressions that use list words and generate the following lists:

1. ['THE', 'QUICK', 'BROWN', 'FOX', 'JUMPS', 'OVER', 'THE', 'LAZY', 'DOG']

A picture containing text

Description automatically generated

1. ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']

Graphical user interface, application

Description automatically generated

1. [3, 5, 5, 3, 5, 4, 3, 4, 3] (the list of lengths of words in list words).

Graphical user interface, application

Description automatically generated

1. [['THE', 'the', 3], ['QUICK', 'quick', 5], ['BROWN', 'brown', 5], ['FOX', 'fox', 3], ['JUMPS', 'jumps', 5], ['OVER', 'over', 4], ['THE', 'the', 3], ['LAZY', 'lazy', 4], ['DOG', 'dog', 3]] (the list containing a list for every word of list words, where each list contains the word in uppercase and lowercase and the length of the word.)

Text

Description automatically generated with low confidence

1. ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog'] (the list of words in list words containing 4 or more characters.)

Text

Description automatically generated with medium confidence