BDAT1004PS2

November 2, 2022

BDAT 1004 - 02 PROBLEM SET 2

NAME: GOODNEWS AGBADU

STUDENT NUMBER: 200532679

QUESTION 1

```
[35]: #Python Module

      a=0

      def b():
          global a # all references to a in c() are to the global a
          a = c(a) # global a is changed

      def c(a):
          return a + 2
```

```
[36]: b()
      b()
      b()
      a
```

```
[36]: 6
```

Explanation:

In every executed statement since b is defined as the function of the global variable

Function c takes a as a parameter and returns each time with increments 2

b() returns 2

b() returns 4

b() returns 6

So that a = 6

QUESTION 2

```
[37]: def filelength(file):
          try:
              file=open(file)
              contents=file.read()
              file.close()
              print(len(contents))
          except:
              FileNotFoundError()
              print(f'{file} not found.')

      filelength('FileLength.txt')
      filelength('goodnewsLength.txt')
```

```
123
goodnewsLength.txt not found.
```

QUESTION 3

```
[39]: class Marsupial(): #a class named Marsupial
          def __init__(self):
              self.contents = [] #creating an ampty list to store contents

          def put_in_pouch(self,thing):
              self.contents.append(thing) #appending to contents

          def pouch_contents(self):
              return self.contents # this will return the contents of the list

      m=Marsupial()
      m.put_in_pouch('doll')
      m.put_in_pouch('firetruck')
      m.put_in_pouch('kitten')
      m.pouch_contents()
```

```
[39]: ['doll', 'firetruck', 'kitten']
```

```
[42]: class kangaroo(Marsupial):#subclass of Marsupial that inherits all the Marsupial
          def __init__(self,x,y): #the coordinates x and y of the Kangaroo object,
              Marsupial.__init__(self)
              self.y=y
              self.x=x
          def jump(self,dx,dy):
              self.x = self.x+dx
              self.y = self.y+dy
          def __str__(self):
              return 'I am a kangaroo located at coordinates({},{})'.format(self.
       ↪x,self.y)
```

```
k=kangaroo(0,0)
print(k)
k.put_in_pouch('doll')
k.put_in_pouch('firetruck')
k.put_in_pouch('kitten')
k.pouch_contents()
k.jump(1,0)
k.jump(1,0)
k.jump(1,0)
print(k)
```

```
I am a kangaroo located at coordinates(0,0)
I am a kangaroo located at coordinates(3,0)
```

QUESTION 4

[43]:
```
def collatz(x):
    if x==1:
        print(x)
        return
    elif x%2==0:
        print(x)
        collatz(x//2)#recursion
    else:
        print(x)
        collatz(3*x+1) #recursion
collatz(1)
collatz(10)
```

```
1
10
5
16
8
4
2
1
```

QUESTION 5

[44]:
```
def binary(n):
    x = str(n % 2)

    if n//2**1 > 0:
        x = x+binary(n//2**1)#Returns n with the bits shifted to the right same␣
 ↪as n>>1
    return x
```

```
print(binary(0))
print(binary(1))
print(binary(3))
print(binary(9))
```

```
0
1
11
1001
```

QUESTION 6

[45]:
```
from html.parser import HTMLParser

class HeadingParser(HTMLParser):#We create a subclass of HTMLParser
    indent = 0
    headerFlag = False


    def handle_starttag(self, tag, attrs):

            #Each heading should be indented as follows:
            #an h1 heading should have ndentation 0,
            #and h2 heading should have indentation 1, etc.
            #Test your implementation using w3c.html.
        if tag == "h1":
            self.headerFlag = True
            self.indent= 0
        elif tag == "h2":
            self.headerFlag = True
            self.indent= 1
        elif tag == "h3":
            self.headerFlag = True
            self.indent= 2
        elif tag == "h4":
            self.headerFlag = True
            self.indent= 3
        elif tag == "h5":
            self.headerFlag = True
            self.indent= 4
        elif tag == "h6":
            self.headerFlag = True
            self.indent= 5
        else:
            self.headerFlag = False
            self.indent= 0
    def handle_data(self, w3c_data):
        if self.headerFlag:
            to_indent = " "*self.indent
            print('{} {}'.format(to_indent,w3c_data))
    def handle_endtag(self, tag):
        if tag in ["h1","h2","h3","h4","h5","h6"]:
            self.headerFlag = False

infile = open("w3c.txt")
content = infile.read()
infile.close()
hp = HeadingParser()
hp.feed(content)
```

```
W3C Mission
 Principles
```

QUESTION 7

[46]:
```
#import HTMLParser, urlopen, urljoin
#coding idea from stackoverflow

from html.parser import HTMLParser
from urllib.request import urlopen
from urllib.parse import urljoin

class Parser(HTMLParser):

    def __init__(self, url):
        HTMLParser.__init__(self)
        self.url = url
        self.url_list = []

    def append_list(self):
        return self.url_list

    def handle_starttag(self, tags, attri):
        for head in attri:
            contents = urljoin(self.url, head[1])
            if contents[:4] == 'http':
                self.url_list.append(contents)
indentation = 0
def webdir(url, distance, indentation):


    distance -= 1
    print(indentation*'  ', url)

    object = urlopen(url).read().decode()
    column = Parser(url)
    column.feed(object)
    url_link = column.append_list()

    url_list = url_link
    indentation += 4

    for links in url_list:
        if distance < 0 or indentation < 0:
            return 1
        else:
            webdir(links, distance, indentation)

webdir('http://reed.cs.depaul.edu/lperkovic/test1.html', 2, 0) #recursive␣
 ↪function
```

```
http://reed.cs.depaul.edu/lperkovic/test1.html
        http://reed.cs.depaul.edu/lperkovic/test2.html
                http://reed.cs.depaul.edu/lperkovic/test4.html
        http://reed.cs.depaul.edu/lperkovic/test3.html
                http://reed.cs.depaul.edu/lperkovic/test4.html
```

QUESTION 8

[2]:
```
!pip install ipython-sql
import sqlite3
con = sqlite3.connect('goodnewsdatabase.db')
cur = con.cursor()
```

```
Requirement already satisfied: ipython-sql in /opt/anaconda3/lib/python3.9/site-
packages (0.4.1)
Requirement already satisfied: ipython>=1.0 in
/opt/anaconda3/lib/python3.9/site-packages (from ipython-sql) (7.29.0)
Requirement already satisfied: prettytable<1 in
/opt/anaconda3/lib/python3.9/site-packages (from ipython-sql) (0.7.2)
Requirement already satisfied: sqlparse in /opt/anaconda3/lib/python3.9/site-
packages (from ipython-sql) (0.4.3)
Requirement already satisfied: sqlalchemy>=0.6.7 in
/opt/anaconda3/lib/python3.9/site-packages (from ipython-sql) (1.4.36)
Requirement already satisfied: six in /opt/anaconda3/lib/python3.9/site-packages
(from ipython-sql) (1.16.0)
Requirement already satisfied: ipython-genutils>=0.1.0 in
/opt/anaconda3/lib/python3.9/site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: jedi>=0.16 in /opt/anaconda3/lib/python3.9/site-
packages (from ipython>=1.0->ipython-sql) (0.18.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/opt/anaconda3/lib/python3.9/site-packages (from ipython>=1.0->ipython-sql)
(3.0.20)
Requirement already satisfied: pexpect>4.3 in /opt/anaconda3/lib/python3.9/site-
```

```
packages (from ipython>=1.0->ipython-sql) (4.8.0)
Requirement already satisfied: backcall in /opt/anaconda3/lib/python3.9/site-
packages (from ipython>=1.0->ipython-sql) (0.2.0)
Requirement already satisfied: setuptools>=18.5 in
/opt/anaconda3/lib/python3.9/site-packages (from ipython>=1.0->ipython-sql)
(58.0.4)
Requirement already satisfied: appnope in /opt/anaconda3/lib/python3.9/site-
packages (from ipython>=1.0->ipython-sql) (0.1.2)
Requirement already satisfied: decorator in /opt/anaconda3/lib/python3.9/site-
packages (from ipython>=1.0->ipython-sql) (5.1.0)
Requirement already satisfied: traitlets>=4.2 in
/opt/anaconda3/lib/python3.9/site-packages (from ipython>=1.0->ipython-sql)
(5.1.0)
Requirement already satisfied: pickleshare in /opt/anaconda3/lib/python3.9/site-
packages (from ipython>=1.0->ipython-sql) (0.7.5)
Requirement already satisfied: pygments in /opt/anaconda3/lib/python3.9/site-
packages (from ipython>=1.0->ipython-sql) (2.10.0)
Requirement already satisfied: matplotlib-inline in
/opt/anaconda3/lib/python3.9/site-packages (from ipython>=1.0->ipython-sql)
(0.1.2)
Requirement already satisfied: greenlet!=0.4.17 in
/opt/anaconda3/lib/python3.9/site-packages (from sqlalchemy>=0.6.7->ipython-sql)
(1.1.2)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in
/opt/anaconda3/lib/python3.9/site-packages (from
jedi>=0.16->ipython>=1.0->ipython-sql) (0.8.2)
Requirement already satisfied: ptyprocess>=0.5 in
/opt/anaconda3/lib/python3.9/site-packages (from
pexpect>4.3->ipython>=1.0->ipython-sql) (0.7.0)
Requirement already satisfied: wcwidth in /opt/anaconda3/lib/python3.9/site-
packages (from prompt-
toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=1.0->ipython-sql) (0.2.5)
WARNING: You are using pip version 22.0.4; however, version 22.3 is

available.

You should consider upgrading via the '/opt/anaconda3/bin/python -m pip install

--upgrade pip' command.
```

```
[ ]: cur.execute("CREATE TABLE CITY_DATA (City text, Country text, Season text,␣
      ↪Temperature float, Rainfall float)")
     data=[('Mumbai', 'India','Winter',24.8, 5.9),('Mumbai','India','Spring',28.4,16.
      ↪2),('Mumbai','India','Summer',27.9,1549.4),
           ('Mumbai','India','Fall',27.6,346.0),('London','United Kingdom','Winter',4.
      ↪2,207.7),
    ('London','United Kingdom','Spring',8.3,169.6),('London','United␣
↪Kingdom','Summer',15.7,157.0),
    ('London','United Kingdom','Fall',10.4,218.5),('Cairo','Egypt','Winter',13.
↪6,16.5),('Cairo','Egypt','Spring',20.7,6.5),
    ('Cairo','Egypt','Summer',27.7,0.1),('Cairo','Egypt','Fall',22.2,4.5)]
for cities in data:
    cur.execute("INSERT INTO CITY_DATA VALUES (?,?,?,?,?)",cities)
    con.commit()
```

[47]:
```
#a) All the temperature data.
cur.execute('SELECT Temperature FROM CITY_DATA')
cur.fetchall()
```

[47]:
```
[(24.8,),
 (28.4,),
 (27.9,),
 (27.6,),
 (4.2,),
 (8.3,),
 (15.7,),
 (10.4,),
 (13.6,),
 (20.7,),
 (27.7,),
 (22.2,)]
```

[48]:
```
#b) All the cities, but without repetition.
cur.execute('SELECT DISTINCT City FROM CITY_DATA')
cur.fetchall()
```

[48]:
```
[('Mumbai',), ('London',), ('Cairo',)]
```

[49]:
```
#c) All the records for India.
cur.execute('SELECT * FROM CITY_DATA WHERE Country LIKE "India%"')
cur.fetchall()
```

[49]:
```
[('Mumbai', 'India', 'Winter', 24.8, 5.9),
 ('Mumbai', 'India', 'Spring', 28.4, 16.2),
 ('Mumbai', 'India', 'Summer', 27.9, 1549.4),
 ('Mumbai', 'India', 'Fall', 27.6, 346.0)]
```

[50]:
```
#d) All the Fall records.
cur.execute('SELECT * FROM CITY_DATA WHERE Season LIKE "Fall%"')
cur.fetchall()
```

[50]:
```
[('Mumbai', 'India', 'Fall', 27.6, 346.0),
 ('London', 'United Kingdom', 'Fall', 10.4, 218.5),
 ('Cairo', 'Egypt', 'Fall', 22.2, 4.5)]
```

[51]:
```
 #e) The city, country, and season for which the average rainfall
#is between 200 and 400 millimeters.

cur.execute('SELECT City, Country, Season FROM CITY_DATA WHERE Rainfall BETWEEN␣
 ↪200 AND 400')
cur.fetchall()
```

[51]:
```
[('Mumbai', 'India', 'Fall'),
 ('London', 'United Kingdom', 'Winter'),
 ('London', 'United Kingdom', 'Fall')]
```

[52]:
```
 #f) The city and country for which the average Fall temperature is above 20␣
 ↪degrees,
#in increasing temperature order.

cur.execute('SELECT City, Country,Temperature FROM CITY_DATA WHERE Temperature␣
 ↪> 20 ORDER BY Temperature DESC')
cur.fetchall()
```

[52]:
```
[('Mumbai', 'India', 28.4),
 ('Mumbai', 'India', 27.9),
 ('Cairo', 'Egypt', 27.7),
 ('Mumbai', 'India', 27.6),
 ('Mumbai', 'India', 24.8),
 ('Cairo', 'Egypt', 22.2),
 ('Cairo', 'Egypt', 20.7)]
```

[53]:
```
#g) The total annual rainfall for Cairo.
cur.execute('SELECT SUM(Rainfall) AS RF, City FROM CITY_DATA WHERE City LIKE␣
 ↪"Cairo" GROUP BY City')
cur.fetchall()
```

[53]:
```
[(27.6, 'Cairo')]
```

[54]:
```
#h) The total rainfall for each season.

cur.execute('SELECT SUM(Rainfall),Season FROM CITY_DATA GROUP BY Season')
cur.fetchall()
```

[54]:
```
[(569.0, 'Fall'),
 (192.29999999999998, 'Spring'),
 (1706.5, 'Summer'),
 (230.1, 'Winter')]
```

QUESTION 9

```
[26]: words = ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
```

```
[55]: lista = [a.upper() for a in words]
      lista
```

```
[55]: ['THE', 'QUICK', 'BROWN', 'FOX', 'JUMPS', 'OVER', 'THE', 'LAZY', 'DOG']
```

```
[56]: listb = [b.lower() for b in words]
      listb
```

```
[56]: ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
```

```
[57]: listc = [len(c) for c in words] #the list of lengths of words in list words).
      listc
```

```
[57]: [3, 5, 5, 3, 5, 4, 3, 4, 3]
```

```
[58]: #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.
      listd = [(d.upper(),d.lower(),len(d)) for d in words]
      listd
```

```
[58]: [('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)]
```

```
[59]: #the list of words in list words containing 4 or more characters.

      liste = [e for e in words if len(e)>=4]
      liste
```

```
[59]: ['quick', 'brown', 'jumps', 'over', 'lazy']
```