大学计算机入门课

Class #5

[教学目的]

- Know how to use the for loops
- Know the new data type --- List

[课程大纲]

- Part 1- for loops
- Part 2 – lists
- Part 3 – function about lists
    - Function return a list
    - For loop for list

part 1 – for loops

当我们想对一个集合里的每一个元素都实施相同的命令时，我们可以用 for loop！

- For loops for strings.

```
>>> for s in "hello":
...     print(s)
...
h
e
l
l
o
```

- Firstly, let's try a function using for loops.
  If we want to print out every digit in a string s line by line

  这个时候，我们需要检查每一位字母，并且用 if 来判断它是否满足是 digit 的条件。如果是的话，我们就 print 它。

```
def print_digit(s):
    """ (str) --> None

    print out every digit in a string s line by line.

    >>> print_digit("1n2n3n4")
    1
    2
    3
    4
    """
    for i in s:
        if i.isdigit():
            print(i)
```

- 现在我们来考虑一个 function， 他能够 count_uppercase in string s.

```
def count_uppercase(s):
    """(str) -> int

    Retrun the number of uppercase letters in s.

    >>> count_uppercase("This is UofT")
    3
    >>> count_uppercase("PYTHON is great")
    6
    """
```

首先我们需要一个 variable count 来 纪录一共出现了多少次的 upper case,
然后用 for loop 去检查每一个 char， 如果是 upper case， 那么我们的 count + 1.

```
def count_uppercase(s):
    """(str) -> int

    Retrun the number of uppercase letters in s.

    >>> count_uppercase("This is UofT")
    3
    >>> count_uppercase("PYTHON is great")
    6
    """
    # count must go out side the loop!
    count = 0
    for c in s:
        #check if uppercase
        if c.isupper():
            # increment count
            count += 1
    return count
```

- 现在我们来考虑一个 bool function by given docstring.

```
def is_ip_address(address):
    """ (str) -> bool

    Return True if address is made up of digits and periods, and False otherwise.

    >>> is_ip_address('252.17.34.9')
    True
    >>> is_ip_address('40 St. George St')
    False
    """
```

first try:

```
    # first try:
    for char in address:
        if char == ".":
            return True
        elif char.isdigit():
            return True
        else:
            return False
    # why is this a problem?!!
```

This code will return true even if the address is given by ".40 st. George".

```
>>> is_ip_address(".40 st. George")
True
>>>
```

how can we fix it?
因为我们要检查每一个 address 里的元素，如果他们全部符合要求我们才 return True， 但是一旦我们发现有一个元素不符合要求，我们立刻 return False。
Another try：

```
    flag = True
    for char in address:
        if char == ".":
            flag = True
        elif char.isdigit():
            flag = True
        else:
            flag = False
    return flag
```

still has a problem

```
>>> is_ip_address('40 St. George St9')
True
>>>
```

fix it now:

```
    for char in address:
        if char != "." and not char.isdigit():
            return False
    return True
```

test it:

```
>>> is_ip_address('40 St. George St9')
False
>>> is_ip_address('40.34.34')
True
```

- For loops for integer
  之前我们一直在讨论关于 string 的 for loops，相同的对于 integer 我们也可以运用 integer。

- Range()

```
range(stop) -> range object
range(start, stop[, step]) -> range object

Return an object that produces a sequence of integers from start (inclusive)
to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, ..., j-1.
start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
These are exactly the valid indices for a list of 4 elements.
When step is given, it specifies the increment (or decrement).
```

range(10) 储存了 0, …, 9 共 10 个 integer 在其中。

```
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
```

range 也可以记载从制定开始到结束的数。

```
>>> for i in range(1, 10):
...     print(i)
...
1
2
3
4
5
6
7
8
9
```

range 也可以不是 1 by 1 的 increasing。

```
>>> for i in range(1, 10, 2):
...     print(i)
...
1
3
5
7
9
>>>
```

- The use of range()

如果我们想要 print 所有从 1 到 given end 中能够整除 9 的数时， 我们需要用到 for loop.

```
def print_multiple_of_nine(end):
    """ (str) -> None

    Print out every multiple of nine from 0 to end.

    Precodition: end >= 0

    >>> print_multiple_of_nine(20)
    0
    9
    18
    """
    for i in range(end):
        if i%(9) == 0:
            print(i)
```

- 更多的应用：
  想一想，结合我们第一节课所学的
  二进制 与 十进制的转化
  consider the function binary_to_base10, which converting the integer from binary to base 10
  101 (binary) = 1 * 2 **0 + 0 * 2 ** 1 + 1 * 2 ** 2 = 5

  the formula of converting binary to base 10:

$$abcd = a \times 2^3 + b \times 2^2 + c \times 2^1 + d \times 2^0$$

  try to using for loop to get the function!!

```
def binary_to_base10(d):
    """(int) -> int

    Return the number of binary d based 10.

    >>> binary_to_base10(101)
    5
    >>> binary_to_base10(100101111)
    303
    """
    # convert d to a sting first!
    d_s = str(d)
    # initialize our result first, since we are keeping adding thing
    # to our result. outside the loop!
    result = 0

    for i in range(len(d_s)):
        result += int(d_s[i]) * 2 ** (len(d_s) - i - 1)
    return result
```

- combine for loops for string and integer

```
def index_and_char(s):
    """ (str) -> None

    Print the index, the s[index] line by line.

    >>> index_and_char("hello")
    0,h
    1,e
    2,l
    3,l
    4,o
    """
    for i in range(len(s)):
        print(str(i) + "," + s[i])
```

- break statement

有时我们并不想要 for loop 一直检查到所有的情况。这时，我们可以运用 break，高速系统我们将要停止运行 for loop。

```
>>> for char in "1234567":
...     if char == "5":
...         break
...     print(char)
...
1
2
3
4
>>>
```

比如说，对于一个 string s， 如果我们想要 print 每一个字母直到第一个数字出现。

```
def char_before_digit(s):
    """ (str) -> None

    Print out every char line by line before the first digit.

    >>> char_before_digit("jhjf12khdfks")
    j
    h
    j
    f
    """
    for char in s:
        if char.isdigit():
            break
        print(char)
```

part 2 – List

python knows a number of compound data types, used to group together other values. The most versatile is the list, which can be written as a list of comma-separated values (items) between square brackets.

```
>>> a = [1,2,3,4,5]
>>> a
    [1, 2, 3, 4, 5]
```

Note: lists might contain items of different types, but usually the items all have the same type.

```
>>> b = [1, "hello", 3]
>>> b
    [1, 'hello', 3]
```

我们之前讨论的 string.split() 其实 return 的就是一个 list：

```
>>> "adsf ads af".split()
    ['adsf', 'ads', 'af']
>>>
```

about list:

- list addition

```
>>> [1,2,3,4,5] + [2,3,4,5]
    [1, 2, 3, 4, 5, 2, 3, 4, 5]
>>>
```

- have index

```
>>> a = [1,2,3,4,5]
>>> a[0]
    1
>>> a[1]
    2
>>> a[2]
    3
>>> a[3]
    4
>>> a[4]
    5
```

- mutable

```
>>> b = [1, "hello", 3]
>>> b[1] = 2
>>> b
    [1, 2, 3]
```

(note: unlikely, string is not mutable.)

- add new items at the end of the list, by using the append()

```
>>> b = [1,2,3]
>>> b.append(4)
>>> b
    [1, 2, 3, 4]
```

- Assignment to slices is also possible, and this can even change the size of the list or clear it entirely.

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters[2:5] = []
>>> letters
    ['a', 'b', 'f', 'g']
```

- The bulid-in function len().

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> len(letters)
7
>>>
```

- And more!

```
clear(...)
    L.clear() -> None -- remove all items from L

copy(...)
    L.copy() -> list -- a shallow copy of L

count(...)
    L.count(value) -> integer -- return number of occurrences of value

extend(...)
    L.extend(iterable) -> None -- extend list by appending elements from the iterable

index(...)
    L.index(value, [start, [stop]]) -> integer -- return first index of value.
    Raises ValueError if the value is not present.

insert(...)
    L.insert(index, object) -- insert object before index

pop(...)
    L.pop([index]) -> item -- remove and return item at index (default last).
    Raises IndexError if list is empty or index is out of range.

remove(...)
    L.remove(value) -> None -- remove first occurrence of value.
    Raises ValueError if the value is not present.

reverse(...)
    L.reverse() -- reverse *IN PLACE*

sort(...)
    L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE*
```

part 3 – function about List

- Consider a function that return a list of upper case char in string s.

```
def get_upper_char(s):
    """(str) -> list

    return a list of upper case char in string s.

    >>> get_upper_char("AbCd")
    ['A', 'C']
    """
    # firstly, iniliaze a empty list
    result = []
    for char in s:
        if char.isalpha() and char.isupper():
            result.append(char)
    return result
```

- For loops for list:
  同样的，所有的 for loops 也可以查看 list 里的每一个 item。

```
>>> for item in ["a", "b", "c", "d"]:
...     print(item)
...
a
b
c
d
```

now, consider a function
that return the scale of the midterm grades.
假设我们实际的成绩是考试成绩乘 multiplier 加上 bonus 得来的。
Design a function that Return the list of marks with each grade times the multiplier and add the bonus, but the maximal mark is 100.

```
def scale_midterm_grades(grades, multiplier, bonus):
    """(list[number], number, number) -> list

    Return the list of marks with each grade times the multiplier and add
    the bonus, but the maximal mark is 100.

    >>> scale_midterm_grades([55, 60, 65, 100],1.1,10)
    [70.5, 76.0, 81.5, 100]
    """
    new_grades = []
    for i in range(len(grades)):
        new_grades.append(min(grades[i] * multiplier + bonus, 100))
        # other ways to do this, but min lets us use one line :)
    return new_grades
```