ANL251 Python Programming

Study Unit 3

Tuples and Dictionaries

Learning Outcomes and Learning Resources

1. Use the Python tuple data structure and its operations appropriately, including indexing, subsetting, unpacking and iterating elements in tuples
   - SU3 Chapter 1
   - https://docs.python.org/3/library/stdtypes.html#typesseq
2. Explain the immutable nature of the Python tuple
   - SU3 Chapter 1
   - https://docs.python.org/3/library/stdtypes.html#typesseq
3. Create Python dictionary and implement operations including indexing, getting, adding or removing elements in dictionary
   - SU3 Chapter 2
   - Textbook Video and Exercise 39
   - https://docs.python.org/3/library/stdtypes.html#typesmapping

Learning Outcomes and Learning Resources

4. Implement sorting on Python dictionaries based on the keys
   - SU3 Chapter 2
   - Textbook Video and Exercise 39
5. Solve problems using Python dictionaries and for-loop
   - SU3 Chapter 2
   - Textbook Video and Exercise 39
   - https://docs.python.org/3/library/stdtypes.html#typesmapping

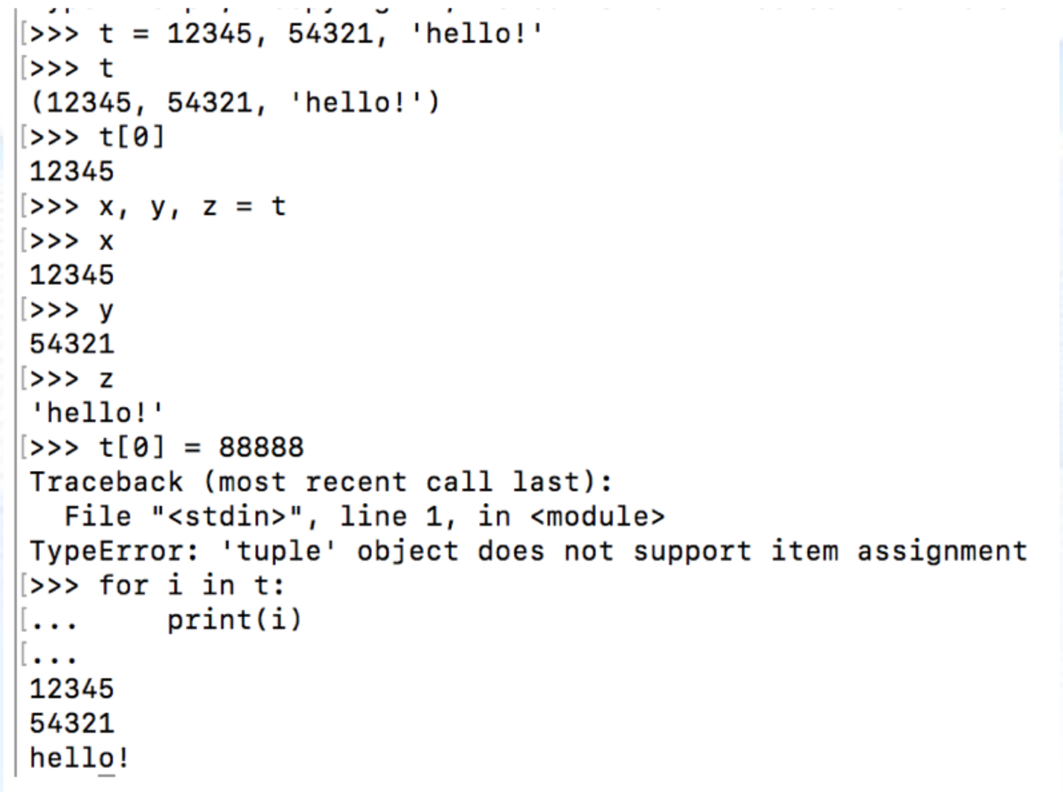
Seminars: discussion and activities to reinforce students' understanding

1. Operations on Tuples

2. Immutable Nature of Tuples

Recap

Operations on Tuples and its Immutable Nature (SU3 Chapter 1 https://docs.python.org/3/library/stdtypes.html#typesseq)



```
>>> t = 12345, 54321, 'hello!'
>>> t
(12345, 54321, 'hello!')
>>> t[0]
12345
>>> x, y, z = t
>>> x
12345
>>> y
54321
>>> z
'hello!'
>>> t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> for i in t:
...     print(i)
...
12345
54321
hello!
```

Figure 3.1 Creating and manipulating tuples

Discussion

Among the Python data types we have learned so far, which are immutable and which are mutable?

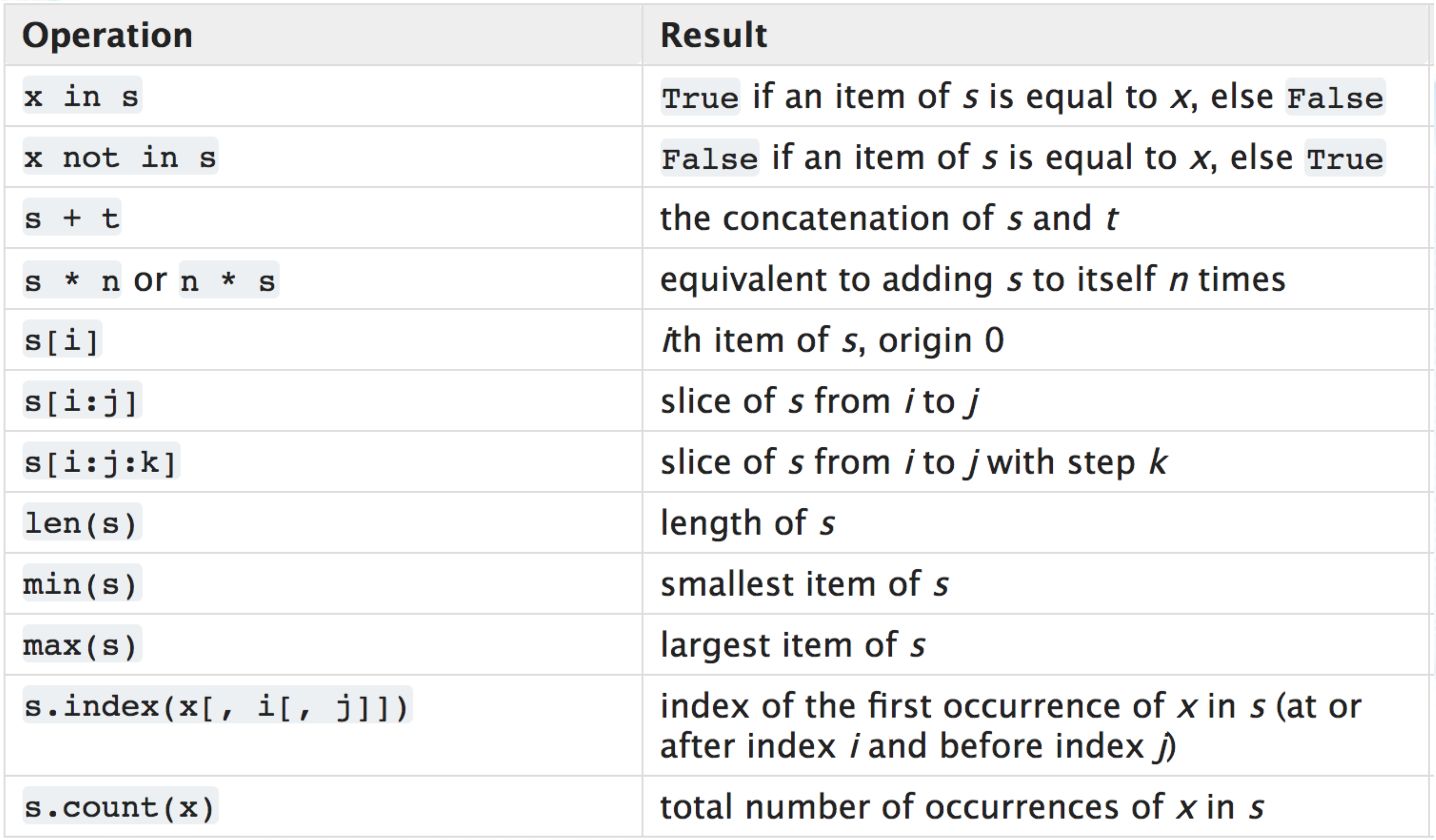
Note:

– Because tuples are immutable, they are more efficient in terms of performance and memory use.
– Tuples are useful in situations where you want to share the data with others but not allow them to modify the data. They can use the data values, but no change is reflected in the original data shared.

Recap

Common sequence operations on tuple (also applicable to **list** in SU2)

https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

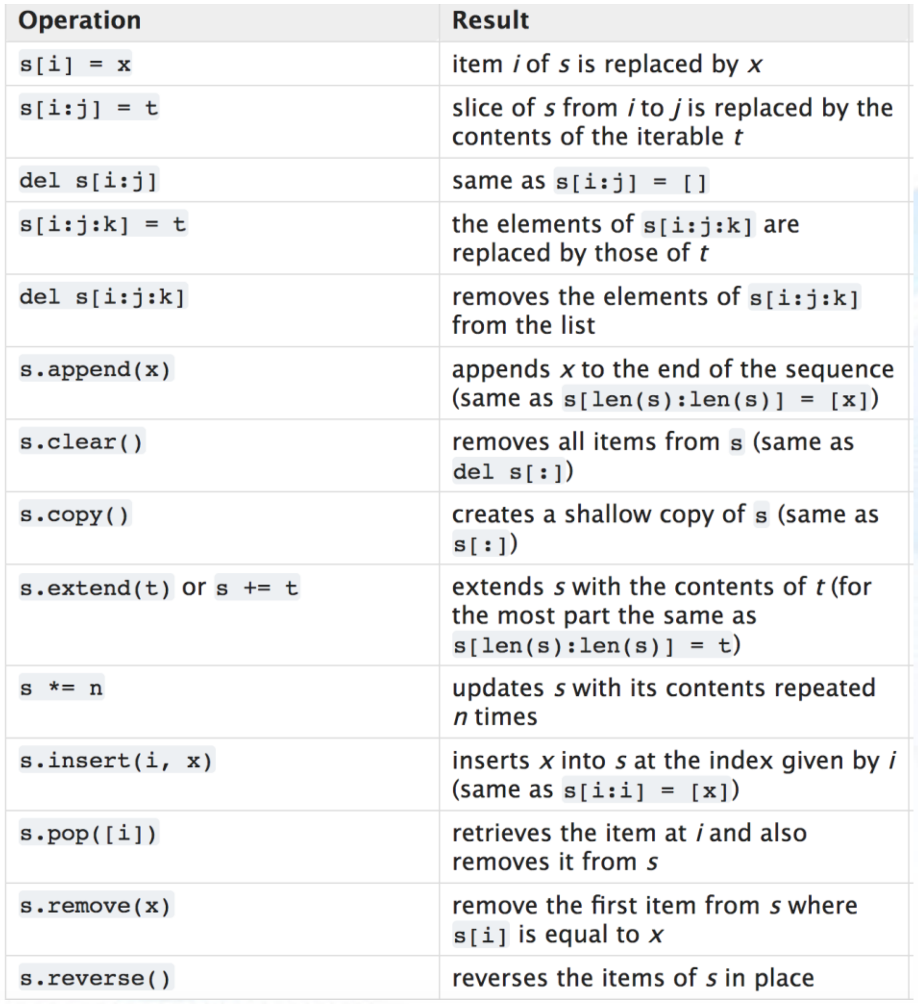


Operation	Result
`x in s`	`True` if an item of *s* is equal to *x*, else `False`
`x not in s`	`False` if an item of *s* is equal to *x*, else `True`
`s + t`	the concatenation of *s* and *t*
`s * n` or `n * s`	equivalent to adding *s* to itself *n* times
`s[i]`	*i*th item of *s*, origin 0
`s[i:j]`	slice of *s* from *i* to *j*
`s[i:j:k]`	slice of *s* from *i* to *j* with step *k*
`len(s)`	length of *s*
`min(s)`	smallest item of *s*
`max(s)`	largest item of *s*
`s.index(x[, i[, j]])`	index of the first occurrence of *x* in *s* (at or after index *i* and before index *j*)
`s.count(x)`	total number of occurrences of *x* in *s*

Recap

Operations on lists
(https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types)

NOT applicable to **tuple**!



Operation	Result
`s[i] = x`	item *i* of *s* is replaced by *x*
`s[i:j] = t`	slice of *s* from *i* to *j* is replaced by the contents of the iterable *t*
`del s[i:j]`	same as `s[i:j] = []`
`s[i:j:k] = t`	the elements of `s[i:j:k]` are replaced by those of *t*
`del s[i:j:k]`	removes the elements of `s[i:j:k]` from the list
`s.append(x)`	appends *x* to the end of the sequence (same as `s[len(s):len(s)] = [x]`)
`s.clear()`	removes all items from `s` (same as `del s[:]`)
`s.copy()`	creates a shallow copy of `s` (same as `s[:]`)
`s.extend(t)` or `s += t`	extends *s* with the contents of *t* (for the most part the same as `s[len(s):len(s)] = t`)
`s *= n`	updates *s* with its contents repeated *n* times
`s.insert(i, x)`	inserts *x* into *s* at the index given by *i* (same as `s[i:i] = [x]`)
`s.pop([i])`	retrieves the item at *i* and also removes it from *s*
`s.remove(x)`	remove the first item from *s* where `s[i]` is equal to *x*
`s.reverse()`	reverses the items of *s* in place

Quiz

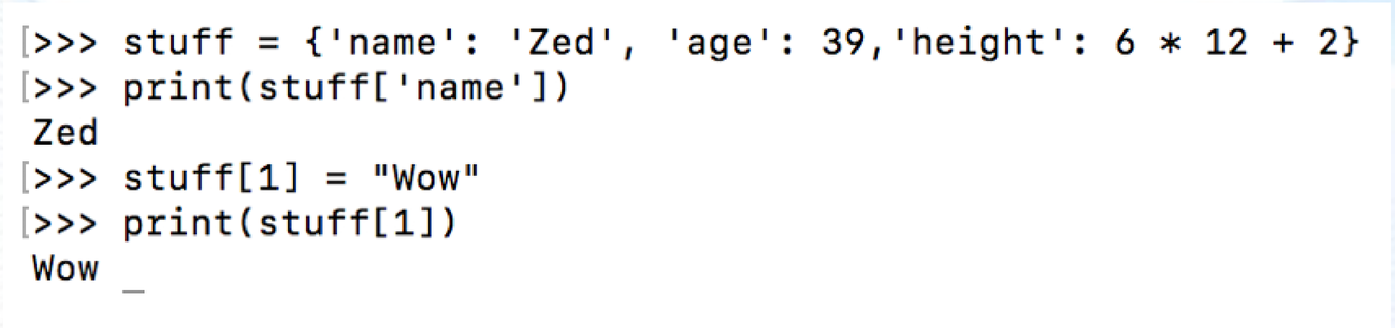
The variable grades refers to (70,80,90). What does each of the following evaluate to?

```
grades[-1]
grades[grades.index(90)]
grades.pop()
grades[len(grades)-1]
```

3. Operations on Dictionaries

Recap

Operations on dictionaries (SU3 Chapter 2, Textbook Video and Exercise 39, https://docs.python.org/3/library/stdtypes.html#typesmapping)



```
>>> stuff = {'name': 'Zed', 'age': 39,'height': 6 * 12 + 2}
>>> print(stuff['name'])
Zed
>>> stuff[1] = "Wow"
>>> print(stuff[1])
Wow _
```

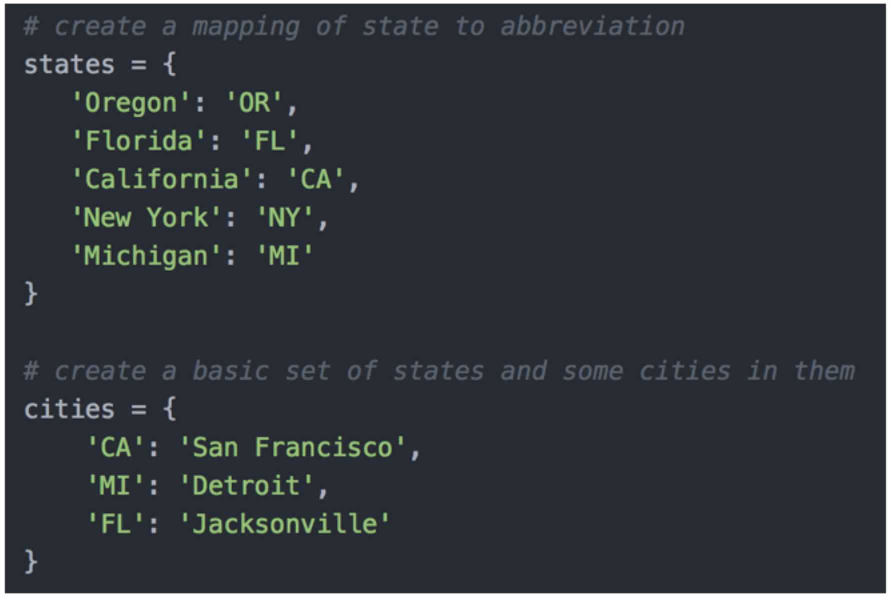
Figure 3.2 Comparing indexes of lists and dictionaries

Note:

- Unlike lists and tuples, which are indexed by a range of numbers, dictionaries are indexed **by keys**.
- Keys must be unique. Values may be duplicated.

Recap

Operations on dictionaries

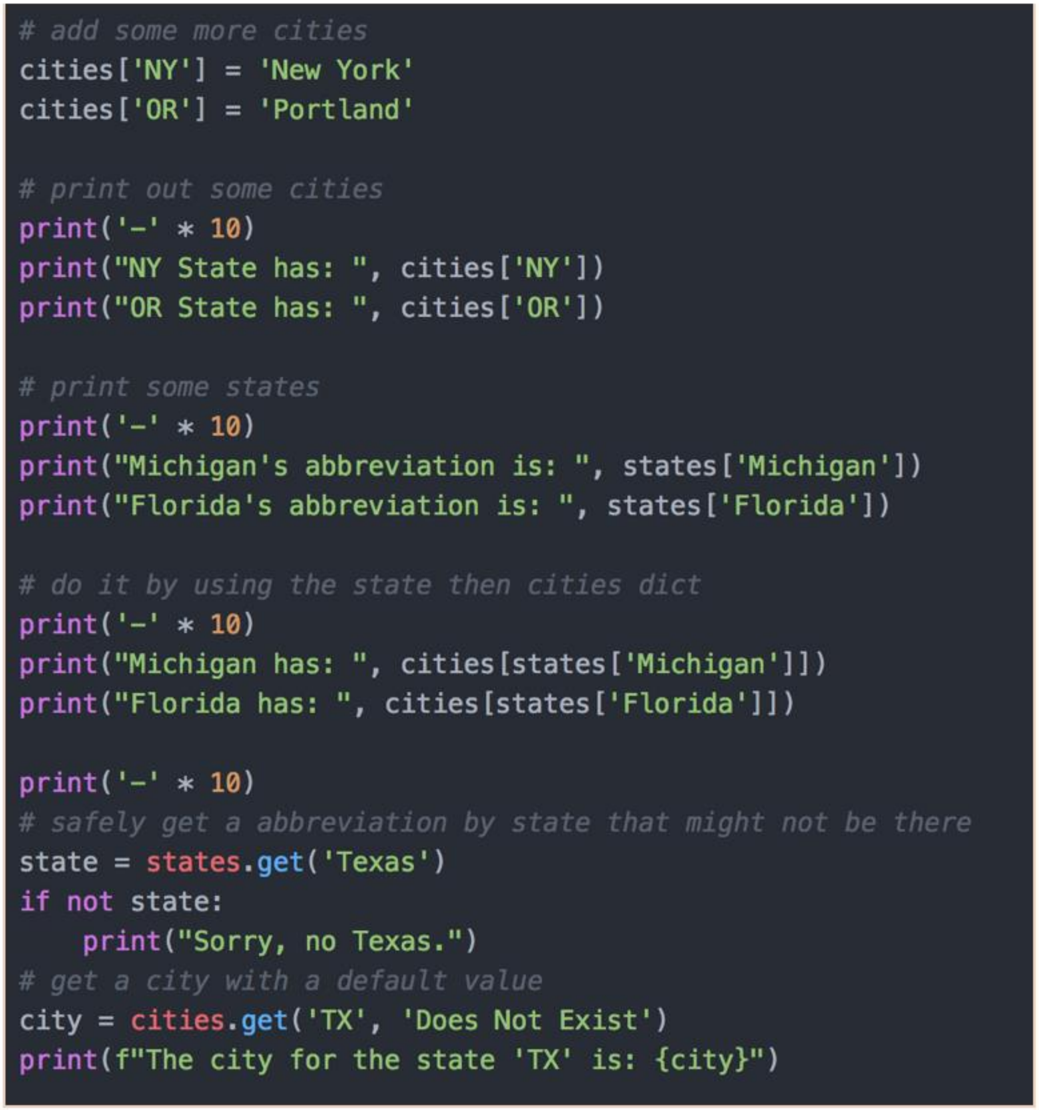


```
# create a mapping of state to abbreviation
states = {
   'Oregon': 'OR',
   'Florida': 'FL',
   'California': 'CA',
   'New York': 'NY',
   'Michigan': 'MI'
}

# create a basic set of states and some cities in them
cities = {
    'CA': 'San Francisco',
    'MI': 'Detroit',
    'FL': 'Jacksonville'
}
```

Figure 3.3 Creating dictionaries

Recap



```
# add some more cities
cities['NY'] = 'New York'
cities['OR'] = 'Portland'

# print out some cities
print('-' * 10)
print("NY State has: ", cities['NY'])
print("OR State has: ", cities['OR'])

# print some states
print('-' * 10)
print("Michigan's abbreviation is: ", states['Michigan'])
print("Florida's abbreviation is: ", states['Florida'])

# do it by using the state then cities dict
print('-' * 10)
print("Michigan has: ", cities[states['Michigan']])
print("Florida has: ", cities[states['Florida']])

print('-' * 10)
# safely get a abbreviation by state that might not be there
state = states.get('Texas')
if not state:
    print("Sorry, no Texas.")
# get a city with a default value
city = cities.get('TX', 'Does Not Exist')
print(f"The city for the state 'TX' is: {city}")
```

Figure 3.4 Adding items into dictionaries and two ways of accessing items in dictionaries

Quiz

Are dictionaries are mutable?

Given d = {'a': 1, 2: 'b'},

- What are the keys?
- what value does d[2] evaluate to?
- What value does d[‘b’] evaluate to?
- What value does d.get(2) evaluate to?
- What value does d.get(‘b’) evaluate to?
- What does d refer to after executing d[‘w’] = 3?
- What does d refer to when executing d[‘w’] = 1 after the above?

Discussion

Operations on dictionaries (https://docs.python.org/3/library/stdtypes.html#typesmapping)

If the variable b refers to {"one": 1, "two": 2, "three": 3, "four": 4},

– what does each of the following evaluate to?

  len(b)
  “one” in b
  4 in b
  b.items()
  b.keys()
  b.values()

– What does b refer to after executing **del b[“four”]**?

4. Sorting Dictionary

Recap

Sorting dictionary (SU3 Chapter 2, Textbook Video and Exercise 39)

Note:

A dictionary is an unordered set of key: value pairs, with the requirement that the keys are unique within one dictionary.

```
tel = {'jack': 4098, 'sape': 4139, 'irv': 4127}
```

Discussion

Sorting dictionary on keys or values (https://docs.python.org/3/library/collections.html#ordereddict-examples-and-recipes)

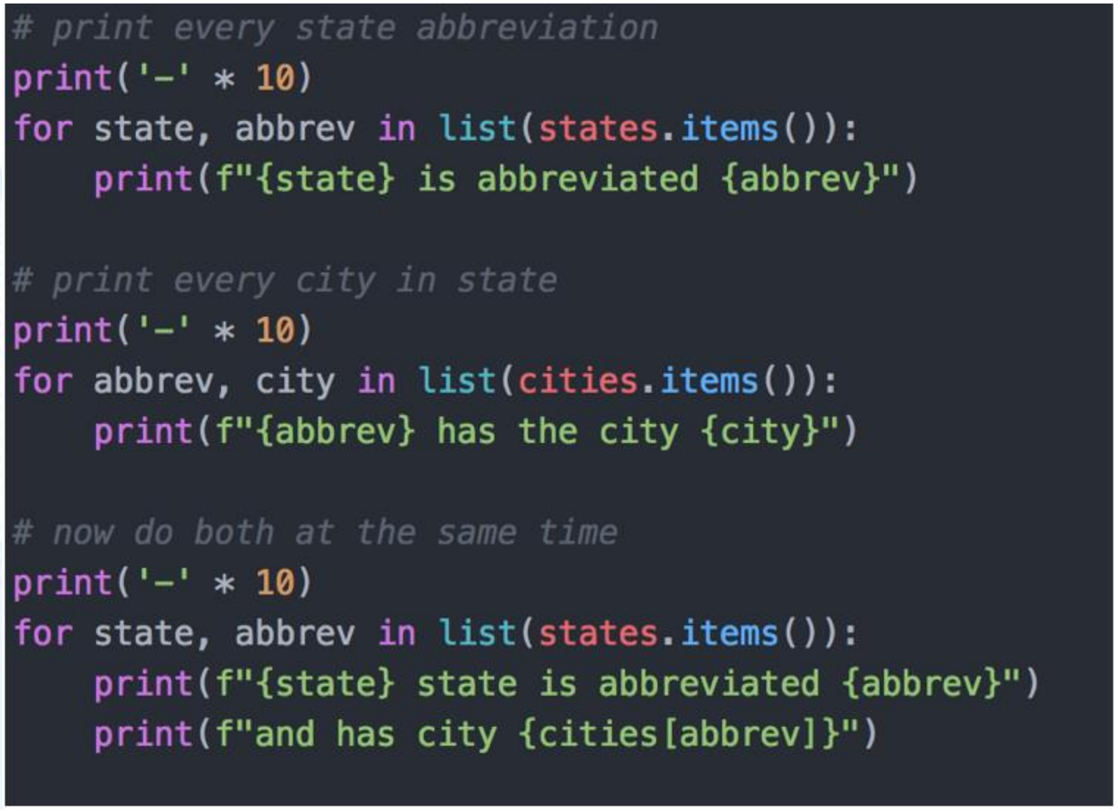
Given d = {'banana': 3, 'apple': 4, 'pear': 1, 'orange': 2}

- write code to generate a dictionary sorted by keys {'apple': 4, 'banana': 3, 'orange': 2, 'pear': 1}
- write code to generate a dictionary sorted by values {'pear': 1, 'orange': 2, 'banana': 3, 'apple': 4}

5. Loop over Dictionary

Recap

Loop over dictionary (SU3 Chapter 2, Textbook Video and Exercise 39, https://docs.python.org/3/library/stdtypes.html#typesmapping)



```python
# print every state abbreviation
print('-' * 10)
for state, abbrev in list(states.items()):
    print(f"{state} is abbreviated {abbrev}")

# print every city in state
print('-' * 10)
for abbrev, city in list(cities.items()):
    print(f"{abbrev} has the city {city}")

# now do both at the same time
print('-' * 10)
for state, abbrev in list(states.items()):
    print(f"{state} state is abbreviated {abbrev}")
    print(f"and has city {cities[abbrev]}")
```

Figure 3.5 Iterating over items in dictionaries using for-loop

Quiz

If the variable dishes refers to {'eggs': 2, 'sausage': 1, 'bacon': 1, 'spam': 500},

- write a program to count the total number of all the dishes.
- write a program to increase each value by 1.

Discussion

Given color_to_fruit = {'orange': 'orange', 'purple': 'plum', 'green': 'pear', 'yellow': 'banana', 'red': 'pomegranate'}, write a program to add a new fruit (e.g. green watermelon), but not to replace the existing one (e.g. pear).

Discussion

Given id_to_grade = {’1389': 55.0, ’1377': 85.0, ’1311': 77.5, ’1078': 62.5, ’0941': 55.0, ‘0052’: 77.5}, write a program to generate a dictionary where each key is a grade and each value is the list of ids of students who earned that grade.

Thank
you