Python Programming

Lecture 3 Iteration

3.1 Iteration: while

The while statement

```python
n = 5
while n > 0:
    print(n)
    n = n - 1

print('Blastoff!')
```

```
5
4
3
2
1
Blastoff!
```

1. Evaluate the condition, yielding True or False.
2. If the condition is false, exit the while statement and continue execution at the next statement.
3. If the condition is true, execute the body and then go back to step 1.
    - This type of flow is called a loop because the third step loops back around to the top.
    - We call each time we execute the body of the loop an iteration.

- The body of the loop should change the value of one or more variables so that eventually the condition becomes false and the loop terminates.
- If the loop repeats forever, it results in an infinite loop.

```python
n = 10
while True:
    print(n)
    n = n - 1
print('Done!')
```

- What will happen if you run this?
- Ctrl+C to terminate it.

- Finishing iterations with break

```python
while True:
    line = input('Please enter:')
    if line == 'done':
        break
    print(line)
print('Done!')
```

- Finishing iterations with continue

```python
while True:
    line = input('Please enter:')
    if line == '#':
        continue
    if line == 'done':
        break
    print(line)
print('Done!')
```

Example 1: Even and Odd

```python
numbers = [12, 37, 5, 42, 8, 3]
even = []
odd = []
while len(numbers) > 0:
    x = numbers.pop()
    if x % 2 == 0:
        even.append(x)
    else:
        odd.append(x)
print(even)
print(odd)
```

```
[8, 42, 12]
[3, 5, 37]
```

Example 2: BMI Calculator v2.0

```python
print("BMI指数计算器\n")
while True:
    try:
        inp_1 = input('请输入您的体重(kg):\n')
        weight = float(inp_1)
        break
    except:
        print('请输入数字')
while True:
    try:
        inp_2 = input('请输入您的身高(cm):\n')
        height = float(inp_2)
        break
    except:
        print('请输入数字')
```

Example 3: Greatest common divisor

```python
a=int(input('Enter the 1st number:'))
b=int(input('Enter the 2nd number:'))
if a >= b:
    x = a
    y = b
else:
    x = b
    y = a
while y!=0:
    r = y
    y = x%y
    x = r
print(x)
```

- Least common multiple?

Example 4: Hot Potato Game

- To begin, let's consider the children's grame Hot potato. In this game children line up in circle and pass an item from neighbor to neighbor as fast as they can. At a certain point in the game, the action is stopped and the child who has the item (the potato) is removed from the circle. Play continues until only one child is left.
- This game is a modern-day equivalent of the famous Josephus problem.

```python
name_list = ['a', 'b', 'c', 'd', 'e', 'f']
i = 1
while len(name_list) > 1:
    if i % 7 == 0:
        y = name_list.pop()
    else:
        y = name_list.pop()
        name_list.insert(0, y)
        # print(name_list)
    i = i + 1
print(name_list)
```

Example 5: Finding prime numbers

```python
i = 2
while i < 100:
    j = 2
    while j**2 <= i:
        if i%j == 0 :
            break
        j = j + 1
    if j**2 > i:
        print(i)
    i = i + 1
```

3.2 Iteration: for

Definite loops using for

- for statement works on the lists.

```python
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends:
    print('Happy New Year:', friend)
print('Done!')
```

```
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Done!
```

- We refer to the while loop as an indefinite loop because it continues executing until a specific condition becomes False, whereas the for loop iterates over a known set of items, running as many times as there are items in the set.
- In particular, friend is the iteration variable for the for loop. The variable friend changes for each iteration of the loop and controls when the for loop completes. The iteration variable steps successively through the three strings stored in the friends variable.

- The indentation errors are common.
- Python uses indentation to determine when one line of code is connected to the line above it. Some languages require the "end" statement.
- Always indent the line after the for statement in a loop.

```python
friends = ['Joseph', 'Glenn', 'Sally']
for x in friends:
print('Happy New Year:', x)
print('Done!')
```

```
IndentationError: expected an indented block
```

```python
message = "Hello Python world!"
    print(message)
```

```
IndentationError: unexpected indent
```

- Forgetting to indent additional lines (logical error)

```python
friends = ['Joseph', 'Glenn', 'Sally']
for x in friends:
    print('Happy New Year:', x)
print('Looking forward to seeing you,', x)
```

```
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Looking forward to seeing you, Sally
```

```python
friends = ['Joseph', 'Glenn', 'Sally']
for x in friends:
    print('Happy New Year:', x)
    print('Looking forward to seeing you,', x)
```

```
Happy New Year: Joseph
Looking forward to seeing you, Joseph
Happy New Year: Glenn
Looking forward to seeing you, Glenn
Happy New Year: Sally
Looking forward to seeing you, Sally
```

range function

- The range function generates a sequence of integers.
- It often used to control the number of iterations in a for loop.

```python
for value in range(1,4):
    print(value)
```

```
1
2
3
```

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

- Example: generate a list of square numbers

```python
squares = []
for value in range(1,11):
    square = value**2
    squares.append(square)
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Replicate the functionality of list functions using for loops.

sum(): how to calculate 1 + 2 + 3 +...+ 100?

```python
total = 0
for value in range(1, 101):
    total = value + total  # total+=value
print(total)
```

len()

```python
a_list = [3, 41, 12, 9, 74, 15]
count = 0
for x in a_list:
    count = count + 1
print('Count: ', count)
```

max(), min()

```python
a_list = [3, 41, 12, 9, 74, 15]
largest = a_list[0]
for x in a_list:
    if x > largest :
        largest = x
print('Largest:', largest)
```

sorted(): sorting is a much more complicated and interesting topic.

Nested Loops

```python
x = []
for i in range(1,4):
    x.append([])
    for j in range(1,4):
        x[i-1].append(1)
print(x)
```

```
[[1, 1, 1], [1, 1, 1], [1, 1, 1]]
```

```python
x = []
for i in range(1,4):
    x.append([])
    for j in range(1,i+1):
        x[i-1].append(1)
print(x)
```

```
[[1], [1, 1], [1, 1, 1]]
```

```python
for i in range(1,4):
    for j in range(1,i+1):
        print(i, end=" ")
    print()
```

```
1
2 2
3 3 3
```

- The default value of end is \n meaning that after the print statement it will print a new line. So simply stated end is what you want to be printed after the print statement has been executed

```python
for i in range(1,4):
    for j in range(i):
        print(i,end="+")
    print()
```

```
1+
2+2+
3+3+3+
```

Summary

- Iteration
    - Reading: Python for Everybody Chapter 5