

Python Programming II

Offensive and Defensive Tool Construction

Table of Contents

[Objectives 2](#_Toc476836086)

[Background Reading 2](#_Toc476836087)

[Important Information 2](#_Toc476836088)

[Problem 1 3](#_Toc476836089)

[Problem 2 3](#_Toc476836090)

[Problem 3 4](#_Toc476836091)

[Problem 4 4](#_Toc476836092)

[Problem 5 4](#_Toc476836093)

Offensive and Defensive Tool Construction

Python Programming II

Objectives

This lab focuses on the following objectives:

* Explain conditional statements.
* Understand recursion and how it influences a stack.
* Use strings, lists and tuples.
* Use dictionaries to index data.
* Read and write files.

Background Reading

Read chapters 6–10 in *How to Think Like a Computer Scientist: Learning with Python*, available at [www.greenteapress.com/thinkpython/thinkCSpy.pdf](http://www.greenteapress.com/thinkpython/thinkCSpy.pdf).

# Important Information

* For *every* lab and home assignment, store all your work in your personal repository in a subdirectory named **mXX**, where XX is the module number. Carefully name the program as described in each problem.
* Your programs are extracted from your repository by a Python script. If there are any errors in the program name, then your instructor will never see your program, and you will receive a mark of zero.
* Push your work to the server often, and ensure that you push the final version of a program by the deadline specified, because the script extracting them can be run at any time after the deadline.

# Problem 1

Write a Python program named **m03p01.py** that prints the first 10 rows of Pascal’s triangle in the shape of a triangle. Define the recursive function pascal(row,col)that calculates each element of the triangle. The output should look like this:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

def pascal (row,col):

if col==0:

return 1

elif col==row:

return 1

else:

return pascal(row-1,col)+pascal(row-1, col-1)

rowlist=[]

rowstr=""

for row in range (10):

for col in range (row+1):

num = pascal(row,col)

rowlist.append(str(num)) #add new element to list as a string

rowstr=(' ').join(rowlist) #join the list items to a string for each row

spaces=(26-len(rowstr))/2

print (" "\*spaces), #print spaces in front of row

print (rowstr) #print one row a time

rowstr="" # reset the string and list

rowlist=[]

# Problem 2

Write a Python program named **m03p02.py** that generates a sine wave on the terminal using the ASCII star character.

* Make the amplitude of the sine wave 10 lines, and make one period of the sine wave 60 points (60 characters).
* Draw the zero axis using the dash character.
* Use the dictionary indexed by the amplitude to keep track of which positions on the   
  x-axis should have the star.

The output should look like this:

\*\*\*\*\*\*\*

\*\* \*\*

\* \*

\*\* \*\*

\* \*

\* \*

\* \*

\* \*

\* \*

\* \*

\*-----------------------------\*-----------------------------

\* \*

\* \*

\* \*

\* \*

\* \*

\* \*

\*\* \*\*

\* \*

\*\* \*\*

\*\*\*\*\*\*\*

# import sys

star\_record={0: [12, 13, 14, 15, 16, 17, 18], 1: [10, 11, 19, 20], 2: [9, 21], 3: [7, 8, 22, 23], 4: [6, 24], 5: [5, 25], 6: [4, 26], 7: [3, 27], 8: [2, 28], 9: [1, 29], 10: [0, 30], 11: [31, 59], 12: [32, 58], 13: [33, 57], 14: [34, 56], 15: [35, 55], 16: [36, 54], 17: [37, 38, 52, 53], 18: [39, 51], 19: [40, 41, 49, 50], 20: [42, 43, 44, 45, 46, 47, 48]}

space=' '

star='\*'

dash='-'

for i in range(21):

for j in range (60):

if i ==10:

if j in star\_record[i]:

sys.stdout.write(str(star))

else:

sys.stdout.write(str(dash))

if j==59:

print'\r'

else:

if j in star\_record[i]:

sys.stdout.write(str(star))

else:

sys.stdout.write(str(space))

if j==59:

print'\r'

# Problem 3

Write a Python program named **m04p04.py** that takes a single parameter, a file name, and displays information about the file using the following format:

File Name: some\_file\_name

File Size: 123

Inode : 2498755

Last Mod : Tue Jan 8 07:00:53 2016

import os,sys

import stat

import time

if len (sys.argv)<2:

print "need one file name"

sys.exit(-1)

else:

filename=sys.argv[1]

print """File Name: %s""" % (filename)

print """File Size: %s""" % (os.stat(filename).st\_size)

print """Inode : %s""" % (os.stat(filename).st\_ino)

tm=os.stat(filename).st\_mtime

print """Last Mod : %s""" % (time.ctime(tm))

#stat.ST\_SIZE

#Size in bytes of a plain file; amount of data waiting on some special files.

#stat.ST\_MTIME

#Time of last modification.

#stat.ST\_INO

#Inode number.

#format of time in python

#print utmp.ut\_time (float number for time in seconds)

#print time.ctime(utmp.ut\_time) (convert seconds to time)

#print time.strptime(time.ctime(utmp.ut\_time)) (time in structure)

#print time.strftime('%Y-%m-%d %H:%M:%S',time.strptime(time.ctime(utmp.ut\_time)) (time in format)

# (time.strftime('%Y-%m-%d %H:%M:%S',time.strptime(time.ctime(utmp.ut\_time)))

# Problem 4

Write a Python program named **m03p04.py** that dumps the ELF file header and identifies its parameters (ELF file headers are described at [https://en.wikipedia.org/wiki/  
Executable\_and\_Linkable\_Format](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format)). Print the parameters using the following format:

File : a.out

Magic : 0x464c457f

Format : 64-bit

Endian : big

Machine: x86-64

import sys,os

import struct

if sys.argv==1:

print """need an executable file name"""

sys.exit(-1)

else:

filename=sys.argv[1]

print """File : %s""" %(filename)

fd=open(filename,'rb')

header=fd.read(6)

magic,form,endian=struct.unpack("IBB",header)

print """Magic :""", hex(magic)

if form==1:

print """Format : %s""" %('32 bit')

else:

print """Format : %s""" %('64 bit')

if endian==1:

print """Endian : %s""" %('little')

else:

print """Endian : %s""" %('big')

fd.seek(18,0)

data=fd.read(2)

machine=struct.unpack('h',data)[0]

if machine==0x02:

val='SPARC'

elif machine==0x03:

val='x86'

elif machine==0x08:

val='MIPS'

elif machine==0x14:

val='PowerPC'

elif machine==0x16:

val='S390'

elif machine==0x28:

val='ARM'

elif machine==0x2A:

val='SuperH'

elif machine==0x32:

val='IA-64'

elif machine==0x3E:

val='x86-64'

elif machine==0xB7:

val='AArch64'

elif machine==0xF3:

val='RISC-V'

print """Machine: %s""" %(val)

# Problem 5

Write a Python program named **m03p05.py** that takes a single parameter, a file name, and prints a hexadecimal dump of it using the following format:

[00000000]: 23 69 6e 63 6c 75 64 65 20 3c 73 74 64 69 6f 2e #include <stdio.

[00000010]: 68 3e 0a 0a 76 6f 69 64 0a 6d 61 69 6e 28 29 0a h>..void.main().

[00000020]: 7b 0a 20 20 70 72 69 6e 74 66 20 28 22 48 65 6c {. printf ("Hel

[00000030]: 6c 6f 20 57 6f 72 6c 64 21 5c 6e 22 29 3b 0a 7d lo World!\n");.}

[00000040]: 0a

Total length 65 (41h)

**Note:** The rest of the problems for this module are available in the homework assignment. See your course schedule for details.

#!/usr/bin/python

# takes a single parameter, a file name, and prints a hexadecimal dump

import sys

def check\_file():

# This method ensures a valid file was provided to the invoked script ##

if len(sys.argv) < 2:

print ""

print "Error -need one file name "

sys.exit(0)

def read\_bytes(filename, chunksize=8192):

# This method returns the bytes of a provided file ##

with open(filename, "rb") as f:

while True:

chunk = f.read(chunksize)

if chunk:

for b in chunk:

yield b

else:

break

def validate\_byte\_as\_printable(byte):

## Check if byte is a printable ascii character. If not replace with a '.' character ##

for c in byte:

if(ord(c) < 128) and (ord(c) >= 32):

return byte

else:

return '.'

check\_file()

memory\_address = 0

ascii\_string = ""

## Loop through the given file while printing the address, hex and ascii output ##

count=0

for byte in read\_bytes(sys.argv[1]):

count+=1

ascii\_string = ascii\_string + validate\_byte\_as\_printable(byte)

if memory\_address%16 == 0:

print( '%08X'%memory\_address),

print(byte.encode('hex')),

elif memory\_address%16 == 15:

print(byte.encode('hex')),

print ascii\_string

ascii\_string = ""

else:

print(byte.encode('hex')),

memory\_address = memory\_address + 1

print'\r'

print ('Total length %d (%xh)'%(count,count))

generate dictionary of star positions for a sine curve.

star\_record={}

lines=[]

newlines=[]

lines\_rvs=[]

space=' '

star='\*'

dash='-'

length=30

lines.append((space\*12)+(star\*7))

fsp=10 #number of spaces in front of stars

msp=7 #number of spaces between stars

for i in range (2):

lines.append(space\*fsp+star\*2+space\*msp+star\*2)

fsp-=1 #next line 1 star move forward

msp+=4 #last line 2 star, make 2 space on each side 2\*2

lines.append(space\*fsp+star\*1+space\*msp+star\*1)

fsp-=2 #next line 2 stars move forward

msp+=2 #last line 1 star, make 1 space on each side 1\*2

fsp+=1 #last loop moved 2 space forward, but next line only move 1 star forward

for i in range (5):

lines.append(space\*fsp+star\*1+space\*msp+star\*1)

fsp-=1

msp+=2

lines.append((star+dash\*29)\*2) # the zero axis.

newlines= lines[:] #hard copy list, pop last str, then reverse list.

newlines.pop()

for line in newlines:

new=(' '\*30)+line

lines\_rvs.append(new)

lines\_rvs.reverse()

for line in lines\_rvs:

lines.append(line)

for line in lines:

print line

linestr=""

for i in range(len (lines)):

starpo=[]

linestr=str(lines[i])

for j in range(len(linestr)):

if linestr[j] == star:

starpo.append(j)

else:

continue

star\_record[i]=starpo

print star\_record