Python Structure:

1. String

- x = 'From marquard@uct.ac.za' → uct → print[14:17] → slice kí tự 14 đến kí tự 17 nhưng k chứa kí tự 17

- str = '123'

x = int(str) + 1

- looping thru str: using for statement is much more elegant & the interation variable is completely taken care of by the for loop

fruit = 'banana'

for letter in fruit:

print(letter)

- str function

Search a str: we use the find() function to search for the substring, if the substring isn't found, find() return -1

lstrip(): remove whitespace at the left

rstrip(): remove whitespace at the right

>>> greet = ' Hello Bob '

>>> greet.lstrip()

'Hello Bob '

>>> greet.rstrip()

' Hello Bob'

>>> greet.strip()

'Hello Bob'

line.startswith('From')

- str are "immutable"- we must make a new str to make any change

2. List

Algorithms: A set of rules or steps used to solve a problem

Structure: A paticular way or organizing data in a computer

- A collection allows us to put many values in a single "valuable" (list, dict, tuple)

- looping to iterate thru a list: using for/in statement

- lists are "mutable"- we can change an element of a list using the index operator (append, sort, max, min, sum)

- list function

The range function returns a list of number

friends = ['Joseph', 'Glenn', 'Sally']

>>> print(len(friends))

3

>>> print(range(len(friends)))

[0, 1, 2]

- str & lists

>>> abc = 'With three words'

>>> stuff = abc.split()

>>> print(stuff)

['With', 'three', 'words']

>>> line = 'first;second;third'

>>> thing = line.split(';')

>>> print(thing)

['first', 'second', 'third']

fhand = open('mbox-short.txt')

for line in fhand:

line = line.rstrip() #strip the white space off the right-hand

if not line.startswith('From') : continue #skip the line not start 'From'

words = line.split() #chop base on space

print(words[2])

→ Sat

3. Dict

- {'key' : 'value'}

- dict are "mutable"

- the get method for dict

if name in counts:

x = counts[name]

else:

x = 0

↔ x = counts.get(name, 0)

→ counts.get go look up in counts, use name as the key and 0 as the default, meaning this is the value I get back if the key doesn't exist

We can use get() and provide a default value of zero when the key isn't yet in the dictionary- and then just add one

counts = dict()

names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']

for name in names:

counts[name] = counts.get(name, 0) + 1

print(counts)

- definite loops and dict: can write a for loop that goes thru all the entries in a dict- it goes thru all of the key in the dict and looks up the values

- convert dict to list

>>> jjj = { 'chuck' : 1, 'fred' : 42, 'jan': 100}

>>> print( jjj.items())

[ ('jan', 100) , ('chuck', 1) , ('fred' , 42)] → list of the key

- items can be used thru for loop

jjj = { 'chuck' : 1, 'fred' : 42, 'jan': 100}

for k, v in jjj.items():

print(k, v)

→ jan 100

chuck 1

fred 42

4. Tuple

- Tuple is the functions much like a list but "immutable"

- the items() method in dict return a list of (key, value) tuples

>>> d = dict()

>>> d['csev'] = 2

>>> d['cwen'] = 4

>>> for (k,v) in d.items():

... print(k,v)

…

csev 2

cwen 4

>>> tups = d.items()

>>> print(tups)

dict\_items ([( 'csev', 2) , ('cwen', 4)])

- sorted(items) would give us the items sort by key

- Learn: list comprehention