Orange County Community College

Computer Science and Technology Department

csc 138 - Scripting

Laboratory Exercise 5

Name: ___Alex DeGhetto_____

Date: _____

Objectives:

- Demonstrate an understanding of iteration structures in Python.
- Utilize looping structures to solve problems.
- Utilize dictionaries to store data

Procedures:

1. Examine the following program:
Use the space below to trace this program, giving the state of the variables, along with its output:

```
#!/usr/bin/python3

for x in range(10):
    for y in range(x):
        print("*",end='')
    print()
```

For statement of X being in the range of 10

for each value of x, y will print * that many times

Output:

```
*
**
***
****
*****
******
*******
********
*********
```

2. Alter the program above so that it prints the following pattern:
Print the code for your new program and attach it to this lab sheet.

```
**********
*********
********
*******
******
*****
****
***
**
*
```

```
#!/usr/bin/python3
n=10
for i in range(n):
  for j in range(i,n):
    print("*", end='')
  print()
```

3. Using what you have learned above, create a program that prints the following pattern:
Print the code along with the output of the program and attach it to this lab sheet.


```
#!/usr/bin/python3
rows = int(input("Enter number of rows: "))

k = 0

for i in range(1, rows+1):
    for space in range(1, (rows-i)+1):
        print(end="  ")

    while k!=(2*i-1):
        print("* ", end="")
        k += 1

    k = 0
    print()
```

4. Using what you have learned above, create a program that prints the following pattern:
Print the code along with the output of the program and attach it to this lab sheet.


```
#!/usr/bin/python3
rows = int(input("Enter number of rows: "))

k = 0

for i in range(1, rows + 1):
  for space in range(1, (rows - i) + 1):
    print(end=" ")
  print("*", end="")
  while k < (2 * i - 3):
    print(" ", end="")
    k += 1

  k = 0
  if i != 1:
    print("*")
  else:
    print(" ")
```

4a. Create a **while** loop that reads a list of 10 temperatures. You will create the values for the temperatures and have print statements indicating if the temperature is in spring, summer, winter or fall.

Spring temps are from 61-85 then print("Spring temp", variable)

Summer temps are greater than 85 then print("Summer temp", variable)

Fall temperatures are from 40-60 then print("Fall temp", variable)

Winter temperatures are below 40 then print("Winter temp", variable)

4b

```
import random
x= random.randrange(1,50)
print(x)
```

Use the above code to generate a random number between 1 and 50.
Create a loop that will run until the user guesses the correct number.
Print out the number of tries that the user took to guess the number.
As the user is guessing the number give them a hint as to whether they are too high or low with their guesses.

5. A Python ***dictionary*** maps a ***key*** to a ***value***. For example, examine the following Python dictionary:

```
grades = {
        'bob': 89,
        'sue': 98,
        'jack': 68,
        'jill': 77
}
```

This dictionary maps student names (the keys) with their corresponding grades (the values). In this example, the keys are *strings* and the values are *integers*. Note the syntax. The key, value pairs are enclosed in curly braces and separated by commas. Type the grades dictionary into your Python script. Then enter the statement:

```
print( grades )
```

What is returned by your Python script?

{'bob': 89, 'sue': 98, 'jack': 68, 'jill': 77}

6. We access an element in a dictionary by referencing its key. For example, to find bob's grade we would issue the command:

```
print( grades['bob'] )
```

Note that we still have to enclose the key in quotes (since it is a string). Add the above statement into your Python script. What does it return?

89

Keys

7. We can obtain a list of the keys for a dictionary with the keys function. In your Python script, add the statement:

grades.keys()

What is returned by the Python script?

dict_keys(['bob', 'sue', 'jack', 'jill'])

8. Add **list(grades.keys())**

What does this return ?

['bob', 'sue', 'jack', 'jill']

Iterating over keys

9. As noted previously, we can iterate over the keys in our dictionary using the for construction. In your Python 3 script, issue the statement:

```
for  student in grades.keys():
        print( student )
```

What is returned?

bob
sue
jack
jill

10. Below, write a set of statements that will print out the student's name and their corresponding grade. Type in and execute your statements to ensure they function properly.

Changing the value of a key

11. To change the value of a key, simply assign a new value to the existing key. For example, print out the keys and values in the grades dictionary. How did you go about this?

print(grades)

What are the current keys and values?

{'bob': 89, 'sue': 98, 'jack': 68, 'jill': 77}

Change bob's grade by issuing the following statement:

grades['bob'] = 85

Again, print out the keys and values of the grades dictionary. Has the value for the key bob changed?
{'bob': 85, 'sue': 98, 'jack': 68, 'jill': 77}
his grade went to 85

Adding keys

12. You can add a key to a Python dictionary by assigning a value to key that does not already exist. For example, issue the following statement in your Python script

grades[‘jim’] = 99

Print out the grades dictionary and list the keys and values below:

{'bob': 89, 'sue': 98, 'jack': 68, 'jill': 77, 'jim': 99}

13. Add two more keys to the grades array. List the statements you executed below:

grades[‘tom’] = 76
grades[‘sam’] = 88

Removing keys

14. We can remove a key from a dictionary with the del statement. For example, to remove jack from the grades dictionary, issue the following statement:

del grades[‘jack’]

Print out the grades dictionary and list the keys and values below:

{'bob': 89, 'sue': 98, 'jill': 77, 'jim': 99, 'tom': 76, 'sam': 88}

‘Length’ of a dictionary

15. We can determine how many elements (keys) a dictionary has with the len command. In your Python script add the command:

len(grades)

What is the output of this command?

6

16. Write a complete Python script that uses some combination of dictionaries and lists to store 5 student names and their accompanying test scores. Make sure each student has at least 4 test scores. The program should iterate over the students in the list and print the average for each student. Attach a copy of your program to this lab sheet.

17. Alter the program you created in #5 above to prompt the user for a student name. Once the user has entered a name, select the appropriate set of student grades, and calculate that student's average, printing it to the screen. Attach a copy of your program to this lab sheet.

18. Alter the script you wrote above to allow the user to add another student (and his/her grades) to the dictionaries/lists that hold your student data. Make sure to read in the student name, along with some number of grades. The user should be able to search for the new student's name via the code you wrote in #6 above.

19. Use the code you have written in #s 5-7 above to write a grade book application that minimally offers the user the following menu:

1. Print the entire grade book (student names and grades).
2. Search for a student and print their grades.
3. Search for a student and print their average.
4. Add a new student and their grades.
5. Add a new grade for a student.
6. Exit.

The program should continue to execute until the user terminates by selecting choice 6. If choice 1 is selected, the program should print all student names along with their grades. If the user selects choice 2, the program should prompt for a student name and print that student's grades. If choice 3 is selected, the script should prompt for a student's name and print that student's average. If the user selects choice 4, the program should prompt the user for a new student name and their grades and add them to the existing data. If choice 5 is selected, the script should prompt the user for a student name and a new grade and add that grade to the data for that student.

Make sure to write a complete Python script. Add judicious comments to your code.

When you have finished, print out your code and attach it to this lab sheet.

```
#!/usr/bin/python3
# Define global variables
gradeDictionary = {}
runScript = True

# Function for yes or no question to user
def yesOrNo(userInput):
  if (userInput == "n" or userInput == "N"):
      return False
  else:
    if (userInput != "y" and userInput != "Y"):
      print("You didnt enter a y or an n")
    return True

# Start the script
while (runScript):
  # Print the menu
```

```
mainMenuInput=int(input("""
1. Print the entire grade book (student names and grades)
2. Search for a student and print their grades.
3. Search for a student and print their average.
4. Add a new student and their grades.
5. Add a new grade for a student.
6. Exit.

Enter an option: """))

# Print the entire grade book
if (mainMenuInput == 1):
  print(gradeDictionary)

# Search for a student and print their grades.
elif(mainMenuInput == 2):
  studentName=input("Enter the student's name: ")
  print(gradeDictionary[studentName])

# Search for a student and print their average.
elif(mainMenuInput == 3):
    studentName=input("Enter the student's name: ")
    print(round(sum(gradeDictionary[studentName]) / len(gradeDictionary[studentName]), 2))

# Add a new student and their grades.
elif(mainMenuInput == 4):
  studentName=input("Enter the student's name: ")
  gradeNumber = 1
  inputGrade = True
  grades = []
  while (inputGrade):
    grade=float(input(f"What is grade {gradeNumber}: "))
    grades.append(grade)
    enterAnotherGrade=input("Would you like to enter another grade (y/n): ")
    inputGrade = yesOrNo(enterAnotherGrade)
    gradeNumber += 1
  gradeDictionary.update({studentName: grades})

# Add a new grade for a student.
elif(mainMenuInput == 5):
  studentName=input("Enter the student's name: ")
  inputGrade = True
  while(inputGrade):
    grade=float(input(f"What is grade {len(gradeDictionary[studentName]) +1}: "))
    gradeDictionary[studentName].append(grade)
    enterAnotherGrade=input("Would you like to enter another grade (y/n): ")
    inputGrade=yesOrNo(enterAnotherGrade)

# Exit the program
elif(mainMenuInput == 6):
  finishScript=input("Would you like to go back to the main menu (y/n): ")
  runScript = yesOrNo(finishScript)
```

20. Write a program that will take a user input as a string and count the number of distinct letters in that string.
Ex: “college” should print out that there are 2 – “l”
1 =’o’
2 = ‘e’
1 = ‘g’
1=’c’

```
#!/usr/bin/python
from collections import Counter
# get input from user for string
mystring = input("Enter a word: ")
cntstr = Counter(mystring)
print(cntstr[mystring])   # -> returns the count of that character
print(cntstr.most_common())  # -> returns the three most common characters
```