Topics covered

Here is a quick outline of what will be covered on Exam #2. The list below contains only the new material covered since exam #1. You need to know all the material that we covered before exam #1 as well.

Condition Controlled Loops (while loops)

1. mechanics & how they work
2. setting up conditions for a while loop
3. infinite loops and how to avoid them
4. sentinels (defining a value that the user enters that causes the loop to end)
5. input validation loops (asking the user to continually enter a value until that value matches some condition)

Accumulator variables

1. setting up and using accumulator variables
2. self referential assignment statements (i.e. counter = counter + 1)
3. augmented assignment operators (i.e. counter += 1)

Lists

1. Simple Variables vs. Lists (simple variables can only hold one piece of data, but lists can hold multiple values) – you can think of a list like a “book” and a variable like a “sheet of paper”
2. Defining lists in Python (i.e. mylist = [1,2,3])
3. Concatenating lists with the “+” operator
4. Repeating lists with the “*” operator
5. Referencing list items using index notation (i.e. mylist[0])
6. Iterating through a list using a “while” loop
7. Iterating through a list using a “for” loop
8. Using the len() function to determine the # of items in a list
9. Updating the value of an item in a list using bracket notation
10. Creating empty lists
11. Finding an item in a list using the “in” operator
12. Adding items to a list using the append method
13. Sorting items in a list using the sort method
14. Finding the position of an item in a list using the index method
15. Inserting an item in a list at a specific index using the insert method
16. Finding the largest and smallest values in a list using the min and max methods
17. Totaling the values of all elements in a list using an accumulator variable
18. Removing an item from a list using the remove method

Generating random numbers

1. random package
2. using random.randint() function
3. selecting a random element from a list
4. creating random names from lists of first and last names

Basic statistics and their meaning

1. average/mean value as a measure of central tendency
2. meadian value as a measure of central tendency

3. min and max (or range) values as measures of dispersion
4. standard deviation value as a measure of dispersion

Probabilities

1. enumerating all possible outcomes of an experiment
2. determining the number of all possible outcomes
3. calculating probabilities of a given outcome

Sample problems

Tracing output

Trace the output of the following programs (each code fragment below is a separate Python program.) For solutions simply run the programs.

1.

```
count = 0

while count < 10:
    print ('Hello')
    count += 1

x = 10
y = 0
```

2.

```
while x > y:
    print (x, y)
    x = x - 1
    y = y + 1

keepgoing = True
x = 100
```

3.

```
while keepgoing:
    print (x)
    x = x - 10
    if x < 50:
        keepgoing = False
```

4.

```
x = 0
while x ** 2 > 36:
    x += 1
    if x % 2 == 1 :
        print (x, x ** 2)

    print ("looping!")
```

5.

```
pokemon = ["squirtle", "pikachu", "charmander", "bulbasaur", "meowth"]

for i in range(0, len(pokemon), 2):
    print (i, pokemon[i])
```

6.

```
pokemon = ["squirtle", "pikachu", "charmander", "bulbasaur", "meowth"]
counter = 0

while counter < len(pokemon):
    print (counter, pokemon[counter])
    counter += 2
```

7.

```
prices = [10,20,30,40,50]

for price in prices:
    price = price * 1.10
    print(price, end=', ')
print (prices)
```

8.

```
prices = [10,20,30,40,50]

for i in range(0, len(price)):
    prices[i] = prices[i] * 1.10
    print(prices[i], end=', ')
print (prices)
```

9.

```
prices = [10,20,30,40,50]
```

```
sum = 0
for price in prices:
    sum +=  price
print (sum/len(prices))
```

10.

```
prices = [110,20,75,140,50,90,15]
prices.sort()
print ( prices[ len(prices)/2 ] )
```

11. (this is a challenging one)

```
    x = [1,2,3]
    counter = 0
    while counter < len(x):
        print (x[counter] * '%')
        for y in x:
            print (y * '*')
        counter += 1
```

Programming Problems

1. Write a program that prints out all even numbers between 1 and 100,000.
2. Write a program that continually asks the user for a series of prices. Keep track of the running total as the user enters prices. Prompt the user after each price to see if they wish to continue entering prices. When they are finished, print out the total amount entered.
3. Write a program that asks the user for a number of days. Then prompt the user for their weight on each day they specified (i.e. if they enter 7 you should ask them for 7 weight values). When they are finished print out their average weight for the period in question. Sample running:

   ```
   Enter a number of days: 3
   Day 1: Enter a weight: 180
   Day 2: Enter a weight: 175
   Day 3: Enter a weight: 170

   Average weight for 3 days: 175.0
   ```

4. Re-write the previous program so that you re-prompt them if the user enters an invalid weight y(anything 0 or less). Ensure that you don't move to the next day without getting a valid weight. Sample running:

   ```
   Enter a number of days: 3
   Day 1: Enter a weight: 180
   Day 2: Enter a weight: 0
   Invalid, try again!
   Day 2: Enter a weight: -5
   ```

```
Invalid, try again!
Day 2: Enter a weight: 175
Day 3: Enter a weight: 170

Average weight for 3 days: 175.0
```

5. You are working for a fireworks company that wants to try out a new marketing strategy. They would like to try a new promotion that gives away free fireworks to customers who purchase a certain quantity – the more you buy, the more free fireworks you get! Here is their current promotion:

```
Buy between 1 and 10 fireworks, get 0 free
Buy between 11 and 20 fireworks, get 1 free
Buy between 21 and 30 fireworks, get 3 free
Buy more than 31 fireworks, get 5 free
```

   Write a program that givn a list of orders for fireworks, creates a new list with the actual fireworks ordered and additional promotional fireworks added to the quantity. For example, if the original list or ordered quantities is :

```
[10, 50, 30, 7, 15]
```

   then the quantities with the added free fireworks should be

```
[10, 55, 33, 7, 16]
```

   You can make-up the list for your program.

6. Write a program that continually asks the user for a grade value. You can assume the user is finished entering grades when they enter in an invalid number (i.e. one greater than 100 or less than 0). Calculate the average, median, min and max scores. After the user finishes with their grades output the calculated statistics and a list of the equivalent letter grades. Here is a sample running of this program:

```
Enter a grade:  90
Enter a grade:  70
Enter a grade:  85
Enter a grade:  75
Enter a grade:  80
Enter a grade:  200

Average: 80
Median:  80
Min: 70
Max: 90
Letter grades: A, C, B, C, B
```

   Letters grades can be determined using the following conversions:

```
100-90: A
80-89.99: B
70-79.99: C
65-69.99: D
Lower than 65: F
```

7. Write a program that generates an N-number for a new NYU student. The N-number consists of 9 characters. The first letter is always 'N' and the remaining 8 characters are digits. The first digit cannot be zero, the other digits can be anything.

   Answer the following questions about the N-numbers that your program is generating (these are not programming questions):

   - What is the total number of unique N-numbers that can be generated?
   - What is the total number of N-numbers that have 9 as their last digit?
   - What is the probability of getting N-number with the last digit equal to 9?
   - What is the probability of getting N-number with the last three digits equal to 9?

8. Write a program that generates a random series of lottery numbers. Your program should generate 5 unique random numbers between 1 and 100. Numbers should not be repeated within a series and should be printed in ascending order. Here are three sample executions of the program:

```
Your lottery #'s are: [19, 61, 62, 78, 99]
----
Your lottery #'s are: [1, 41, 64, 66, 78]
----
Your lottery #'s are: [19, 20, 28, 41, 97]
```

9. Assume you are tossing three fair coins. What is the probability of getting three heads? What is the probability of getting exactly 2 heads? What is the probability of getting at least two heads?

   Write a program that attempts to verify your answers to the above questions.

Solutions

Question #1

```
for i in range(2, 100001, 2):
    print (i)
```

Question #2

```
total = 0
again = "yes"
while again == "yes":
    p = float(input("Enter a price: "))
    total += p
    again = input("Enter another price? (yes or no): ")
```

Question #3

```
total = 0
days = int(input("Enter a number of days: "))
for day in range(days):
    prompt = "Day " + str(day) + ": Enter a weight: "
    weight = input(prompt)
    total += weight

print ("Average weight for", days, "days:", total/days)
```

Question #4

```
total = 0
days = int(input("Enter a number of days: "))
for day in range(days):
    while True:
        prompt = "Day " + str(day) + ": Enter a weight: "
        weight = input(prompt)
        if weight > 0:
            total += weight
            break
        else:
            print ("Invalid, try again")

print ("Average weight for", days, "days:", total/days)
```

Question #5

```
def myfun(a,b):
    print a**b

myfun(5,2)
```

Questions #6 and #7

```
def get_num_free_fireworks(num):
    if num <= 10:
        return 0
    elif num <= 20:
        return 1
    elif num <= 30:
        return 3
    else:
        return 5



while True:
    q = int(input("Enter a quantity: "))
    free = get_num_free_fireworks(q)
```

```
    print ("You are eligible for", free, "free fireworks!")
    again = input("Would you like to run the program again?")
    if again == "no":
        break
```

Questions #8 and #9

```
def compute_grade(g):
    if g >= 90:
        return 'A'
    elif g >= 80:
        return 'B'
    elif g >= 70:
        return 'C'
    elif g >= 65:
        return 'D'
    else:
        return 'F'


while True:

    g = int(input("Enter a grade: "))
    if g > 100 or g < 0:
        break

    letter = compute_grade(g)
    print (g, "is a(n)", letter)
```