Data Processing and Analysis in Python
Lecture 4
Loop Statements

UNIVERSITY OF MARYLAND

ROBERT H. SMITH SCHOOL OF BUSINESS

DR. ADAM LEE

Definitive Iteration: The for Loop

- Repetition statements (or loops) repeat an action
- Each repetition of action is known as pass or **iteration**
- Two types of loops:
  - **Definite iteration**: repeat action a predefined number of times
  - **Indefinite iteration**: perform action until program determines it needs to stop
- **for** loop is the control statement that most easily supports definite iteration
- **for** <variable> **in** range(<a numeric expression>)**:**
  <sequence of statements>

UNIVERSITY OF MARYLAND

Examples

```
>>> for eachPass in range(4):
    print("It's alive!", end = ' ')
It's alive! It's alive! It's alive! It's alive!

>>> number = 2
>>> exponent = 3
>>> product = 1
>>> for eachPass in range(exponent):
    product = product * number
    print(product, end = ' ')
2 4 8
>>> product
8

# exponent = 0 ? What would happen? product = ?
```

UNIVERSITY OF MARYLAND

Create a Sequence of Numbers

- range(<a numeric expression>)

```
# range(stop-value)
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# range(start-value, stop-value)
>>> range(1, 10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]

# range(start-value, stop-value, step-value)
>>> range(1, 10, 2)
[1, 3, 5, 7, 9]
```

Count-Controlled Loops

- Loops that count through a range of numbers:

```
>>> product = 1
>>> for count in range (4):
        product = product * (count + 1)
>>> product
24
```

- To specify a explicit lower bound:

```
>>> product = 1
>>> for count in range (1, 5):
        product = product * count
>>> product
24
```

Example

```
>>> lower = int(input("Enter the lower bound: "))
Enter the lower bound: 1
>>> upper = int(input("Enter the upper bound: "))
Enter the upper bound: 10
>>> theSum = 0
>>> for number in range(lower, upper + 1):
    theSum = theSum + number
>>> theSum
55
```

Augmented Assignment

- The assignment symbol can be combined with the arithmetic and concatenation operators to provide augmented assignment operations.
- <variable> <operator>= <expression> # equivalent to <variable> = <variable> <operator> <expression>

```
a += 3 # Equivalent to a = a + 3
a -= 3 # Equivalent to a = a - 3
a *= 3 # Equivalent to a = a * 3
a /= 3 # Equivalent to a = a / 3
a %= 3 # Equivalent to a = a % 3

s += '.' # Equivalent to s = s + '.'
```

Traversing the Contents of a Data Sequence

- **for** <variable> **in** <sequence>**:**
  <do something with variable>
- Values in a sequence can be visited with a for loop

```
>>> product = 1
>>> for count in [1, 2, 3, 4]:
        product = product * count
>>> product
24
```

- Strings are also sequences of characters

```
>>> for character in "Hi there!":
        print(character, end = ' ')
H i   t h e r e !
```

UNIVERSITY OF MARYLAND

Exercises

- Count Down?

- Times/Multiplication Table?

Formatting Text for Output

- Many data-processing applications require output that has tabular format
- **Field width**: Total number of data characters and additional spaces for a datum in a formatted string
- The print() function automatically begins printing an output datum in the first available column
- print("<format string>" **%** <datum>) # format operator %
  - %<field width>**d**: to format integer
  - %<field width>[.<precision>]**f**: to format floating number
  - %<field width>**s**: to format string
- print("<format string>" **% (**<datum-1>, ..., <datum-*n*>**)**)

Examples

```
>>> for exponent in range(7, 11):
    print("%-3d%12d" % (exponent, 10 ** exponent))
7          10000000
8         100000000
9        1000000000
10 10000000000

>>> salary = 100.00
>>> print("Your salary is $" + str(salary))
Your salary is $100.0
>>> print("Your salary is $%0.2f" % salary)
Your salary is $100.00
```

Conditional Iteration: The while Loop

- **while** loop can be used to describe conditional iteration
  - Also called entry-control loop, i.e. condition is tested at top of loop
  - Statements within loop can execute zero or more times
  - Example: A program's input loop that accepts values until user enters a sentinel that terminates the input
- **while** <continuation condition>**:**
  <sequence of statements>
- Improper use may lead to infinite loop
  - Type Ctrl+c to halt loop that hang

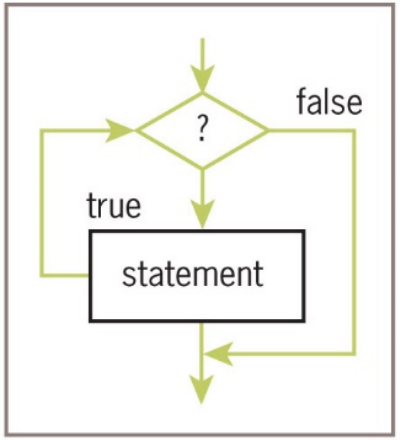

Figure 3-6 The semantics of a `while` loop

Example: a sentinel that terminates the input

```
theSum = 0.0
data = input("Enter a number or just enter to
quit: ")
while data != "":
    number = float(data)
    theSum += number
    data = input("Enter a number or just enter to
quit: ")
print("The sum is", theSum)

Enter a number or just enter to quit: 3
Enter a number or just enter to quit: 4
Enter a number or just enter to quit: 5
Enter a number or just enter to quit:
The sum is 12.0
```

- **data** is the **loop control variable**

UNIVERSITY OF MARYLAND

Count-Controlled Loops

```
# Summation with a for loop
theSum = 0
for count in range(1, 100001):
    theSum += count
print(theSum)

# Summation with a while loop
theSum = 0                  # same as for loop
count = 1                   # loop control variable
while count < 100001:       # when to continue
    theSum += count         # same as for loop
    count += 1              # prevent infinite loop
print(theSum)               # same as for loop
```

UNIVERSITY OF MARYLAND

While True Loop and Break Statement

- The while loop can be complicated to write correctly
  - It is possible to simplify its structure and improve its readability
  - Within this body, the input datum is received
  - It is then tested for the loop's **termination condition** in a one-way selection statement
  - The **break** statement will cause an exit from the loop

```
theSum = 0.0
While True:
    data = input("Enter a number or just enter to quit:
")
    if data == "":
        break
    theSum += float(data)
print("The sum is", theSum)
```

Examples

```
while True:
    number = int(input("Enter the numeric grade: "))
    if number >= 0 and number <= 100:
        break
    else:
        print("Error: grade must be between 100 and 0")
print(number) # Just echo the valid input

done = False
while not done:
    number = int(input("Enter the numeric grade: "))
    if number >= 0 and number <= 100:
        done = True
    else:
        print("Error: grade must be between 100 and 0")
print(number) # Just echo the valid input
```

UNIVERSITY OF MARYLAND

Random Numbers

- **random** module supports several ways
  - Programming languages include resources for generating random numbers
  - **randint()** returns random number from among numbers between two arguments, included

```
>>> import random
>>> for roll in range(10):
    print(random.randint(1, 6), end = ' ')
2 4 6 4 3 2 3 5 1 2
```

A Simple Guessing Game

```
import random
smaller = int(input("Enter the smaller number: "))
larger = int(input("Enter the larger number: "))
myNumber = random.randint(smaller, larger)
count = 0
while True:
    count += 1
    userNumber = int(input("Enter your guess: "))
    if userNumber < myNumber:
        print("Too small!")
    elif userNumber > myNumber:
        print("Too large!")
    else:
        print("Congratulations! You've got it in",
count, "tries!")
        break
```

A Simple Guessing Game

```
Enter the smaller number: 1
Enter the larger number: 100
Enter your guess: 50
Too small!
Enter your guess: 75
Too large!
Enter your guess: 63
Too small!
Enter your guess: 69
Too large!
Enter your guess: 66
Too large!
Enter your guess: 65
You've got it in 6 tries!
```

UNIVERSITY OF MARYLAND