

Python Programming I

Offensive and Defensive Tool Construction

Table of Contents

[Objectives 2](#_Toc477178104)

[Background Reading 2](#_Toc477178105)

[Important Information 2](#_Toc477178106)

[Introduction: Why Python? 3](#_Toc477178107)

[Problem 1 4](#_Toc477178108)

[Problem 2 4](#_Toc477178109)

[Problem 3 4](#_Toc477178110)

[Problem 4 4](#_Toc477178111)

[Problem 5 5](#_Toc477178112)

[Problem 6 5](#_Toc477178113)

[Problem 7 5](#_Toc477178114)

[Problem 8 6](#_Toc477178115)

Offensive and Defensive Tool Construction

Python Programming I

Objectives

This lab focuses on the following objectives:

* Explain the purpose of scripting languages and Python.
* Explore the basic syntax of Python and compare it to C.
* Use variables, expressions and statements in Python.
* Explain function definitions and function calls (pure and with return value).

Background Reading

Read chapters 1–5 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.

# Introduction: Why Python?

Both C and Python are high-level languages, and both need to be translated into machine code to execute. C is compiled whereas Python is interpreted.

Describe the differences between the two using the table below (draw a simple diagram of the path from source code to program execution and output):

|  |  |
| --- | --- |
| **C Program** | **Python Program** |
|  |  |

1. You have learned how to program in C. Examine this simple program in C:

#include <stdio.h>

void main (int argc, char \*\*argv)

{

printf (“Hello World!\n”);

}

1. Write the equivalent program using Python.

Instructor sign-off: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Problem 1

Write a Python program named **m02p01.py** (module 2, problem 1) that outputs a single line consisting of the year, month, date, your SAIT email address, and your first and last name, each separated by a single space. Use the following format (substitute your own name and email for “john smith”).

2016-01-31 john.smith@edu.sait.ca john smith

**import datetime**

**datem = datetime.datetime.now().strftime("%Y-%m-%d")**

**firstname='jun'**

**lastname='wang'**

**email=firstname+'.'+lastname+'@edu.sait.ca '**

**name=firstname+ ' '+lastname**

**print ('%s %s %s'%(datem,email,name))**

# Problem 2

Write a Python program named **m02p02.py** that outputs a single line showing the host name of your computer and how much memory is installed on your computer. (Hint: Look for *MemTotal* in pseudo-file /proc/meminfo.) Use the following format:

Hostname: SuperLinux, Memory: XXXX MB

**import sys, os**

**#find the hostname**

**filename='/etc/hostname'**

**fd=open(filename,'r')**

**hostname=fd.readline().rstrip('\n')**

**#find the memory size**

**filename1='/proc/meminfo'**

**fd1=open(filename1,'r')**

**lines=fd1.readlines()**

**for line in lines:**

**(a,b)=line.split(':',1)**

**if "MemTotal" ==a:**

**words=b.split() #size in KB**

**memorysize=int(words[0])/1024**

**print ("Hostname: %s, Memorysize: %s MB" % (hostname, memorysize))**

# Problem 3

Python has 29 keywords. Write a Python program named **m02p03.py** that outputs this list of keywords (one per line, sorted alphabetically), and then outputs the same list but sorted alphabetically in reverse.

and

as

…

yield

yield

…

as

and

**import keyword**

**keywords=keyword.kwlist**

**#print in alphabetic order**

**for eachkey in keywords:**

**print eachkey**

# Problem 4

Write a Python program named **m02p04.py** that takes two numbers as parameters, and one of the operator words (“plus,” “minus,” “times”), and prints the result of the operation. For example,

**m02p04.py 5 6 times**

prints:

5 x 6 = 30

**import sys,os**

**#sys.argv is a list contains commandline parameters, index from 0**

**if len (sys.argv)<4:**

**print "want some arguments"**

**sys.exit(-1)**

**else:**

**a=int(sys.argv[1]) #a takes the first int**

**b=int(sys.argv[2]) #b takes the second int**

**if sys.argv[3]=='times': #the operation sign**

**print ('%d x %d = %d' % (a,b,a\*b))**

**elif sys.argv[3]=='plus':**

**print ('%d + %d = %d' % (a,b,a+b))**

**elif sys.argv[3]=='minus':**

**print ('%d - %d = %d' % (a,b,a-b))**

# Problem 5

Rewrite the program from Problem 4 so that the user is prompted for the two numbers and the operator after the program starts:

m02p05.py

First Number: **5**

Second Number: **6**

Operation: **times**

5 x 6 = 30

**a=int(raw\_input("First Number: "))**

**b=int(raw\_input("Second Number: "))**

**operation=(raw\_input("Operation: "))**

**if operation =='times':**

**print ('%d x %d = %d' % (a,b,a\*b))**

**elif operation =='plus':**

**print ('%d + %d = %d' % (a,b,a+b))**

**elif operation=='minus':**

**print ('%d - %d = %d' % (a,b,a-b))**

# Problem 6

Rewrite the program from Problem 4 so that it works like a calculator. The user is prompted for the two numbers and an operator after the program starts, and then continues repetitively until the word “exit” is typed:

m02p06.py

Calc: **5 6 times**

5 x 6 = 30

Calc: **3 4 times**

3 x 4 = 12

1. Calc: **7 8 times**
2. 7 x 8 = 56

Calc: **exit**

**while (1):**

**input\_str = raw\_input("Calc: ")**

**if (input\_str == 'exit'):**

**break;**

**else:**

**input\_str=input\_str.split()**

**#split input string to a list of parameters and get riddle of the space after each parameter**

**num1= int(input\_str[0])**

**num2= int(input\_str[1])**

**operation=(input\_str[2])**

**if operation == 'times':**

**print ('%d x %d = %d' % (num1,num2,num1\*num2))**

**elif operation == 'plus':**

**print ('%d + %d = %d' % (num1,num2,num1+num2))**

**elif operation == 'minus':**

**print ('%d - %d = %d' % (num1,num2,num1-num2))**

# Problem 7

Write a Python program named **m02p07.py** that prints a list of all users on your Linux system who have a home directory in the format **/home/username**, and print a pretty table, containing the username using 20 characters and aligned left, and the home directory. (Hint: Make use of the file /etc/passwd. The table borders are made up of plus signs, dashes, and vertical bars.)

m02p07.py

+-----------+---------------------------+

| user | /home/user |

| fred | /home/fred |

+-----------+---------------------------+

Write a function named print\_user\_line(username) to print each individual user line. The function will print the table top or bottom border line if the username is None.

**import sys, os**

**#this function for printing table of usernames and the related home directory**

**def user\_line(username):**

**borderline='+'+'-'\*11+'+'+'-'\*27+'+'**

**print borderline**

**for usr in username:**

**print ("| %-10s| %-26s|" %(usr, '/home/'+usr))**

**print borderline**

**#find the username in file '/etc/passwd'**

**filename='/etc/passwd'**

**fd=open(filename,'r')**

**# every user line ends with /bin/bash, so does root.**

**# Root has no /home/ directory**

**usr\_list=[]**

**lines=fd.readlines()**

**for line in lines:**

**if ('/bin/bash' in line) and ('root' not in line):**

**line=line.split(':')**

**usr\_list.append(line[0]) #usrname is in the first column**

**user\_line(usr\_list)**

# Problem 8

Install Python module **pyutmp**:

sudo pip install pyutmp

if pip is not installed:

sudo apt install pip

Write a Python program named **m02p08.py** that prints all user login/logout information for the past 24 hours. Use the sequence: date, user, where from, sorted in descending time order:

2016-11-07 14:57:39 fred /dev/pts/24

2016-11-04 16:34:59 bob server.sait.ca

2016-10-02 01:03:45 alice somewhere.google.com

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

**import sys,os**

**import time**

**from pyutmp import UtmpFile**

**logs = []**

**for utmp in UtmpFile():**

**if utmp.ut\_user\_process:**

**#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)**

**logs.append(str (time.strftime('%Y-%m-%d %H:%M:%S',time.strptime(time.ctime(utmp.ut\_time))))+' '+str( utmp.ut\_user )+' '+str( utmp.ut\_line))**

**for i in range((len(logs)-1),-1,-1): #reverse list from ascending to descending**

**print logs[i]**