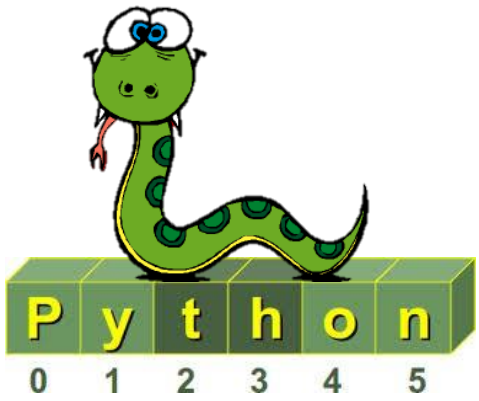
IDCE 302: Chapter 7

Strings & For Loop

Jie Tian, PhD

Clark University



Outline

- Data types
- Working with string (length, string slicing)
- for loop
- String comparison
- More string functions…

Revisit Values and Types

Integer (int)
Float (float)
String (str)

Check the type of a integral value

```
>>> type(4)
<type 'int'>
```

Check the type of a value with decimals

```
>>> type(3.14159)
<type 'float'>
```

Check the type of a string:

```
>>> type("Hello, World!")
<type 'str'>
```

A Compound Data Type

- An `int` and a `float` are simple data types.
- A string (`str`) is a compound data type, because it consists of smaller pieces .
- **A string is a sequence of characters**!

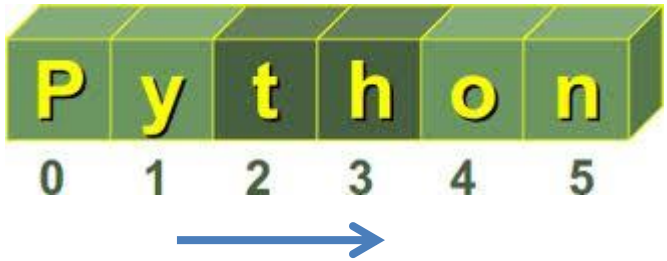
```
>>> fruit = "apple"
>>> letter_1 = fruit[0]
>>> letter_5 = fruit[4]
>>> print letter_1
a
>>> print letter_5
e
```

Length -- the len() function

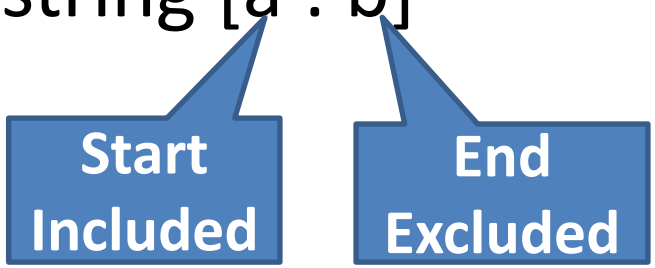
```
>>> fruit = "apple"
>>> len(fruit)  #get the length
5

>>> length  = len(fruit)
>>> last =  fruit[length-1]
>>> last
'e'
```

String Slicing

- slice out a sub-string from a string
- Index value



- string [a : b]




```
s = 'Hello'
part = s[1 : 4]
part
'ell'
```

String Slicing Examples

```
          01234567890123456789O
>>> s = "Peter, Paul, and Mary"
>>> print s[0:5]
Peter
>>> print s[7:11]
Paul
>>> print s[17:21]
Mary
              012345
>>> fruit = "banana"
>>> fruit[:3]  # start is optional
"ban"
>>> fruit[3:] # end is optional
"ana"
```

Traversal using the `while` loop

Task: To print each character in the string!

```
# use "while" to traverse a string
fruit = "apple"
index = 0
while index < len(fruit):
    letter = fruit[index]
    print letter
    index = index + 1
```

for Loop

- *for* loops are used when you have a piece of code which you want to repeat ***n*** times.

```
for iterating_var in sequence:
      statements
```

```
fruit = "apple"
# use 'for … in …' to traverse a string
for char in fruit:
    print char
```

```
for iterating_var in sequence:
      statements
```

- For loop is for you to **traverse** through a **sequence**.
- Examples of sequence:

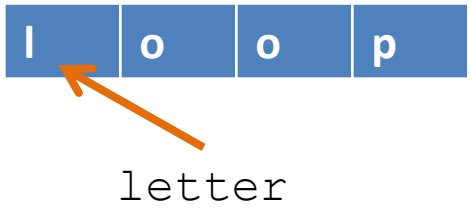
  string→ a sequence of characters, ‘hello’

  list→ a sequence of values (to be discussed next week)

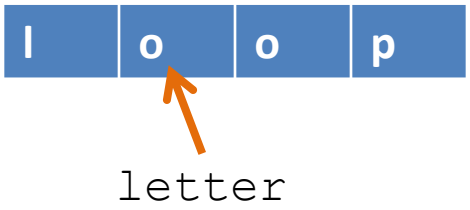
  [0,1,2,3]
- Iterating_var is a variable that **iteratively refers** (or points) to the elements of the sequence. One of them at a time.
- You can choose any legal variable name for iterating_var

```
word = 'loop'
for letter in word:
        print letter
```

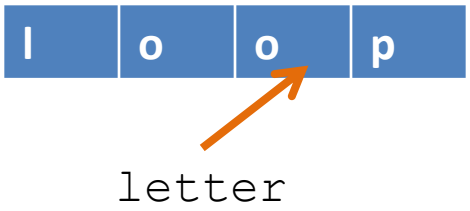
Iteration 1



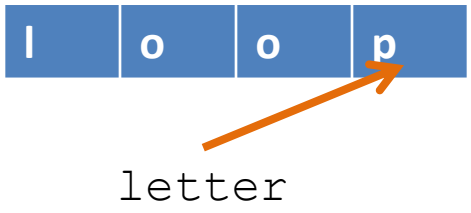
Iteration 2



Iteration 3



Iteration 4



```
prefixes = "JKLM"
suffix = "ack"

for letter in prefixes:
    print letter + suffix
```

The output of this program is:

```
Jack
Kack
Lack
Mack
```

- You want to repeat the code a certain number of times.

```
for x in range(0,3):
      print 'We\'re on time %d' % (x)
```

```
>>> seq = range (0, 3)
>>> seq
[0, 1, 2]
```

Exercise: use for loop to get the sum of all the even numbers between 1 and 99

String Comparison

- Why do we need to compare strings?

1) To check if two strings are equal.

2) To put strings (such as names, products) in order.

```
>>> str1='worcester'

>>> str2='boston'

>>> print str1>str2

True
```

ASCII

- ASCII stands for American Standard Code for Information Interchange.
- Computers can only understand numbers.
- So an ASCII code is the numerical representation of a character ('a' or '@‘)
- Comparing strings are actually comparing the corresponding ASCII codes of their characters.

33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-

48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<

64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K

97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l

Strings are immutable!

- A string's contents can **NOT** be modified
- Not working by modifying:

```
greeting = "Hello, world!"
greeting[0] = "J"       # ERROR! Can't modify it
```

- It works by creating a new string

```
greeting = "Hello, world!"
newGreeting = "J" + greeting[1:]    #create a new one
```

find Function (We write it)

```
"""
Input
    str: a string to search from
    ch:  a letter to be searched
Return value
    index of the first found, 0-based, -1 if not found
"""
def find(str, ch):
    index = 0
    while index < len(str):
        if str[index] == ch:
            return index
        index = index + 1
    return -1

print find("Hello world", "w")
print find("Hello world", "s")
```

Output:

```
6
-1
```

Looping & Counting

```
"""
Input
    str: a string to search the letter from
    ch:  a letter to be searched in str
output
    Returns the index of the first found ch in str. The index is 0-based
    If not found, return -1
"""
def countLetter(str, ch):
    count = 0
    for char in str:
        if char == ch:
            count = count + 1

    return count

# test cases
print countLetter("Programming for GIS", "G")
print countLetter("Programming for GIS", "m")
```

Output

```
1
2
```

string Functions

Used after a string

```
# find the index of a character of interest
find()
'hello'.find('e')

# check if the string only includes numbers and letters
isalnum()

# check if the string only includes numbers
isdigit()

#check if the string only includes letters
isalpha()

islower()  isspace()   join()
```

The string Module

- Includes many useful functions for string operations

```
Import string
string.find()

>>> print string.ascii_lowercase
abcdefghijklmnopqrstuvwxyz

>>> print string.ascii_uppercase
ABCDEFGHIJKLMNOPQRSTUVWXYZ

>>> print string.digits
0123456789
```

```
>>> import string  #import the module before using it
>>> str1='boston'
>>> string.find(str1, 'n')
5
>>> string.find(str1, 'x')
-1

>>> fruit = "banana"
>>> index = string.find(fruit, "a")
>>> print index
1

>>> string.find("banana", "na", 3) # in range [3:]
4

>>> string.find("bob", "b", 1, 2) # in range [1, 2)
-1

>>> string.find("bob", "b", 1, 3) # in range [1, 3)
2
```

Example: 3 ways of writing your isLower() function

```
import string

def isLower(ch):
    return string.find(string.lowercase, ch) != -1

def isLower(ch):
    return ch in string.lowercase

def isLower(ch):
    return 'a' <= ch <= 'z' #it is valid in Python
```

Summary

- Data types (integer, float, string)
- String is a sequence of characters.
- String is immutable!
- string.function()
- For loop when you want to run a piece of code for a certain number of times
- String module if more string functions needed.

Exercise

Write a Python program to count the letters from "a" to "m" and from "A" and "M" in a given string. For example, string "Worcester" contains 3 letters in the given range, and string "Boston" contains only 1 letter.

Hint Example:

```
def isLower(ch):
    return 'a' <= ch <= 'z'
```