Code Academy

Chapter 4: Loops (4.8 – 4.9)

PART4:PROCESSING STRINGS, RANDOM NUMBERS AND SIMULATIONS

Code Academy

Chapter 3-Part4:

- **Goals**
  - To implement programs that read and process data sets
  - To use a computer for simulations

- **Contents:**
- Processing Strings
- Application: Random Numbers and Simulation

Processing Strings

Code Academy

- A common use of loops is to process or evaluate strings.
- For example, you may need to count the number of occurrences of one or more characters in a string or verify that the contents of a string meet certain criteria.

Processing Strings Examples:

- Counting Matches
- Finding All Matches
- Finding the First or Last Match
- Validating a String
- Building a New String

Counting Matches

- Suppose you need to count the number of uppercase letters contained in a string.
- We can use a for loop to check each character in the string to see if it is upper case
- The loop below sets the variable **char** equal to each successive character in the string
- Each pass through the loop tests the next character in the string to see if it is uppercase

```
uppercase = 0
for char in string :
   if char.isupper() :
      uppercase = uppercase + 1
```

Counting Vowels

- Suppose you need to count the vowels within a string
- We can use a for loop to check each character in the string to see if it is in the string of vowels “aeiuo”
- The loop below sets the variable **char** equal to each successive character in the string
- Each pass through the loop tests the lower case of the next character in the string to see if it is in the string “aeiou”

```
vowels = 0
for char in word :
   if char.lower() in "aeiou" :
      vowels = vowels + 1
```

Finding All Matches Example

- When you need to examine every character in a string, independent of its position we can use a for statement to examine each character
- If we need to print the position of each uppercase letter in a sentence we can test each character in the string and print the position of all uppercase characters
- We set the range to be the length of the string
  - We test each character
  - If it is uppercase we print I, its position in the string

```
sentence = input("Enter a sentence: ")
for i in range(len(sentence)) :
   if sentence[i].isupper() :
      print(i)
```

Finding the First Match

- This example finds the position of the first digit in a string.

```
found = False
position = 0
while not found and position < len(string) :
   if string[position].isdigit() :
      found = True
   else :
      position = position + 1

if found :
   print("First digit occurs at position", position)
else :
   print("The string does not contain a digit.")
```

Finding the Last Match

- Here is a loop that finds the position of the last digit in the string.
- This approach uses a while loop to start at the last character in a string and test each value moving from the end of the string to the start of the string
  - Position is set to the length of the string - 1
  - If the character is not a digit, we decrease position by 1
  - Until we find a digit, or process all the characters

```
found = False
position = len(string) - 1
while not found and position >= 0 :
   if string[position].isdigit() :
      found = True
   else :
      position = position - 1
```

Validating a String

- In the United States, telephone numbers consist of three parts—area code exchange, and line number—which are commonly specified in the form (###)###-####.

Validating a String (code)

- We can examine a string to ensure that it contains a correctly formatted phone number. (e.g., (703)321-6753). You can use a loop to test each character to see it it is correct for its position, or a number

```
string=input("Enter phone in the format (XXX)XXX-XXXX:")
valid = len(string) == 13
position = 0
while valid and position < len(string) :

    if position == 0:
        valid = string[position] == "("
    elif position == 4 :
        valid = string[position] == ")"
    elif position == 8 :
        valid = string[position] == "-"
    else:
        valid = string[position].isdigit()
    position = position + 1

if valid :
    print("The string contains a valid phone number.")
else:
    print("The string does not contain a valid phone number.")
```

Validating a String (code)

- Alternatively, we can combine the previous four logical conditions into a single expression to produce more compact loop

```
string=input("Enter phone in the format (XXX)XXX-XXXX:")
valid = len(string) == 13
position = 0
while valid and position < len(string) :
   valid = ((position == 0 and string[position] == "(") or
            (position == 4 and string[position] == ")") or
            (position == 8 and string[position] == "-") or
            (position != 0 and position != 4 and position != 8  and
             string[position].isdigit()))
   position = position + 1

if valid :
    print("The string contains a valid phone number.")
else:
    print("The string does not contain a valid phone number.")
```

Building a New String

- One of the minor annoyances of online shopping is that many web sites require you to enter a credit card without spaces or dashes, which makes double-checking the number rather tedious.
- How hard can it be to remove dashes or spaces from a string?

Code Academy

Building a New String (code)

- The contents of a string cannot be changed.
- But nothing prevents us from building a new string.
- Here is a loop that builds a new string containing a credit card number with spaces and dashes removed:
  - We read the credit card number
  - We initialize a new string to the empty string
  - We test each character in the user input
    - If the character is not a space or dash we append it to the new string

```
userInput = input("Enter a credit card number: ")
creditCardNumber = ""
for char in userInput :
   if char != " " and char != "-" :
      creditCardNumber = creditCardNumber + char
```

Code Academy

Example: Grading Multiple Choice Exam **multiplechoice.py**

Multiplechoice.py program demonstrates processing strings. The program reads a string that contains a test taker's answers to a multiple choice exam and grades the test.

```
#  This program grades a multiple choice exam in which each question has four
#  possible choices: a, b, c, or d.

# Define a string containing the correct answers.
CORRECT_ANSWERS = "adbdcacbdac"

# Obtain the user's answers, and make sure enough answers are provided.
done = False
while not done :
   userAnswers = input("Enter your exam answers: ")
   if len(userAnswers) == len(CORRECT_ANSWERS) :
      done = True
   else :
      print("Error: an incorrect number of answers given.")

# Check the exam.
numQuestions = len(CORRECT_ANSWERS)
numCorrect = 0
results = ""
```

Code Academy

Example: Grading Multiple Choice Exam **multiplechoice.py** (2)

```
for i in range(numQuestions) :
   if userAnswers[i] == CORRECT_ANSWERS[i] :
      numCorrect = numCorrect + 1
      results = results + userAnswers[i]
   else :
      results = results + "X"

# Grade the exam.
score = round(numCorrect / numQuestions * 100)

if score == 100 :
   print("Very Good!")
else :
   print("You missed %d questions: %s" % (numQuestions - numCorrect, results))

print("Your score is: %d percent" % score)
```

```
Enter your exam answers: adaacacddac
You missed 3 questions: adXXcacXdac
Your score is: 73 percent
```

Code Academy

Random Numbers/Simulations

- Games often use random numbers to make things interesting
  - Rolling Dice
  - Spinning a wheel
  - Pick a card
- A simulation program uses the computer to simulate an activity in the real world (or the imaginary world).
- A simulation usually involves looping through a sequence of events

Code Academy

Generating Random Numbers

- The Python library has a *random number generator* that produces numbers that appear to be random
  - The numbers are not completely random. The numbers are drawn from a sequence of numbers that does not repeat for a long time
  - random() returns a number that is >= 0 and < 1

Simulating Die Tosses

- Goal:
  - To generate a random integer in a given range we use the randint() function
  - Randint has two parameters, the range (inclusive) of numbers generated

ch04/dice.py

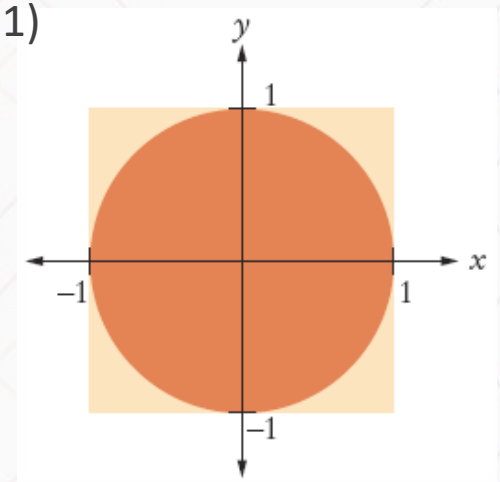
```
1  ##
2  #  This program simulates tosses of a pair of dice.
3  #
4
5  from random import randint
6
7  for i in range(10) :
8     # Generate two random numbers between 1 and 6, inclusive.
9     d1 = randint(1, 6)
10    d2 = randint(1, 6)
11
12    # Print the two values.
13    print(d1, d2)
```

Program Run

```
1 5
6 4
1 1
4 5
6 4
3 2
4 2
3 5
5 2
4 5
```

The Monte Carlo Method

- Used to find approximate solutions to problems that cannot be precisely solved
- Example: Approximate PI using the relative areas of a circle inside a square. Simulate shooting a dart into a square surrounding circle of radius 1. To do this,
  - Generate random x- and y-coordinates between -1 and 1
  - If the generated point lies in the circle it is a hit (when $x^2+y^2 \leq 1$)
  - Tries are total number of tries
  - $\pi/4$ = Hits / Tries
  - π = 4 x Hits / Tries



→ To generate a random floating point between a and b:

```
r = random()                    # 0 ≤ r < 1
x  = a + ( b – a ) * r          # -1≤ x < ≤ 1
```

Monte Carlo Example

```
1   ##
2   #  This program computes an estimate of pi by simulating dart throws onto a square
3   #
4
5   from random import random
6
7   TRIES = 10000
8
9   hits = 0
10  for i in range(TRIES) :
11
12     # Generate two random numbers between -1 and 1
13     r = random()
14     x = -1 + 2 * r
15     r = random()
16     y = -1 + 2 * r
17
18     # Check whether the point lies in the unit circle
19     if x * x + y * y <= 1 :
20        hits = hits + 1
21
22  # The ratio hits / tries is approximately the same as the ratio
23  # circle area / square area = pi / 4.
24
25  piEstimate = 4.0 * hits / TRIES
26  print("Estimate for pi:", piEstimate)
```

Program Run

```
Estimate for pi: 3.1464
```

Guess Game: Guess number 1 - 100

```
#Program simulates a guess number game
#The player is asked to guess a number bewteen 1 and 100

from random import randint
#generate a random integer 1-100
r = randint(1,100)

done = False
while not done:
    #get user guess
    userGuess = int(input("Guess a number between 1 and 100: "))
    if userGuess <1 or userGuess >100 :
        print("Your guess is outside range 1-100!")
    else:
        if userGuess == r:
            print("Congratulation! You win.")
            done = True
        elif userGuess > r:
            print("Go lower")
        else:
            print("Go higher")
```

```
Guess a number between 1 and 100: 200
Your guess is outside range 1-100!

Guess a number between 1 and 100: 50
Go lower

Guess a number between 1 and 100: 25
Go lower

Guess a number between 1 and 100: 10
Go higher

Guess a number between 1 and 100: 15
Go higher

Guess a number between 1 and 100: 18
Go lower

Guess a number between 1 and 100: 17
Congratulation! You win.
```

Summary

- In a simulation, you use the computer to simulate an activity.
  - You can introduce randomness by calling the random number generator.