PROGRAMMING

Lecture 11

Hanbat National University
Dept. of Computer Engineering
Changbeom Choi

REVIEW

Data structures (objects composed of another objects):

Tuples

(1, 3, 5, 7, 9)

(“red”, “green”, “blue”)

(777, “a lucky number”)

Strings

Lists

Dictionary

OUTLINE

Preview: tuples and strings

Lists

Summary: strings, tuples, lists

Reading assignment

Chapters 8 - 11 of the textbook

PREVIEW: TUPLES AND STRINGS

Tuples

A **tuple** is an **ordered sequence** of data **elements**. It is an **immutable** object. In other words, a tuple can never be changed unless it is re-created.

tup0 = () (or tup0 = tuple())

tup1 = (1, 2, 3, 4, 5)

tup2 = (1.57, 3.14, 9.02)

tup3 = (“one”, “two”, “three”)

tup4 = (1, 3.14, “two”, 2, (5, 6))

A singleton tuple

Is (2) a tuple ? Well, ……

Is (2) the same as 2 ?

Yes !

Then, how to express a singleton tuple, e.g., a tuple containing only 2 as the element?

(2,)

Selection

```
>>>tp1 = (1, 2, 3, 4, 5)
>>>tp1
(1, 2, 3, 4, 5)
>>>tp1[0]
1
>>>tp1[2]
3
>>>tp1[-1]
5
>>>tp1[-2]
4
```

Slicing

```
>>>tp1[1:3]
(2, 3)
>>>tp1[:3]
(1, 2, 3)
>>>tp1[1:]
(2, 3, 4, 5)
>>>tp1[:]
(1, 2, 3, 4, 5)
>>>tp1
(1, 2, 3, 4, 5)
```

Concatenation

```
>>>tp1 = (1, 2)
>>>tp2 = (3, 4)
>>>tp1 + tp2
(1, 2, 3, 4)
```

Strings

A string is an ordered sequence of characters. It is an immutable object.

st1 = “abcedghijklmnopqrstuvwxyz”

st2 = “0123456789”

St3 = “cce20003 is fantastic !”

>>>st1 + st2 concatenation

‘abcdefghijklmnopqrstuvwxyz0123456789’

Selection

```
>>>s1 = "abcdefg"

>>>s1[0]
'a'
>>>s1[2]
'c'
>>>s[-1]
'g'
>>>s[-2]
'f'
```

Slicing

```
>>>s1[1:3]
'bc'
>>>s1[:3]
'abc'
>>>s1[1:]
'bcdefg'
>>>s1[:]
'abcdefg'
>>>s1
'abcdefg'
```

LISTS

Top 10 Countries in 2012 Summer Olympic Medals

Rank	NOC	Gold	Silver	Bronze	Total
1	United States	46	29	29	104
2	China	38	27	23	88
3	Great Britain	29	17	19	65
4	Russia	24	25	32	81
5	Rep. of Korea	13	8	7	28
6	Germany	11	19	14	44
7	France	11	11	12	34
8	Italy	8	9	11	28
9	Hungary	8	4	6	18
10	Australia	7	16	12	35

How can we store this much data in Python?

Should we use 4×10 variables?

How many variables to store the medal information for all countries ?

The solution is to store the information in **lists**.

A **list** is an ordered sequence of date elements. It is a **mutable** object. In other words, existing elements can be **replaced** and **deleted**, and new elements can be **added**. In a list, the data elements are stored in a pair of **square brackets** unlike in a tuple.

list0 = [] (or list0 = list())

list1 = [1, 2, 3, 4, 5]

list2 = [1.57, 3.14, 9.02]

list3 = [“one”, “two”, “three”]

list4 = [1, 3.14, “two”, 2, (5, 6), [7,8]]

Now we can store the **2012 Summer Olympic Medal Table** using lists:

countries = [“United States”, “China”, ….. , “Australia”]

golds = [49, 38, ….. , 7]

silvers = [29, 27, ….. . 16]

bronzes = [29, 23, ….., 12]

Selection, slicing, and **concatenation** are allowed in lists as in tuples:

```
>>>counties[4]
'Rep. of Korea"
>>>golds[4]
13
>>>countries[-1]
'Australia'
>>>countries[2:4]
['Great Britain', 'Russia']
>>>golds[2:4]
[29, 24]
```

```
>>>list1 = countries[0:2]
>>>list1
['United states', 'China']
>>>list2 = countries[2:3]
['Great Britain']
>>>list1 + list2
['United States', 'China',
'Great Britain')]
```

The function **range()** generats a sequence integer, one at a time.

```
>>>range(0, 10, 2)
range(0, 10, 2)

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

```
for i in list(range(5)):
    print(i, end=" ")    What will be printed?
for i in [0, 1, 2, 3, 4]:
    print(i, end=" ")    What will be printed?
for i in ["smith", "John", "James"]:
    print(i, end =" ")    What will be printed?
for i in (0, 1, 2, 3, 4):
    print(i, end = " ")    What will be printed?
For i in "abcde"
    print(i, end =" ")    What will be printed?
```

How to change a list

```
>>>nobles = [“helium”,  “neno”, “argon”, “krypton”, “xenon”]
>>>nobles[1] = “neon”
>>>nobles
[“helium”, “neon”,  “argon”, “krypton”, “xenon”]

>>>nobles.append(“radon”)
>>>nobles
[“helium”, “neon”,  “argon”, “krypton”, “xenon”,  “radon”]
>>>len(nobles)
6
```

```
>>>nobles.insert(1, “nitrogen”)
>>>nobels
[“helium”, “nitrogen”,  “neon”,  “argon”, “krypton”, “xenon”, “radon”]
>>>len(nobles)
7
```

```
>>>nobles.remove("nitrogen")
>>>nobles
["helium", "neon",  "argon", "krypton", "xenon",  "radon"]
>>>len(nobles)
6
```

or

```
>>>gas = nobles.pop(1)
>>>nobles
["helium", "neon",  "argon", "krypton", "xenon",  "radon"]
>>>gas
'nitrogen'
>>>len(nobles)
6
```

```
>>>nobles.sort()
>>>nobles
["argon", "helium", "krypton", "neon", "radon", "xenon"]
>>>nobles.reverse()
>>>nobles
["xenon", "radon", "neon", "kryton", "helium", "argon"]
```

Never do the following !!

```
>>>list1 = nobles.sort()
>>>list1
None
```

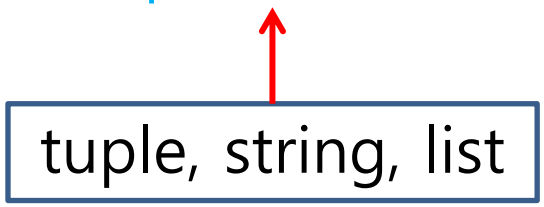
Other methods for lists

lst.pop() : Remove the last element of lst and return it.

lst.index(v): Return the index of the first element of lst which is equal to v.

lst.count(v): Count the number of elements in lst, which is equal to v.

lst.extend(K): Append all elements in a sequence K to lst.



>>>countries

["United States", "China", "Great Britain", "Russia", "Rep. of Korea", "Germany", "France", "Italy", "Hungary", "Australia"]

>>>golds

[46, 38, 29, 24, 13, 11, 11, 8, 8, 7] 195

>>>silvers

[29, 27, 17, 25, 8, 19, 11, 9, 4, 16] 165

>>>bronzes

[29, 23, 19, 32, 7, 14, 12, 11, 6, 12] 165

Functions: min, max, and sum

```
>>>min(golds), max(golds), sum(golds)
(7, 46, 195)
>>>min(silvers), max(silvers), sum(silvers)
(4, 29, 165)
>>>min(bronzes), max(bronzes), sum(bronzes)
(6, 32, 165)
```

Creating a **list** for **total numbers** of **medals** by **nations**

```
>>>totals = []
>>>for i in range (len(countries)):
        medals= golds[i] + silvers[i] + bronzes[i]
        totals.append((medals, countries[i]))
>>>print (totals)
```

```
[(104, 'United States'), (88, 'China'), (65, 'Great Britain'),
(81, 'Russia'), (28, 'Rep. of Korea'), (44, 'Germany'),
(34, 'France'), (28, 'Italy'), (18, 'Hungary'), (35, 'Australia')]
```

```
>>>totals.sort()
>>>totals
[(18, 'Hungary'), (28, 'Italy'), (28, 'Rep. of Korea'),
(34, 'France'), (35, 'Australia'), (44, 'Germany'), (65, 'Great Britain'), (81, 'Russia'), (88, 'China'), (104, 'United States')]

>>>totals.reverse()
>>>totals
[(104, 'United States'), (88, 'China'), (81, 'Russia'), (65, 'Great Britain'), (44, 'Germany'), (35, 'Australia'), (34, 'France'), (28, 'Rep. of Korea'), (28, 'Italy'), (18, 'Hungary')]
```

Printing **top five countries** in their total numbers of medals

```
>>>total[0:5]
[(104, 'United States'), (88, 'China'), (81, 'Russia'),
(65, 'Great Britain'), (44, 'Germany')
```

Unpacking

```
top_five = totals[0:5]
for p in top_five:
    medals, nation = p
    print (medals, nation)
```

```
top_five = totals[0:5]
for medals, nation in top_five:
    print (medals, nation)
```

104 United States
88 China
81 Russia
65 Great Britain
44 Germany

Printing the **top five countries in lexicographical ranking**

```
table = []
for i in range(len(countries)):
    table.append((golds[i], silvers[i], bronzes[i], countries[i]))
print (table)
```

```
[(46, 29, 29, 'United States'), (38, 27, 23, 'China'),
(29, 17, 19, 'Great Britain'), (24, 25, 32, 'Russia'),
(13, 8, 7, 'Rep. of Korea'), (11, 19, 14, 'Germany'),
(11, 11, 12, 'France'), (8, 9, 11, 'Italy'),
(8, 4, 6, 'Hungary'), (7, 16, 12, 'Australia')]
```

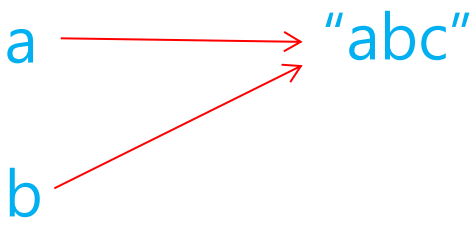
```
>>>table.sort()
>>>top_five = table[-5:]
>>>top_five .reverse()
>>>for g, s, b, nation in top_five:
        print (g, s, b, nation)
```

What is it?

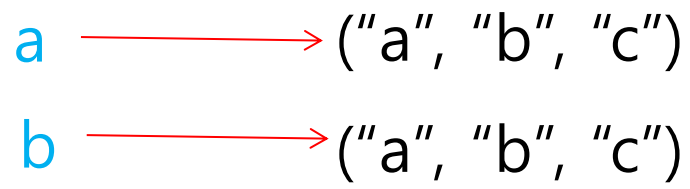
```
46 29 29 United States
38 27 23 China
29 17 19 Great Britain
24 25 32 Russia
13 8 7 Rep. of Korea
```

(

```
>>> a = "abc"
>>> b = "abc"
>>> a == b
True      equivalent
>>>a  is  b
True      identical
```

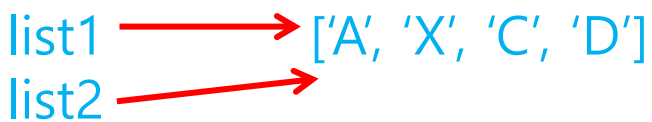



```
>>> a = ("a", "b", "c")
>>> b = ("a", "b", "c")
>>> a == b
True      equivalent
>>>a  is  b
False     not identical
```



Aliasing

```
>>>list1 = ["A", "B", "C"]
>>>list 2 = list 1
>>>list2 .append("D")
>>>list2
['A', 'B', 'C', 'D']
>>>list1[1] = "X"
>>>list2
['A', 'X', 'C', 'D']
>>>list1 is list2
True
```




```
>>>list1 = ["A", "B", "C"]
>>>list2 = ["A", "B", "C"]
>>>list2 .append("D")
>>>list2
['A', 'B', 'C', 'D']
>>>list1[1] = "X"
>>>list2
['A', 'B', 'C', 'D']      Why?
>>>list1 is list2
False
```

SUMMARY: STRINGS, TUPLES, & LISTS

- **Strings, tuples, and lists** all represent **sequences of elements**.

- **Tuples and lists** are very **similar**: the only difference is their **mutability**, i.e., **tuples** are **immutable** while **lists** are **mutable.**

- **Strings** can be conceptually regarded as a form of **tuples** restricted to **text data**, although their representation schemes are quite **different**.

Type conversion

tuples ⟷ lists

```
>>>stg = "abcde"
>>>tpl = ("a", "b", "c", "d", "e")
>>>lst =  ["a", "b", "c", "d", "e"]

>>>tuple(lst)
("a", "b", "c", "d", "e")
>>>list(tpl)
["a", "b", "c", "d", "e"]
```

strings ⟶ tuples, lists

```
>>>tuple(stg)
("a", "b", "c", "d", "e")
>>>list(stg)
["a", "b", "c", "d", "e"]
```