Problem Set 2

Rishabh Patadia - 200496110, Section 3

Question 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

In [1]:

```
# Code given in the question

# this statement declares and initialises the variable "a" to value of 0
a = 0

# here the function "b" is declared, since it is not called, its value is not read
# that is why even of the funciton "c" is not defined, it does not raise any error
# function declaration
def b():
    # here instead of initialising a local variable "a"
    # global keyword is used to refer the global variable a from previous statement
    global a
    # in this statement the global value is changed to the value returned by "c" functi
    a = c(a)

# function declaration, with accepting 1 argument
def c(a):
    # returning the input value in the function after incrementing it by 2
    return a + 2


# Code to be executed, Given in the question
```

```
# a = 0
b() # b calls c and passing value of a, c increment a by 2,
# a = a + 2, a = 0 + 2,THEREFORE a = 2
b() # b calls c and passing value of a, c increment a by 2,
# a = a + 2, a = 2 + 2, THEREFORE a = 4
b() # b calls c and passing value of a, c increment a by 2,
# a = a + 2, a = 4 + 2, THEREFORE a = 6
a # prints the current value of a = 6
```

Out[1]:
```
6
```

Question 2

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.
```

In [2]:
```
# Code given in the question

# function declaration and accepting 1 argument
def file_length(file_name):
    # inserting a try-except block to handle the errors raised
    # in the try block the code that may raise the error is written
    try:
```

```
        # this statement opens a file name that is passed in the function
        # if the file is not found it will raise "FileNotFoundError" error
        # and move on to the "except" block
        # if the file is found it is saved in the "file" variable
        file = open(file_name)
        # the contents of the "file" is read using "read" method and saved in the "cont
        contents = file.read()
        # "close" method is used to free up the memory and give up the access of file f
        file.close()
        # "len" function is used to calculate the length of the contents varible,
        #in this case it returns the charater count
        print(len(contents))
    # in the except block, code is written to handle the error
    # if the error is mentioned, that specific error is handled,
    #else all the error raised are handled
    # here we are only handling the "FileNotFoundError" error
    except FileNotFoundError:
        # the "format" method is used to insert the value of the variable in the string
        # and the position of the varibale is decided by the "{}" in the string
        print("File '{}' not found.".format(file_name))
```

In [3]:
```
file_length('w3c.html') # the folder that contains this file also contains the "w3c.htm
```

```
396
```

In [4]:
```
file_length('3c.html')
```

```
File '3c.html' not found.
```

Question 3

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 y axis, 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)
```

In [5]:

```
# class definintion
class Marsupial:
    # using dunder methods or magic method to override the inbuilt
    #method while instantiating the class
    def __init__(self):
        # intialising a blank array in "contents" variable
        #when a new variable is instantiated
        self.contents = []
    # defining user defined method that adds the the passed
    #argument to the contents array
    def put_in_pouch(self, a):
        # append is used to add an element in an array
        self.contents.append(a)
    # defining a user defined method to print the "contents" variable
    def pouch_contents(self):
        print(self.contents)

# class definintion with inheritance
class Kangaroo(Marsupial):
    # using dunder methods or magic method to override the
    #inbuilt method while instantiating the class
    def __init__(self, x=0, y=0):
        # using the inherited class __init__ method in the current class __init__ metho
        Marsupial.__init__(self)
        # creating additional variables "x" and "y"
        self.x = x
        self.y = y
    # overring the inbuilt string method to display a custom string
    def __str__(self):
        # the string method always needs to return a string
        return "I am a Kangaroo located at coordinates ({},{})".format(self.x, self.y)
    # custom method defined to add the interval passed as argument to the x-y coordinat
    def jump(self, x, y):
        self.x += x
        self.y += y



# Code to be executed, Given in the question
#----------------------------------
m = Marsupial() # creating an instant of Marsupial class and storing in "m"
m.put_in_pouch('doll') # adding the passed argument to the contents variable of "m"
m.put_in_pouch('firetruck') # adding the passed argument to the contents variable of "m
```

```
m.put_in_pouch('kitten') # adding the passed argument to the contents variable of "m"
m.pouch_contents() # displaying the "contents" variable of "m"

#---------------------------------
k = Kangaroo(0,0) # creating an instant of Kangaroo class and storing in "k"
print(k) # printing the "k" to call its __str__ method
k.put_in_pouch('doll') # adding the passed argument to the contents variable of "k"
k.put_in_pouch('firetruck') # adding the passed argument to the contents variable of "k
k.put_in_pouch('kitten') # adding the passed argument to the contents variable of "k"
k.pouch_contents() # displaying the "contents" variable of "k"
k.jump(1,0) # passing the change in values of x-y to the k
k.jump(1,0) # passing the change in values of x-y to the k
k.jump(1,0) # passing the change in values of x-y to the k
print(k) # printing the "k" to call its __str__ method
```

```
['doll', 'firetruck', 'kitten']
I am a Kangaroo located at coordinates (0,0)
['doll', 'firetruck', 'kitten']
I am a Kangaroo located at coordinates (3,0)
```

Question 4

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
```

In [6]:

```
# defining a function which accepts 1 argument
def collatz(x):
    # printing the passed argument
    print(x)
    # checking whether the passed argument is 1
```

```
    if (x == 1):
        # if passed argument is 1 then terminating the function with null return
        return
    # if the passed argument is not 1, then checking whether the argument is a multiple
    elif (x%2 == 0):
        # if the passed argument is a multiple of 2, then return is half,
        # since division is a float opertration in python, conversion to
        #int is necesary to display required output
        return collatz(int(x/2))
    # of the argument is not 1, not divisible by 2, then it must be an odd number other
    else:
        # returning the 1 plus three times the argument
        return collatz(3*x + 1)
```

In [7]:
```
collatz(1)
```

```
1
```

In [8]:
```
collatz(10)
```

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

Question 5

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(3)
11
>>> binary(9)
1001
```

In [9]:
```
# defining UDF with one argument
def binary(x):
    # checks if arg equals to 1 or 0
    if (x == 0 or x == 1):
        # if so prints it
        print(x, end="")
    else:
        # else calls itself and passes the whole number of arg divided by 2
        binary(x//2)
```

```
        # then prints the remainder after diving by 2
        print(x%2, end="")
```

In [10]:
```
binary(0)
```

```
0
```

In [11]:
```
binary(1)
```

```
1
```

In [12]:
```
binary(3)
```

```
11
```

In [13]:
```
binary(9)
```

```
1001
```

In [14]:
```
binary(10)
```

```
1010
```

Question 6

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.

```
>>> infile = open('w3c.html')
>>> content = infile.read()
>>> infile.close()
>>> hp = HeadingParser()
>>> hp.feed(content)
W3C Mission
    Principles
```

In [15]:
```
from html.parser import HTMLParser # to import the inbuilt class mentioned in the quest
```

```
# Given in the question
infile = open('w3c.html')
content = infile.read()
infile.close()

# implementing HeadingParser class as mentioned in the question
class HeadingParser(HTMLParser):
    rec = 0 # record of indent variable
    look = {'h'+str(_+1):_ for _ in range(6)} # lookup table to identify the type of he
    prnt = False # flag variable to decide when to print

    # overriding the handle_starttag method
    def handle_starttag(self, tag, attrs):
        # checking if the tag is any heading tag
        if(tag in self.look.keys()):
            #  change print flag to true
            self.prnt = True
            # changing the record of indent variable to appropriate indent
            self.rec = self.look[tag]
        else:
            #  change print flag to true
            self.prnt = False
            # changing the record of indent variable to 0
            self.rec = 0

    # overriding the handle_data method
    def handle_data(self, data):
        # checking if the print flag is true and the data is non empty string or string
        if (self.prnt and data.strip()!=''):
            print("\t"*self.rec, data)

#instantiating the HeadingParser class
parser = HeadingParser()
# feeding the content extracted from "w3c.html"
parser.feed(content)
```

```
W3C Mission
        Principles
```

Question 7

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
```

In [16]:

```
import requests
from bs4 import BeautifulSoup
def webdir(url, depth, indent):
    print("\t"*indent + url)
    if (depth==0):
        return
    listOfUrl = []
    grab = requests.get(url)
    print(grab.text)
    soup = BeautifulSoup(grab.text, 'html.parser')

    for link in soup.find_all("a"):
        data = link.get('href')
        listOfUrl.append(url[:-11:1]+"/"+data)
    print(listOfUrl)
    for _ in listOfUrl:
        if (_[-4:0].lower() == "html"):
            webdir(_, depth-1, indent+1)

webdir('http://reed.cs.depaul.edu/lperkovic/test1.html', 2, 0)
```

```
http://reed.cs.depaul.edu/lperkovic/test1.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Test page</title>
</head>
<body>
<h2> Hello World </h2>
<a href="http://reed.cs.depaul.edu/lperkovic/test2.html"> Link
to test2.html </a>
<p> <a href="test3.html"> Link to test3.html</a><br>
</p>
<p>Paris Paris Paris<br>
Tokyo<br>
</p>
</body>
</html>

['http://reed.cs.depaul.edu/lperkovic/http://reed.cs.depaul.edu/lperkovic/test2.html',
'http://reed.cs.depaul.edu/lperkovic/test3.html']
```

Question 8

Write SQL queries on the below database table that return:
a) All the temperature data.
b) All the cities, but without repetition.
c) All the records for India.

d) All the Fall records.
e) The city, country, and season for which the average rainfall is between 200
and 400 millimeters.
f) The city and country for which the average Fall temperature is above 20
degrees, in increasing temperature order.
g) The total annual rainfall for Cairo.
h) The total rainfall for each season.

City	Country	Season	Temperature (C)	Rainfall (mm)
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

Assumptions:

1. Since the column names cannot have space in the SQL

column names are (City, Country, Season, Temperature, Rainfall)

2. Table name is weatherData

a

```
SELECT Temperature FROM weatherData;
```

b

```
SELECT DISTINCT City FROM weatherData;
```

c

```
SELECT * FROM weatherData WHERE Country = 'India';
```

d

```
SELECT * FROM weatherData WHERE Season = 'Fall';
```

e

```
SELECT City, Country, Season FROM weatherData WHERE Rainfall>=200 AND Rainfall<=400;
```

f

```
SELECT City, Country FROM weatherData WHERE Season = 'Fall' AND Temperature>20 ORDER BY Temperature ASC;
```

g

```
SELECT SUM(Rainfall) FROM weatherData WHERE City = 'Cairo';
```

h

```
SELECT SUM(Rainfall) FROM weatherData WHERE Season = 'Winter';
SELECT SUM(Rainfall) FROM weatherData WHERE Season = 'Spring';
SELECT SUM(Rainfall) FROM weatherData WHERE Season = 'Summer';
SELECT SUM(Rainfall) FROM weatherData WHERE Season = 'Fall';
```

Question 9

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:
a) ['THE', 'QUICK', 'BROWN', 'FOX', 'JUMPS', 'OVER', 'THE', 'LAZY', 'DOG']
b) ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
c) [3, 5, 5, 3, 5, 4, 3, 4, 3] (the list of lengths of words in list words).
d) [['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.)
e) ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog'] (the list of words in list words containing 4 or more characters.)

In [18]:

```
# Given in the question
words = ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']

# This is the general template of list comprehension
# newList = [ expression(element) for element in oldList if condition ]    => from GFG
# This expression calculates the list of new elements in newList which contains express
# applied on every element in oldList if the condition is true

# Answer A
# Here the expression is element.upper()
AnswerA = [_.upper() for _ in words]
print("Ans A =", AnswerA)

# Answer B
# Here the expression is element.lower()
AnswerB = [_.lower() for _ in words]
print("Ans B =", AnswerB)

# Answer C
# Here the expression is len(element)
AnswerC = [ len(_) for _ in words]
print("Ans C =", AnswerC)

# Answer D
```

```python
# Here the expression is [element.upper(), element.lower(), len(element)]
AnswerD = [[_.upper(), _.lower(), len(_)] for _ in words]
print("Ans D =", AnswerD)

# Answer E
# Here the expression is element with the condition as len(element>4)
AnswerE = [_ for _ in words if len(_)>=4]
print("Ans E =", AnswerE)
```

```
Ans A = ['THE', 'QUICK', 'BROWN', 'FOX', 'JUMPS', 'OVER', 'THE', 'LAZY', 'DOG']
Ans B = ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
Ans C = [3, 5, 5, 3, 5, 4, 3, 4, 3]
Ans D = [['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]]
Ans E = ['quick', 'brown', 'jumps', 'over', 'lazy']
```