CPSC 1301, Computer Science I Lab Assignment

Lab 03b

1 Can a function call another function?

Rule: Function calls may be nested to one or more levels.

1.1 func A calls Func B

```
1  # lab03b_01.py
2
3  def funcA():
4      funcB()
5
6  def funcB():
7      print("This is function B")
8
9  funcA()
```

OUTPUT

```
This is function B
```

1.2 func A calls Func B calls Func C

```
1  # lab03b_02.py
2
3  def funcA():
4      print("calling funcB()")
5      funcB()
6
7  def funcB():
8      print("calling funcC()")
9      funcC()
10
11 def funcC():
12     print("This is function C")
13
14 funcA()
```

OUTPUT

```
calling funcB()
calling funcC()
This is function C
```

2 Can a function call itself?

Rule: Function calls may be recursively nested, that is, the function may repeatedly call itself as many times as desired.

2.1 func A calls itself counting to <argument>

```
1  # lab03b_03.py
2
3  import sys
4
5  if len(sys.argv) != 2:
6      print("INCORRECT. Call with one integer parameter.")
7      sys.exit()
8
9  top = int(sys.argv[1])
10 print("I'm counting to", top)
11
12 def myFun(n):
13     if n > top:
14         print("Done!")
15     else:
16         print(n)
17         myFun(n + 1)
18
19 myFun(1)
```

OUTPUT

```
I'm counting to 10
1
2
3
4
5
6
7
8
9
10
Done!
```

2.2 func A calls itself counting from <argument>

```
1  # lab03b_05.py
2
3  import sys
4
5  if len(sys.argv) != 2:
6      print("Call this method with one integer parameter")
7      sys.exit()
8
9  numBeers = int(sys.argv[1])
10
11 def drink(nb):
12     if nb == 0:
13         print("All gone, burp ...")
14     else:
15         print(nb, "beers left")
16         drink(nb - 1)
17
18 drink(numBeers)
```

OUTPUT

```
6 beers left
5 beers left
```

```
4 beers left
3 beers left
2 beers left
1 beers left
All gone, burp ...
```

3 How do you keep track of the result?

Rule: You add a parameter to the function that updates itself with each call and returns (or prints) the result for the base case.

3.1 func A calls itself adding to <argument>

```
1  % lab03b_04.py
2
3  import sys
4
5  if len(sys.argv) != 2:
6      print("INCORRECT. Call with one integer parameter.")
7      sys.exit()
8
9  top = int(sys.argv[1])
10 print("I'm adding to", top)
11
12 def myFun(sum, n):
13     if n > top:
14         print("The sum is", sum)
15     else:
16         myFun(sum + n, n + 1)
17
18 myFun(0, 0)
```

OUTPUT

```
I'm adding to 10
The sum is 55
```

3.2 adding integers <start> to <end> inclusive

```
1  #!python
2
3  # Name lab03b_08.py
4  # Author: Charles Carter
5  # Date: May 17, 2021
6  # Purpose: to add all integers between <start> and <end>
7
8  def addem(total, start, end):
9      if start > end:
10         print("The total is", total)
11     else:
12         addem(total + start, start + 1, end)
13
14 print("Enter the starting integer: ", end = '')
15 start = int(input())
16 print("Enter the ending integer: ", end = '')
17 end = int(input())
18 print("start is", start, " and end is", end)
19
20 addem(0, start, end)
```

OUTPUT

```
Enter the starting integer: 4
Enter the ending integer: 8
start is 4  and end is 8
The total is 30
```

3.3 func A calls itself dividing <dividend> by <divisor>

```
1  # lab03b_06.py
2
3  import sys
4
5  if len(sys.argv) != 3:
6      print("Call this method with two integer parameters: <dividend>, <divisor>")
7      sys.exit()
8
9  dividend = int(sys.argv[1])
10 divisor = int(sys.argv[2])
11 quotient = 0
12
13 print("calling divide( %d, %d, %d)" % (quotient, dividend, divisor))
14 def divide(quotient, dend, dsor):
15     if dend < dsor:
16         print("The quotient is", quotient, "and the remainder is", dend)
17     else:
18         divide(quotient + 1, dend - dsor, dsor)
19
20
21 divide(quotient, dividend, divisor)
```

OUTPUT

```
calling divide( 0, 23, 8)
The quotient is 2 and the remainder is 7
```

3.4 building a string recursively

```
1  # lab03b_10.py
2
3
4  def build(what):
5      print("Please enter a character or digit, <RETURN> to finish: ", end = '')
6      char = input()
7      if char == "":
8          print(what)
9      else:
10         build(what + char)
11
12 build("")
```

OUTPUT

```
    Please enter a character or digit, <RETURN> to finish: P
Please enter a character or digit, <RETURN> to finish: y
Please enter a character or digit, <RETURN> to finish: t
Please enter a character or digit, <RETURN> to finish: h
Please enter a character or digit, <RETURN> to finish: o
Please enter a character or digit, <RETURN> to finish: n
Please enter a character or digit, <RETURN> to finish:
```

Python

4 What about lists or strings?

Rule: A list or string has a *first* element which can be accessed with `list[0]` (i.e., *first* is the zeroth element), and *rest* elements which can be accessed with `list[1:]` (i.e., the second element to the last element).

4.1 manipulating a list

```
1  # lab03b_09.py
2
3  groceries = ["milk","bread","diapers","chips","salsa"]
4
5  def printlist(g):
6      print("Here is my list")
7      print(g)
8      print("----------------------")
9
10 def shop(g):
11     if len(g) == 0:
12         print("headed to checkout ...")
13     else:
14         print(g[0], "in cart")
15         shop(g[1:])
16
17 printlist(groceries)
18 shop(groceries)
19
20     \textbf{OUTPUT}
21
22     \begin{verbatim}
23 Here is my list
24 ['milk', 'bread', 'diapers', 'chips', 'salsa']
25 ----------------------
26 milk in cart
27 bread in cart
28 diapers in cart
29 chips in cart
30 salsa in cart
31 headed to checkout ...
32     \end{verbatim}
```

4.2 doing a string

```
1  #!python
2
3  # Name lab03b_07.py
4  # Author: Charles Carter
5  # Date: May 17, 2021
6  # Purpose: to return a string of characters
7
8  import string
9
10 myString = "hello world"
11
12 def recur_str(s, l):
13     if len(l) == 0:
14         return s
15     else:
16         return recur_str(s + l[0].upper(), l[1:])
17
```

```
18 if __name__ == '__main__':
19     s = recur_str("", myString)
20     print('string is ', s)
```

OUTPUT

```
string is  HELLO WORLD
```