Introduction to Programming Using Python (II)

Fadoua Ghourabi
Ochanomizu University
ghourabi.fadoua@ocha.cc.ac.jp

April 25, 2019

Before we start...

Few points I forgot in the previous class

1) Indention

- Indention is mandatory in python and used to delimit a block (if-else block, while-loop block, for-loop block)
- Remember in Java and C, we use "{" and "}" to delimit a block
- Indention is automatic... but do not forget ":" after if, elif, else, while, for, def

Before we start...

2) Useful operations on `str`

- Use operator `in` to check if a character or a substring is in a string e.g. `'champions' in 'We are the champions!'`
- Use a built-in function `len` to compute the length of an object

Basic of programming

Elements of any programming language

1. Objects, types and basic operations
2. Variables and assignment
3. Conditionals
4. Loops
5. Functions
6. Data structures

1) Tuple

- A tuple is an ordered sequence of elements
- The elements of a tuple can be of any type
- The elements of a tuple need not be of the same type

```
t1 = () # empty tuple
t1 = (0, 'one', 2.0) # elements of different types
```

- Tuples can be repeated, concatenated, indexed and sliced

Try:

```
t1, t2 = ('a', 'b', 'c'), (2, 5/6)
2*t1
t1 + t2
t2[0]
(t1 + t2)[3]
(2*t1 + t2)[:6]
```

- Tuple may contain tuples

```
t3 = (t1, 3.25)
```

1) Tuple

- `for` statement can be used to iterate over a tuple

Practice.

Write a function that computes the intersection of two tuples

```
t1, t2, t3 =('a','c'), ('k', 'm', 'n', 'p', 'q'), ('a','b','c')
intersect(t1,t2) returns ()

intersect(t1,t3) returns ('a', 'c')
```

Note.

1. How do we represent a singleton tuple?
   - `(1) = 1` and `(3+4) = 7` are of type `int`
   - but `(1,)` and `(3+4,)` are singleton tuples containing 1 and 7 respectively

1) Tuple

- Functions use tuples to return more than one values
- `intersect(t1,2*t1+t2)` returns `(('a', 'b', 'c'), 3)`
- We can safely write

  ```
  x, y = intersect(t1, 2*t1+t2)
  ```

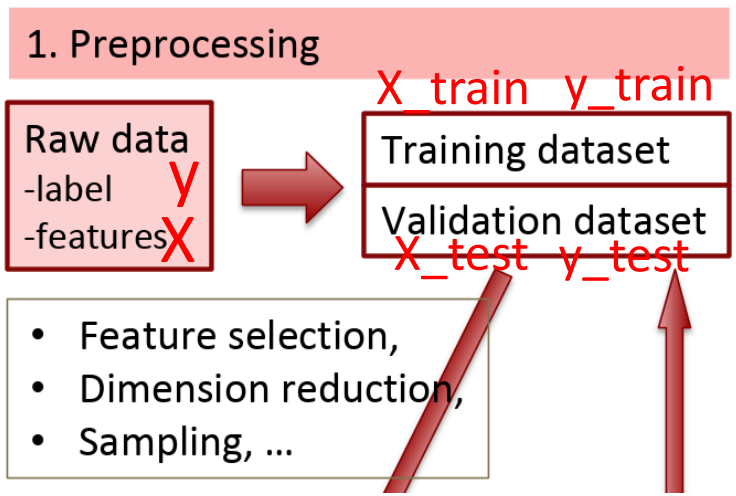
  because we know that the function returns a tuple of two elements.

```
def intersect(s1,s2):
  result= ()
  for e in s1:
    if e in s2:
      result = result + (e,)
  return result, len(result)
```

1) Tuple

- A common example in pre-processing for machine learning: splitting a data into training data and test data
  - How do you split your data?
  - SciKit library provides a tool to split in a random manner
  - Function `train_test_split` returns a tuple of training datasets and validation/test datasets

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state=0)
```



First step of Megumi san's PBTS study plan

test_size = 0.2 means the validation dataset is 20% of the raw data
random_state is a parameter for the random split
... but we will discuss these points later in machine learning chapter.

2) List

- A list is an ordered sequence of elements
- The elements of a list can be of any type
- The elements of a list need not be of the same type

```
L1 = [] # empty list
L1 = [0, 'one', 2.0] # elements of different types
```

- Lists can be repeated, concatenated, indexed and sliced

Try:

```
L1, L2 = ['a', 'b', 'c'], [2, 5/6]
2*L1
L1 + L2
L2[0]
(L1 + L2)[3]
(2*L1 + L2)[:6]
```

- List may contain lists

```
L3 = [L1, 3.25]
```

2) List

Let L be a list, we can mutate L in the following way

- L.append(e) adds the object e to the end of L
- L.insert(i, e) inserts the object e in L at index i
- L.extend(L1) adds the elements of L1 to the end of L (same result as L = L + L1)
- L.pop() removes and returns the last element of L (exception when L = [] empty)
- L.pop(i) removes and returns the element of L at index i (exception when L = [] empty)
- L.sort() sorts the elements of L in an ascending order
- L.reverse() reverses the elements of L

Other operations (no mutation)

- L.count(e) returns the number of occurrence of e in L
- L.index(e) returns the index of the first occurrence of e in L (exception when e is not in L)

2) List

Practice.

1. Write a function `summation` that computes the sum of a list of integer
   `summation([0,7,13,2,66,85,10]) = 183`
2. Write a function `map_sum` that maps function `summation` to list of lists
   `map_sum([[0],[2,6,7],[5,12]]) = [0, 15, 17]`
3. What is the result of
   `[summation(l) for l in [[0],[2,6,7],[5,12]]]`?

2) List

- List comprehension is a concise way to apply an operation to elements of a list
  Try:
  ```
  [summation(l) for l in [[0],[2,6,7],[5,12]]]
  [x % 2 for x in [0,'a',13,'?',66,85,10] if type(x) == int]
  ```
- List comprehension is an invention of Python
- Which programming style has better legibility for you? List comprehension or old-fashioned loop?

```
L = [0,'a',13,'?',66,85,10]
L = [x % 2 for x in L if type(x) == int]
```

```
L = [0,'a',13,'?',66,85,10]
newL = []
for x in L:
    if type(x) == int:
        newL.append(x%2)
L = newL
```

Tuple vs. List

- A list is an ordered sequence of elements ... but so is tuple
- What is the difference between tuple and list?
  - List is a mutable structure = can be modified
  - Tuple is an immutable structure = cannot be modified

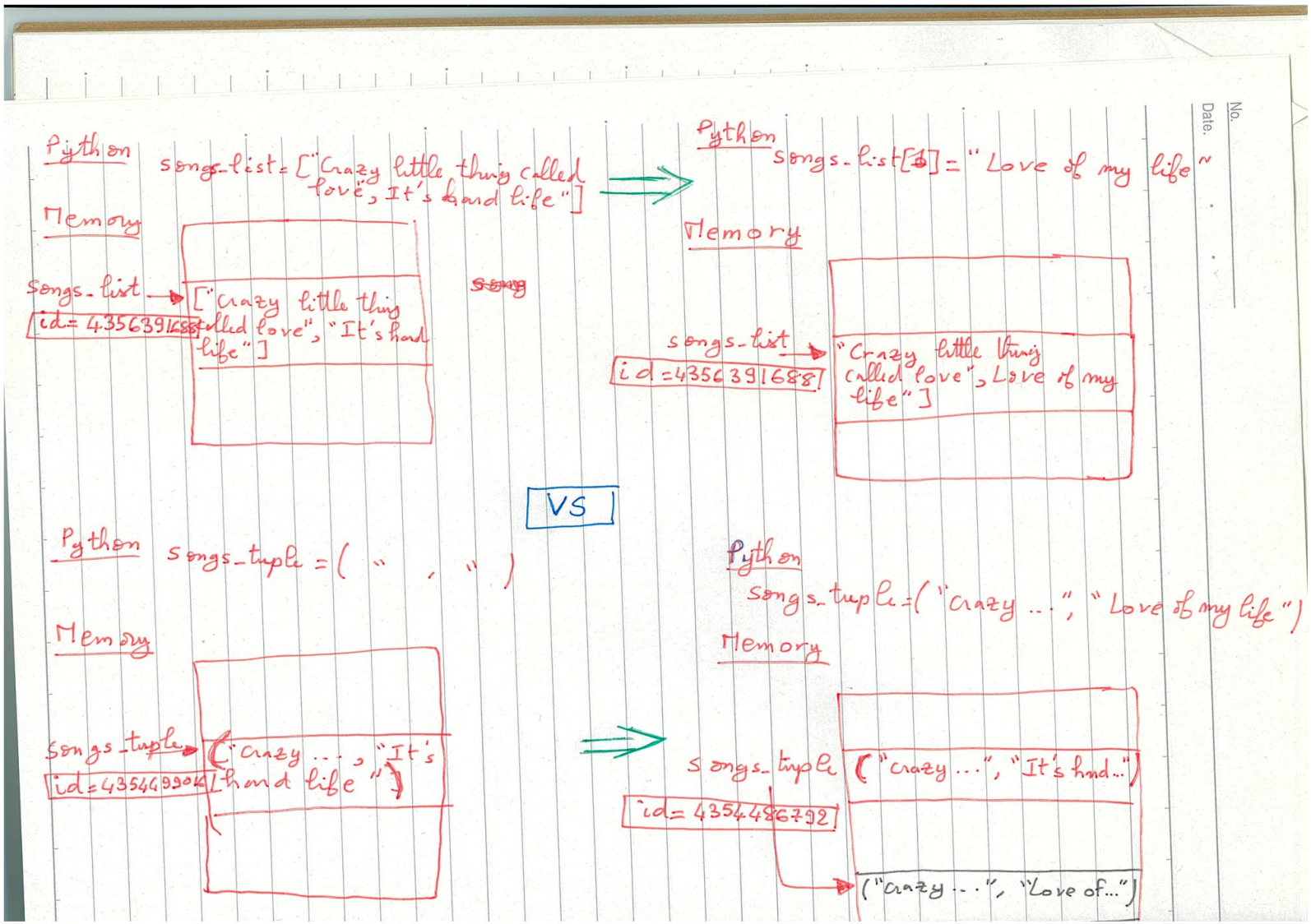
```
songs_list = ["Crazy little thing called love", "It's a hard life"]
songs_tuple = ("Crazy little thing called love", "It's a hard life")
songs_list[1] = "Love of my life"  # allowed
songs_tuple[1] = "Love of my life"  # error
```

```
In [53]: songs_tuple[1]="Love of my life"

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-53-808888f78813> in <module>()
----> 1 songs_tuple[1]="Love of my life"

TypeError: 'tuple' object does not support item assignment
```

Tuple vs. List



Tuple vs. List

Computation efficiency (time and memory)

```
import time
start = time.time()
L=[]
for x in range(10000):
    L.append(x)
end = time.time()
print(end-start)
```

- One object
- Computation time is about 0.00165s

```
import time
start = time.time()
T=()
for x in range(10000):
    T = T + (x,)
end = time.time()
print(end-start)
```

- At least 10000 objects
- Computation time is about 0.1645s

List wins!

Tuple vs. List

Tracking how an object changes (debugging)

```
L1 = [1,2,3]
L2 = L1
L2[0] = 0
[4 / x for x in L1]
```

- Error: division by zero!
- Imagine you are working on a big project with many references to the same list object. It is difficult to track the changes.

```
T1 = (1,2,3)
T2 = T1
T2 = (0,2,3)
[4 / x for x in T1]
```

- No division by zero because T1 = (1, 2, 3)
- result [4.0, 2.0, 1.333]

Tuple wins!

Dictionary

- A dictionary is a set of key/value pairs
  `studentID = {'name':'X Y', 'gender':'F', 'age':22, 'address':'Tokyo'}`
- We use the key to access elements of a dictionary
  `studentID['gender']` return `'F'`
- Careful: studentID[2] refers to the key 2 rather than the 23th element.
- Try:
  (1) In a for statement, the counter iterates over keys

```
for e in studentID:
    print(studentID[e])
```

  (2) To add new key/value pair
  `studentID['grades']=[10,8,6]`
  (3) To remove a key/value pair
  `studentID.pop('grades')`

Problem: sort algorithms

1. Write a function bubble_sort(list) to sort a list k.
   Bubble short compares two adjacent elements i and i+1 and swap them if k[i] $>$ k[i+1]
2. Write a function selection_sort(list) to sort a list k
   Selection sort split a list into ordered sublist and unordered sublist. The algorithm searches for the smallest element and placed it at the end of the ordered sublist.