Programming in Python
Lecture 2- Lists, Tuples, Dictionaries

Plan for today

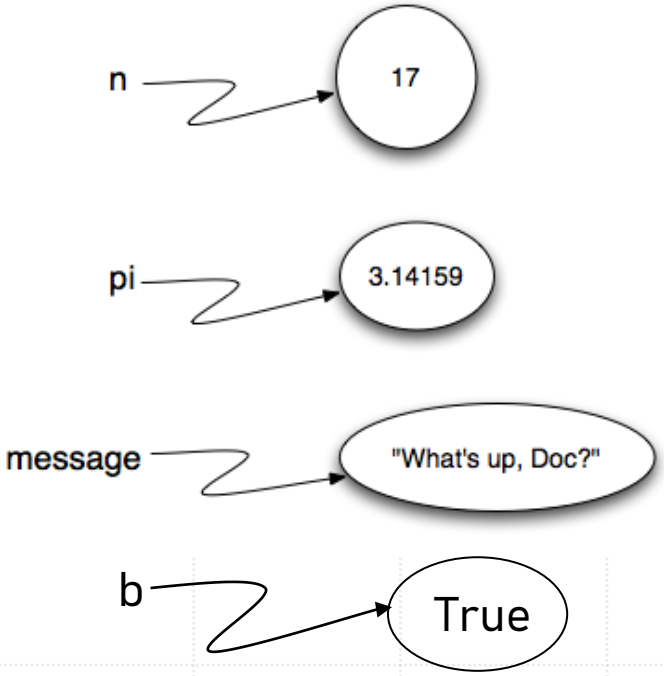
- Lists
- Tuples
- Dictionaries

Reminder

- Variables
- Numbers
- Strings
- Computational Operators
- Logical Operators

Why Do We Need Different Types?

- Saving memory
- Execution speed
- Variables types:
  - Int(integer)
  - Float(numbers with decimal point)
  - Strings(text sequences)
  - Booleans(True or False)



Arithmetic Operators

Operator	Use	Description
+	x + y	Adds x to y
-	x - y	Subtracts x from y
*	x * y	Multiplies x by y
**	x ** y	X to the power y
/	x / y	Divides x by y
%	x % y	Computes the remainder of dividing x by y

Strings Slicing

```
str="51689”
print( str[1])
>>>‘1‘
print( str[0:3])
>>>‘516‘
print( str[1:])
>>> '1689"
print( str[-3:-1])
>>> ‘16’
print( str[:-3])
>>> ‘51"
print([::-1])
>>> ‘98615’
```

5	1	6	8	9
0	1	2	3	4
-5	-4	-3	-2	-1

Comparison Operators

- Compares two variables and returns a Boolean type result/variable

Operator	Name	Description
x < y	Less than	true if x is less than y, otherwise false.
x > y	Greater than	true if x is greater than y, otherwise false.
x <= y	Less than or equal to	true if x is less than or equal to y, otherwise false.
x >= y	Greater than or equal to	true if x is greater than or equal to y, otherwise false.
x == y	Equal	true if x equals y, otherwise false.
x != y	Not Equal	true if x is not equal to y, otherwise false.

Logical Operators

Operates on two Booleans and returns a Boolean

Operator	Description
x and y	Both True: **True**, otherwise: **False**.
x or y	At least one is True: **True**, Otherwise: **False**.
not x	x is False → **True**, x is True → **False**

Homework

1. create a Python script which accepts the user's first and last name and print them in reverse order with a space between them

input:

>>> ‘Dor Shani’

Output:

>>> ‘Hello Shani Dor’

2. create a Python script which accepts an integer (n) and computes the value of n+nn+nnn.

input:

>>>n =5

Output:

>>> 615

Homework

3. create a Python script which get a string from a given string where all occurrences of its first char have been changed to '$',except the first char itself.

Input:

>>> ‘restart’

Output:

>>> ‘resta$t’

4. create a Python script which single string from two given strings, separated by a space and swap the first two characters of each string.

Input:

>>> ‘abc’, ‘xyz’

Output:

>>> ‘xyc’, ‘abz’

Plan for today

- **Lists**
- Tuples
- Dictionaries

Lists

A list is an ordered sequence of elements.

Create a list in Python:

```
>>> list = [9,8,6,1,5]
>>> print(list)
[9,8,6,1,5]
```

Lists are Indexable

```
>>> list = [9,8,6,1,5]
>>> list[0]
9
>>> list[4]
5
>>> list[-3]
6
>>> list[::2]
[9,6,5]
>>> my_list[5]
Traceback (most recent call last):
File "<pyshell#7>", line 1, in <module>
my list[5]
IndexError: list index out of range
```

9	8	6	1	5
0	1	2	3	4
-5	-4	-3	-2	-1

Slicing

```
# reverse
>>>list[::-1]
[5, 1, 6, 8, 9]
```

```
# slicing does NOT change original list
>>>list
[9,8,6,1,5]
```

Lists

Lists can contain strings:

```
>>> Days = ["Sun", "Mon", "Tue", "Wed",
 "Thu", "Fri", "Sat"]
>>> Days[5]
'Fri'
>>> len(Days)
7
```

Lists can contain any type of object:

```
>>> student = ['Harely', 27, 1.65, 710]
```

Lists – Dynamic

Maintain a list of the students either by name or by id:

```
>>> students = [‘Sharon’,9654520, ‘Liat’,837561405
  ‘Yaniv’,968554353]
>>> students[2]
‘Liat’
```

- Roze decided to join the course, so we update the list:

```
# append - add an element to the end of the list
>>> students.append(‘Roze')
>>> students
students = [‘Sharon’,9654520, ‘Liat’,837561405
  ‘Yaniv’,968554353,’Roze’]
```

Lists – Dynamic

- Yaniv wants to leave the course:

```
>>> students.remove('Yaniv')
>>> students
['Sharon', 9654520, 'Liat', 837561405, 968554353,
 'Roze']
```

remove removes only **the first** occurrence of a value.

Assignment rules in Python

Assignments of List Variables

```
>>> list_1= [1,2,3]
>>> list_2= list_1
>>> list_1 = [6,7,8,9]
>>> list_2
[1,2,3]
>>> list_1
[6,7,8,9]
```

So far - no surprises

Assignments of List Variables

```
>>> list_2= list_1
>>> list_1[0] = 1000
>>> list_1
[1000,7,8,9]
>>> list_2
[1000,7,8,9]
```

Surprise!

Nested Lists

```
NL = [ [3, 7, 2],[6, 0, 1] ]
Print(NL[1])
>>>[6, 0, 1]
Print(NL[1][0])
>>> 6
len(NL)
>>> 2
```

List Methods

Function	Description
L. append(elem)	Adds an element at the end of the list.
L. clear()	Removes all the elements from the list
L. copy()	Returns a copy of the list
L. count()	Returns the number of elements with the specified value
L. extend(elem)	Add the elements of a list (or any iterable), to the end of the current list
L. index(num)	Returns the index of the first element with the specified value
L. insert(elem)	Adds an element at the specified position
L. pop(elem,index)	Removes the element at the specified position
L. remove(elem)	Removes the first item with the specified value
L. reverse()	Reverses the order of the list
L. sort()	Sorts the list
sum(L)	Returns the sum of elements with in the list

Questions?

Hands On

Plan for today

- **Lists**
- **Tuples**
- Dictionaries

Tuples are immutable lists

```
>>> list = [1,2,3]
>>> list [1]=10
>>> tuple = (1,2,3)
>>> tuple[1] = 10
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
  my_tuple[1] = 10
TypeError: 'tuple' object does not support item assignment
```

Tuples are immutable lists

- Why use Tuples?
  - Faster than lists in certain operations
  - The interpreter enforces that they are not changed, and we'll see why this is useful.

Python's types: Mutable vs. Immutable

Mutable	Immutable
list	Numeric types: int, float, complex
set	tuple
dict	string

Tuples

A tuple is similar to a list, but it is immutable.

```
>>> B = (“Let", “It", "be") # definition
>>> B
(“Let", “It", "be")
>>> B[0]      # indexing
"Let"
>>> B[-1]     # backwords indexing
'Be'
>>> B[1:2]    # slicing
('It')
```

Tuples

```
>>> t[0] = 'do'  # try to change
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    t[0]='do'
TypeError: 'tuple' object does not support item assignment
```

No append / extend / remove in Tuples!

Tuples

- Fixed size
- Immutable (similarly to Strings)
- What are they good for (compared to list)?
  - Simpler (“light weight”)
  - Staff multiple things into a single container
  - Immutable (e.g., records in database, safe code)

Questions?

Hands On

Plan for today

- **Lists**
- **Tuples**
- **Dictionaries**

Dictionaries

- Key – Value mapping
  - No order
- Fast!
- Usage examples:
  - Database
  - Dictionary
  - Phone book

key	value
firstName	Bugs
lastName	Bunny
location	Earth

Dictionaries

Access to the data in the dictionary:

- A key is one-to-one function
- Given a key, it is easy to get the value.
- Given a value, you need to go over all the dictionary to get the key.

Dictionaries

Dictionary: a set of key-value pairs.

```
>>> dict_1= {key1:val1, key2:val2,…}
```

Keys are **unique** and **immutable**.

Dictionaries

Example: ID list- Map names to IDs:

```
>>> ID_list = {'Eric': '30145', 'Shlomi': '38171',
  'Kobi': '85736'}
>>> print(ID_list)
{'Eric': '30145', 'Shlomi': '38171', 'Kobi':
  '85736'}
```

Note:The pairs order changed!

Dictionaries

Access dictionary Items:

```
>>> ID_list ['Eric’]
'30145'
```

Add a new person:

```
>>> ID_list[‘David'] = ‘84759'
>>> print(ID_list )
{'Eric': ‘30145', ‘Shlomi': ‘38171', ‘Kobi':
  ‘85736’, ‘David’:‘84759' }
```

Dictionaries

What happens when we add a key that already exists?

```
>>> ID_list ['David']= '75647'

>>> print(ID_list)

{'Eric': '30145', 'Shlomi': '38171', 'Kobi':
  '85736', 'David':'75647 ' }
```

How can we add another Kenny McCormick in the phone book?

Dictionary Methods

Function	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
items()	Returns a list containing a tuple for each key value pair
popitem()	Removes the last inserted key-value pair
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Questions?

Hands On

Summary

- Lists
- Tuples
- Dictionaries

Homework

1. What is the fifteenth letter of the alphabet?
2. What is the code for the twenty-third letter of the alphabet?
3. What is the fourth letter of the code for the eighth letter of the alphabet?

Input:

```
>>> Alphabet = [["A", "Alfa"],["B", "Bravo"],["C", "Charlie"], ["D", "Delta"], ["E", "Echo"],
["F", "Foxtrot"], ["G", "Golf"], ["H", "Hotel"], ["I", "India"], ["J", "Juliett"], ["K", "Kilo"],
["L", "Lima"], ["M", "Mike"], ["N", "November"], ["O", "Oscar"], ["P", "Papa"],
["Q", "Quebec"], ["R", "Romeo"], ["S", "Sierra"], ["T", "Tango"], ["U", "Uniform"],
["V", "Victor"], ["W", "Whiskey"], ["X", "X-ray"], ["Y", "Yankee"], ["Z", "Zulu"]]
```

Homework

4. Create a Python script that to Convert a given tuple of positive integers into an integer

Input:

```
>>>  nums=(1,2,3)
```

Output:

```
>>>  123
```

5. Create a Python script that change the first elemant in a tuple

Input:

```
>>> x = ("apple", "banana", "cherry")
```

Output:

```
>>>  ('kiwi', 'banana', 'cherry')
```

Homework

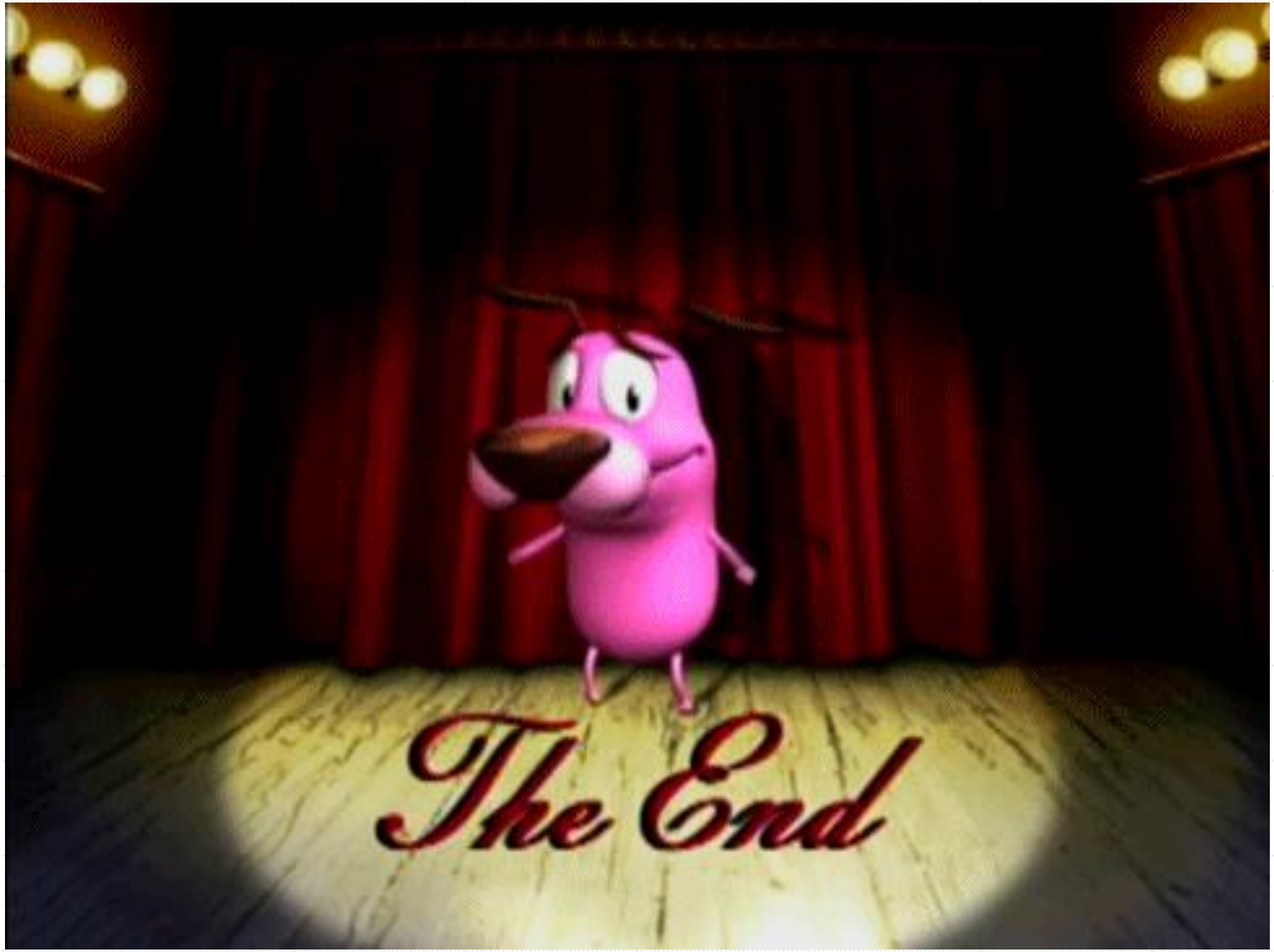
6. Create a Python script that combine values in python list of dictionaries.

Input:

```
>>> item_list = [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}]
```

Output:

```
>>> {'item1': 400, 'item2': 300}
```


The End