**Lab: While Loops and Conditional Statements**

Work on these problems in class. Work on your Python code in IDLE. When you have finished each problem, **please copy and paste your Python code into this Word document**. Then when you are done, or at the end of class, **save and submit this worksheet to this week’s Classroom Labs dropbox**.

**Comment your code – you should write comments for every program!**

Work with your lab group on these problems. Start by going through the Loops and Conditionals Powerpoint. Please ask other groups, classmates or the instructor for help if you are stuck or have any questions.

**Part 1: If statements**

Write a program which asks you to enter an integer number of cents.

Your program should then tell you if you have less than a dollar, more than a dollar, or a dollar exactly.

Paste your code here.

''' Program: Problem #1 - I got a dollar!

Author(s): Tom Stutler, Seth Stoxen, Ashton Kelley

Last Date Modified: 9/29/14

The intent of this program is to prompt the user for a number of cents they have and return whether or not they have a dollar.

'''

#Define function to see if you have less than a dollar, exactly one dollar, or you have more than a dollar.

def dollar(cents):

‘’’ (int) -> str

This function takes in the users entered cents, calculates if they have a dollar, and tells the user if they do.

‘’’

if cents > 100:

print ("You have more than one dollar!")

elif cents == 100:

print ("You have exactly one dollar!")

elif cents < 100:

print ("You don't quite have a dollar.")

#Prompt user for the amount of cents they have and assign to a variable.

cents = int(input("Enter how many cents you have: "))

#Use dollar() to calculate how many dollars, if any, the user has.

dollar(cents)

Now, modify your program. It should do the same as before, but print an error message if the user enters a negative number. Test your program.

Paste your code here.

''' Program: Problem #1 - I got a dollar!

Author(s): Tom Stutler, Seth Stoxen, Ashton Kelley

Last Date Modified: 9/29/14

The intent of this program is to prompt the user for a number of cents they have and return whether or not they have a dollar.

'''

#Define function to see if you have less than a dollar, exactly one dollar, or you have more than a dollar.

def dollar(cents):

‘’’ (int) -> str

This function takes in the users entered cents, calculates if they have a dollar, and tells the user if they do.

‘’’

if cents > 100:

print ("You have more than one dollar!")

elif cents == 100:

print ("You have exactly one dollar!")

elif 0 < cents < 100:

print ("You don't quite have a dollar.")

else:

print ("Please retry and enter a positive amount of cents.")

#Prompt user for the amount of cents they have and assign to a variable.

cents = int(input("Enter how many cents you have: "))

#Use dollar() to calculate how many dollars, if any, the user has.

dollar(int(input("Enter how many cents you have: ")))

**Part 2: Random**

Type in the code for the guessing program found in D2L. Run and test the program.

*Add comments to the program* and then paste your Python code here.

‘’’Program: Problem #2 – Random number between two entered.

Author(s): Tom Stutler, Seth Stoxen, Ashton Kelley

Last Date Modified: 9/30/14

The intent of this program is to have the user enter two numbers. The program then generates a random number bewtween the two user entered numbers and prompts the user to guess the randomly generated number. If the guess is incorrect it will tell tell the user “Too small” or “To large”. Once the user guesses correctly it will tell them they’re correct and how many guesses it took.

‘’’

import random

smaller = int(input("Enter the smaller number: "))

larger = int(input("Enter the larger number: "))

#This is creating a random number between (and including) the two number that are input)

myNumber = random.randint(smaller, larger)

#This is the base number for how many guesses have been made.

count=0

while True:

#This number goes up every time the guesser guesses. Every time that the answer is wrong, the counter goes up by one by looping back throught the function.

count += 1

userNumber = int(input("Enter your guess: "))

if userNumber <myNumber:

print("Too small")

elif userNumber > myNumber:

print ("Too large")

else:

#this shows how many guesses it took to find the random number.

print("Congratulations! You've got it in", count, "tries!")

break

**Part 3: While loops**

Write a program which picks a random card from a standard deck, until you tell it to stop.

Your program will use a while true loop. Every time the loop runs, it will display a random number between 1 and 10, or jack, queen, king; and a suit – hearts, clubs, spades or diamonds.

Then it should ask you if you want to pick another random card. Use what the user types to figure out whether to break out of the loop, or to repeat again.

Hint: start by writing a program that can display a random number between 1 and 13.

Next, can your loop also pick a random number between 1 and 4 – then can you use an if statement to turn 1 into hearts, 2 into clubs, 3 into spades, 4 into diamonds?

Then, can turn the numbers 11, 12 and 13 into J, Q and K?

"""

Program: Problem #3 – Draw a card!

Author(s): Tom Stutler, Seth Stoxen, Ashton Kelley

Last Date Modified: 09-30-14

This program will ask a player if they want a card. If som it will generate a random card with card face and suit names included. It will then ask them if they want another card or not.

"""

import random

def draw\_card():

"""This is the function to choose a card."""

#Retry gets reassigned at the end of the upcoming while loop.

retry = input("Would you like to draw a card?")

#This loops around if the person wants a different card at the end.

while retry == "y":

#This chooses a random number from 1-13 and then assigns it to a card.

cardNum = random.randint(1, 13)

if cardNum == 1:

cardNum = str("Ace")

if cardNum == 11:

cardNum = str("Jack")

if cardNum == 12:

cardNum = str("Queen")

if cardNum == 13:

cardNum = str("King")

#This chooses a random number from 1-4 and then assigns it to a suit.

cardSuit = random.randint(1, 4)

if cardSuit == 1:

cardSuit = str("Hearts")

if cardSuit == 2:

cardSuit = str("Diamonds")

if cardSuit == 3:

cardSuit = str("Spades")

if cardSuit == 4:

cardSuit = str("Clubs")

print ("Your card is the", cardNum, "of", cardSuit,)

retry = input("Would you like a different card? (y/n)")

#this takes the input from the retry variable to decide if it wants to start over or not. It will start over if they say y but break if they input n. Anything else will result in yelling at them and then starting over anyway.

if retry == "y":

True

elif retry == "n":

break

else:

print ("You need to enter 'y' or 'n'")

main()

def main():

"""This is to get the whole thing started. Its existence is to be able to loop if the user doesnt enter a 'y' or an 'n' at the end of the drawn\_card function."""

draw\_card()

main()

**Part 4: (each group member should complete this part – but still work together)**

Write a Python program for the following projects, adapted from *Fundamentals of Python* by Kenneth Lambert. Start by thinking through the problem, come up with some examples, and pseudocode (at least an outline) use these examples to verify your program. Remember to comment your code.

Project 2, page 118:

Write a program that accepts the lengths of three sides of a triangle as inputs. The program output should indicate whether or not the triangle is a right triangle.

Project 3, page 118

Modify the guessing game from part 2 so the user thinks of a number that the computer must guess. The computer must make no more than the minimum number of guesses.

Project 6, page 119

The German mathematician Gottfried Leibniz developed the following method to approximate the value of π:

π/4 = 1 – 1/3 + 1/5 - 1/7 + …

Write a program that allows the user to specify the number of iterations used in this approximation and display the resulting value.

**CSci 1500 - Assignment 6 – 20 pts.**

For the next 3 problems, Write a *detailed* pseudocode solution for each of the following problems. Then, using your pseudocode, convert your pseudocode into Python programs. Here are the things you need to do for each problem:

* Design your solutions to produce the program output to work correctly for any valid user input.
* Convert your pseudocode into Python. Remember to give data types for all variables. Remember to include your name in a comment at the top of the program. Run the program and verify that it works properly for a variety of input values.

**What you need to turn in:** 1) Python script \*.py file 2) Python script saved as a \*.txt 3) console output saved as a \*.txt file showing the program run with various inputs. 4) Pseudocode file. Use a word processor to write and save your pseudocode.

HW Problem #1) Project 8, page 119

The greatest common divisor of two positive integers, A and B, is the largest number that can be evenly divided into both of them. Euclid’s algorithm can be used to find the greatest common divisor (GCD) of two positive integers. You can use this algorithm in the following manner:

1. Compute the remainder of dividing the larger number by the smaller number.
2. Replace the larger number with the smaller number and the smaller number with the remainder.
3. Repeat the process until the smaller number is zero

The larger number at this point is the GCD of A and B. Write a program that lets the user enter two integers and then prints each step in the process of using Euclidean algorithm to find their GCD.

HW Problem #2) Project 9, page 120

Write a program that receives a series of numbers from the user and allows the user to press the enter key to indicate he or she is finished providing inputs. After the user presses the enter key, the program should print the sum of the numbers and their average.

HW Problem #3)

Rapid Delivery charges by weight for delivery of packages. The delivery charge for the first pound (16 ounces) is $3.00 and $0.50 is added onto the charge for each additional four ounces. For example, a package weighing more than 16 but at most 20 ounces costs $3.50 to deliver; a package weighing more than 20 but at most 24 ounces costs $4.00 to deliver; etc. Write a program that inputs the weight of a package in ounces (an integer value), then outputs the weight and the charge for delivery. Use an int-valued, data input function, **get\_weight**, to prompt for and input the package weight, a float-valued function, **delivery\_charge**, to calculate and return the delivery charge, and a function, **display\_charge**, to display the weight, in pounds and ounces, and the delivery charge. Display the charge with a dollar sign and two places of accuracy.

**Hint:** If the weight of a package in ounces, *w*, is over 16 ounces, then the number of additional weight charges equals [(*w*-13)/ 4], where “/” is integer division (choose the correct Python symbol)

.

Here is an example of what output should look like from running your program.

Enter package weight(oz): **23**

Package weight = 1 lb. 7 oz.

Delivery charge = $4.00

+-\*