CISC 333- LAB 1

REVIEW PYTHON PROGRAMMING LANGUAGE & USING PYTHON

REVIEW TOOLS: PYLINT, GIT, & ATOM EDITOR

This lab contains the following projects and activities:

Project 1.1	Write Simple Python Programs
Project 1.2	Writing Complex Python Program using Functions
Project 1.3	Read and Write from a file
Lab Review Questions	
Post-Lab Cleanup	None

BEFORE YOU BEGIN

Lab 1 is based you have basic Python programming skills or you have started or completed a Python programming course, online or classroom. Such as online course:

https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-1-python-basics?ex=1

NOTE	*In this lab, you will program in Python. You will prepare for complex programming needed for Defensive programming and analyzing Encryption algorythms*

SCENARIO

You will use the HU Lab IT laptops. The laptops are pre-installed with Python, Atom editor, and programming tools.

After completing this lab, you will be able to:

- Write, debug, and execute simple Python programs
- Use the Atom editor
- Debug with Pylint
- Use Python functions
- Read and Write to a file from within Python

Directions	Create the following Python Programs and confirm results with Lab Assistant or course Instructor
Outcomes	After completing this exercise, you will know how to: ▲ Write Python programs ▲ Use Atom editor to create and debug Python programs.
Completion time	60 minutes
Precautions	Be attentive while writing your programs

■ PART 1.1:.

```
print 'Hello, world!'
 name = raw_input('What is your name?\n')
print 'Hi, %s.' % name
friends = ['john', 'pat', 'gary', 'michael']
for i, name in enumerate(friends):
    print "iteration {iteration} is {name}".format(iteration=i, name=name)
```

Question 1	*Describe the output from the program.*


```
parents, babies = (1, 1)
while babies < 100:
```

```
    print 'This generation has {0} babies'.format(babies)
    parents, babies = (babies, parents + babies)

def greet(name):
    print 'Hello', name
greet('Jack')
greet('Jill')
greet('Bob')
```

Question 2	*Describe the output from the program.*

```
import re
for test_string in ['555-1212', 'ILL-EGAL']:
    if re.match(r'^\d{3}-\d{4}$', test_string):
        print test_string, 'is a valid US local phone number'
    else:
        print test_string, 'rejected'


prices = {'apple': 0.40, 'banana': 0.50}
my_purchase = {
    'apple': 1,
    'banana': 6}
grocery_bill = sum(prices[fruit] * my_purchase[fruit]
                   for fruit in my_purchase)
print 'I owe the grocer $%.2f' % grocery_bill
```

Question 3	*Describe the output from the program.*

.

```
# This program adds up integers in the command line
import sys
try:
    total = sum(int(arg) for arg in sys.argv[1:])
    prinyt 'sum =', total
except ValueError:
    print 'Please supply integer arguments'
```


Question 4	*Describe the output from the program*

■ PART 1.2: Classes, Unit testing, Itertools, Random Number Generator

```
from time import localtime
activities = {8: 'Sleeping',
              9: 'Commuting',
              17: 'Working',
              18: 'Commuting',
              20: 'Eating',
              22: 'Resting' }
time_now = localtime()
hour = time_now.tm_hour
for activity_time in sorted(activities.keys()):
    if hour < activity_time:
        print activities[activity_time]
        break
elsse:
    print 'Unknown, AFK or sleeping!'
```

Question 5	*Describe the output from the program*

```
 classs BankAccount(object):
    def __init__(self, initial_balance=0):
        self.balance = initial_balance
    def deposit(self, amount):
        self.balance += amount
    def withdraw(self, amount):
        self.balance -= amount
    def overdrawn(self):
        return self.balance < 0
my_account = BankAccount(15)
my_account.withdraw(5)
print my_account.balance
```

Question 6	*Describe the output from the program*


```
import itertools
def iter_primes()::
    # an iterator of all numbers between 2 and +infinity
    numbers = itertools.count(2)
    # generate primes forever
    while True:
```

```
        prime = numbers.next()
        yield prime
        # this code iteratively builds up a chain of
        # filters...slightly tricky, but ponder it a bit
        numbers = itertools.ifilter(prime.__rmod__, numbers)
for p in iter_primes():
    if p > 1000:
        break
    print p
```

Question 7	*Describe the output from the program*

```
import random
guesses_made = 0
name = raw_input('Hello! What is your name?\n')
number = random.randint(1, 20)
print 'Well, {0}, I am thinking of a number between 1 and 20.'.format(name)
while guesses_made < 6:
    guess = int(raw_input('Take a guess: '))
    guesses_made += 1
    if guess < number:
        print 'Your guess is too low.'
    if guess > number:
        print 'Your guess is too high.'
    if guess == number:
        break
if guess == number:
    print 'Good job, {0}! You guessed my number in {1} guesses!'.format(name,
        guesses_made)
else:
    print 'Nope. The number I was thinking of was {0}'.format(number)
```

Question 8	*Describe the output from the program*

■ PART 1.3: Opening files,

```
# indent your Python code to put into an email
import glob
# glob supports Unix style pathname extensions
python_files = glob.glob('*.py')
for file_name in sorted(python_files):
    print '    ------' + file_name
    with open(file_name) as f:
        for line in f:
            print '    ' + line.rstrip()
    print
```

Question 9	*Describe the output from the program*