DICTIONARIES

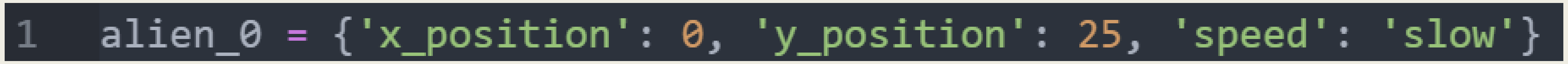
Mapping

Dictionaries (1 of 2)

Ex1_alien.py & Ex2_user.py & Ex3_favorite_languages.py

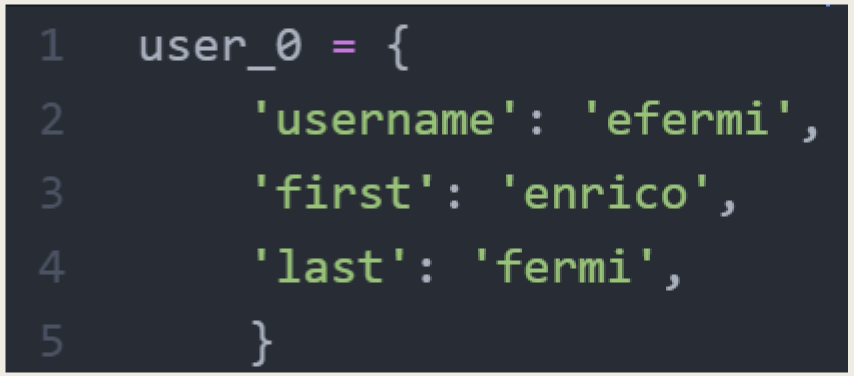
- Python has an efficient and flexible mapping device, called a **dictionary**

  The value of alien_0[“speed”] is “slow”

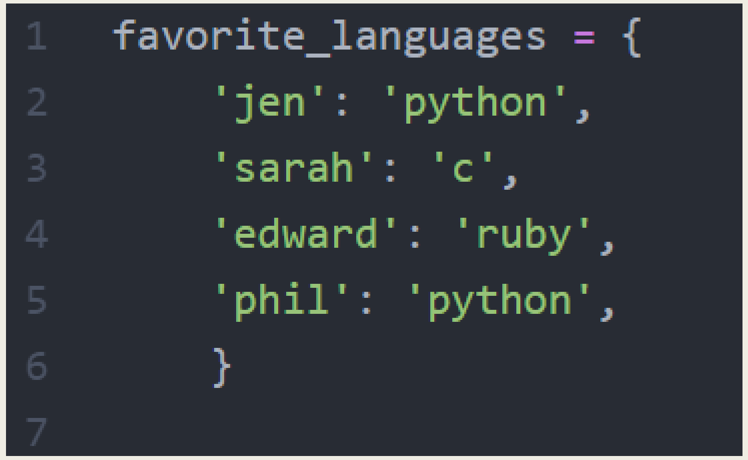


```python
alien_0 = {'x_position': 0, 'y_position': 25, 'speed': 'slow'}
```

The value of user_0[“username”] is “efermi”



```python
user_0 = {
    'username': 'efermi',
    'first': 'enrico',
    'last': 'fermi',
    }
```



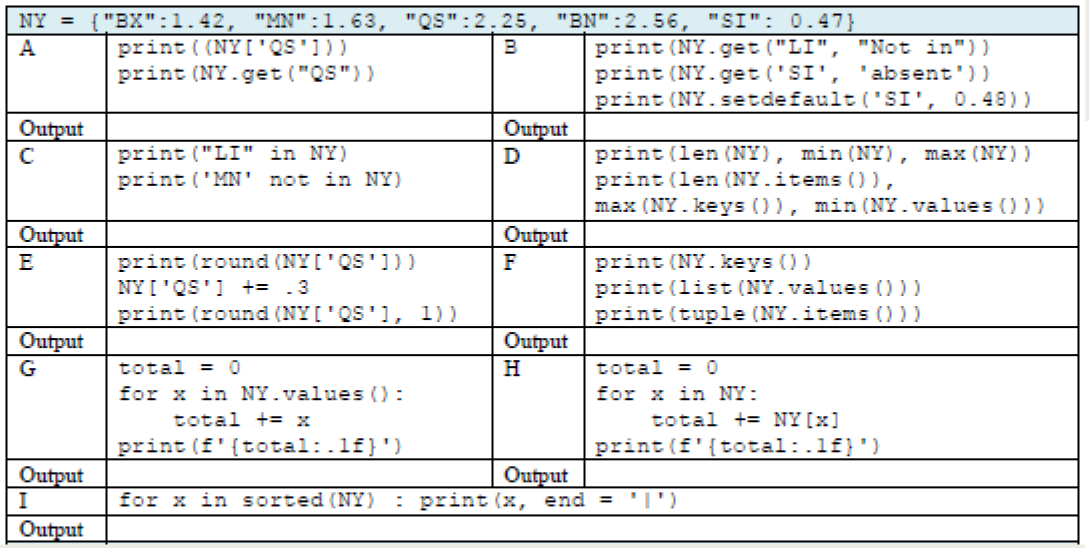
```python
favorite_languages = {
    'jen': 'python',
    'sarah': 'c',
    'edward': 'ruby',
    'phil': 'python',
    }

```

- The value of favorite_languages[“jen"] is “python"

Dictionaries (2 of 2)

- Python dictionary is defined as
  - *A collection of comma separated pairs*
  - *Of the form "key:value"*
  - *Enclosed in curly braces*
- The **key** must be **immutable** objects (such as strings, numbers, or tuples)
- The **keys** are unique
- The **value** associated with *key1* is given by the expression `dictionaryName[key1]`
- The **value** can have any data types, and the values needn't be unique

PE8_1



`NY = {"BX":1.42, "MN":1.63, "QS":2.25, "BN":2.56, "SI": 0.47}`			

A	`print((NY['QS']))`
	`print(NY.get("QS"))`
		B	`print(NY.get("LI", "Not in"))`
			`print(NY.get('SI', 'absent'))`
			`print(NY.setdefault('SI', 0.48))`

Output		Output	

C	`print("LI" in NY)`
	`print('MN' not in NY)`
		D	`print(len(NY), min(NY), max(NY))`
			`print(len(NY.items()),`
			`max(NY.keys()), min(NY.values()))`

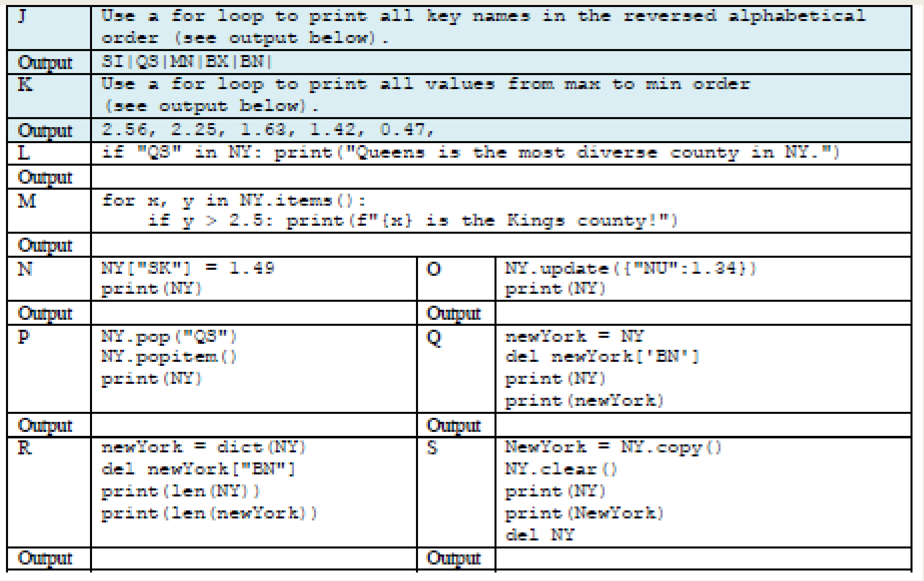
Output		Output	

E	`print(round(NY['QS']))`
	`NY['QS'] += .3`
	`print(round(NY['QS'], 1))`
		F	`print(NY.keys())`
			`print(list(NY.values()))`
			`print(tuple(NY.items()))`

Output		Output	

G	`total = 0`
	`for x in NY.values():`
	`    total += x`
	`print(f'{total:.1f}')`
		H	`total = 0`
			`for x in NY:`
			`    total += NY[x]`
			`print(f'{total:.1f}')`

Output		Output	
I	`for x in sorted(NY) : print(x, end = '|')`		
Output			



J	Use a for loop to print all key names in the reversed alphabetical order (see output below).		
Output	`SI|QS|MN|BX|BN|`		
K	Use a for loop to print all values from max to min order (see output below).		
Output	`2.56, 2.25, 1.63, 1.42, 0.47,`		
L	`if "QS" in NY: print("Queens is the most diverse county in NY.")`		
Output			

M	`for x, y in NY.items():`
	`    if y > 2.5: print(f"{x} is the Kings county!")`		

Output			

N	`NY["SK"] = 1.49`
	`print(NY)`
		O	`NY.update({"NU":1.34})`
			`print(NY)`

Output		Output	

P	`NY.pop("QS")`
	`NY.popitem()`
	`print(NY)`
		Q	`newYork = NY`
			`del newYork['BN']`
			`print(NY)`
			`print(newYork)`

Output		Output	

R	`newYork = dict(NY)`
	`del newYork["BN"]`
	`print(len(NY))`
	`print(len(newYork))`
		S	`NewYork = NY.copy()`
			`NY.clear()`
			`print(NY)`
			`print(NewYork)`
			`del NY`

Output		Output	

Dictionary Operations (1 of 4)

PE8_1.py

Operation	Description
c = {}	Creates an empty dictionary
dict()	Creates a dictionary
c = dict(d)	Use the built-in function to creates a copy of the dictionary d
c = d.copy()	Use the method to create a copy of the dictionary d
len(d)	Number of items (that is, key:value pairs) in the dictionary
min(d)	Smallest value of d.keys(), provided all keys have the same data type
max(d)	Largest value of d.keys, provided all keys have the same data type
x in d	Has value True if x is a key of the dictionary; otherwise, has value False
x not in d	Has value True if x is not a key of the dictionary; otherwise, has value False

Dictionary Operations (2 of 4)

PE8_1.py

Operation	Description
d[key1]	Returns the value associated with key1. Raises an error if key 1 is not a key of d
d[key1] = value1	If key1 is already a key in the dictionary, changes the value associated with key1 to value1; otherwise, adds the item key1:value1 to the dictionary
d.get(key1, default)	If key1 is not a key of the dictionary, returns the default value; otherwise, returns the value associated with key1
d.setdefault(key1, value1)	Requires 1-argument, returns the value of the item with the specified key. If the key does not exist, insert the key, with the specified value. If the value is not specified, returns None
d.update(c)	Merges all dictionary c's entries into dictionary d; if two items have the same keys, the value from c replaces the value from d

Dictionary Operations (3 of 4)

PE8_1.py

Operation	Description
d.pop(key1)	Requires 1-argument and removes the item with the specified key name
d.popitem()	Requires no argument and removes the last inserted item
d.clear()	Removes all items (that is, key:value pairs) from the dictionary
del d[key1]	Removes the item having key1 as key; raises an exception if key1 is not found

Dictionary Operations (4 of 4)

PE8_1.py

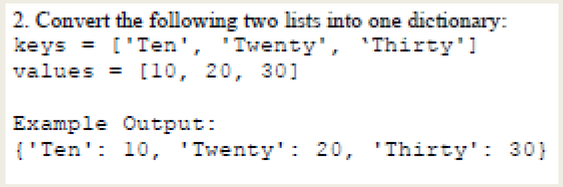
Operation	Description
tuple(d)	Returns a tuple of the keys in the dictionary
list(d)	Returns a list of the keys in the dictionary
list(d.keys())	Returns a list of the keys in the dictionary
list(d.values())	Returns a list of the values in the dictionary
list(d.items())	Returns a list of two-tuples of the form (key, value)where d(key) = value
set(d)	Returns a set of the keys in the dictionary
zip(k, v)	Returns an iterator, from two or more iterators
for k in d:	Iterates over all the keys in the dictionary

- More Dictionary Methods

The *dict()* Function

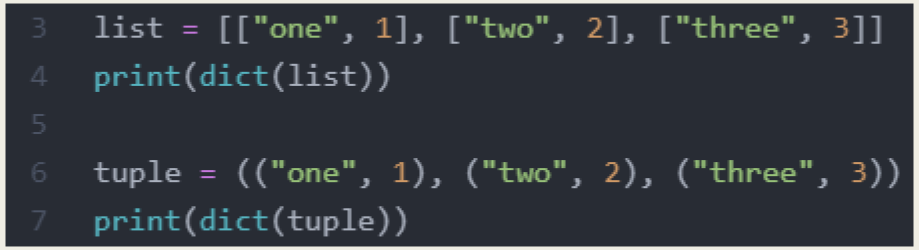
PE8_2.py

- 2-item **lists** or 2-item **tuples** can be converted to a **dictionary** with *dict()* function
- Example

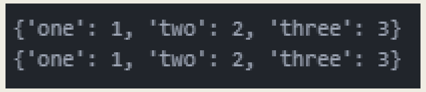


2. Convert the following two lists into one dictionary:

```
keys = ['Ten', 'Twenty', 'Thirty']
values = [10, 20, 30]

Example Output:
{'Ten': 10, 'Twenty': 20, 'Thirty': 30}
```



```python
list = [["one", 1], ["two", 2], ["three", 3]]
print(dict(list))

tuple = (("one", 1), ("two", 2), ("three", 3))
print(dict(tuple))
```

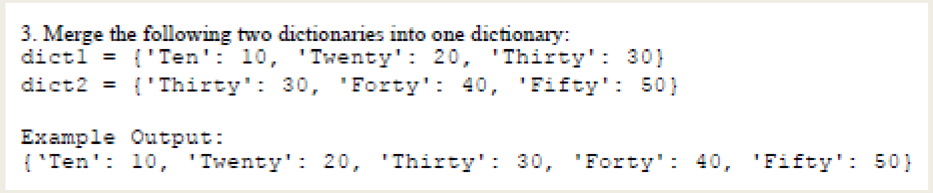


```
{'one': 1, 'two': 2, 'three': 3}
{'one': 1, 'two': 2, 'three': 3}
```

The *zip()* Function

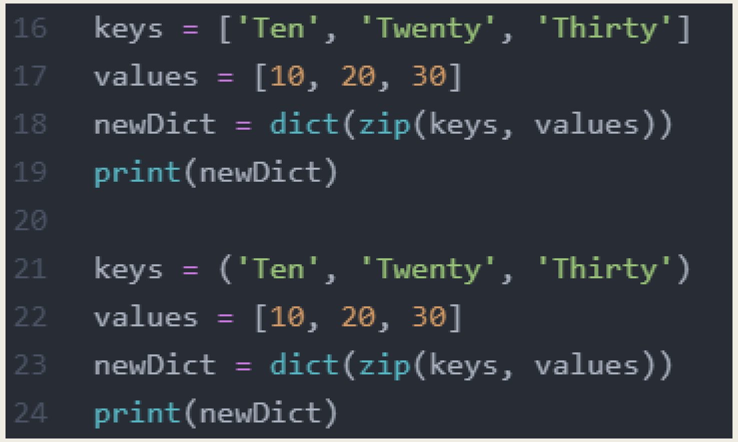
PE8_3.py

- 2 lists or 2 tuples or a list and a tuple can be converted to a dictionary with dict() and zip() functions
- Example

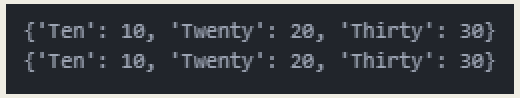


3. Merge the following two dictionaries into one dictionary:

```
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
dict2 = {'Thirty': 30, 'Forty': 40, 'Fifty': 50}

Example Output:
{'Ten': 10, 'Twenty': 20, 'Thirty': 30, 'Forty': 40, 'Fifty': 50}
```



```python
keys = ['Ten', 'Twenty', 'Thirty']
values = [10, 20, 30]
newDict = dict(zip(keys, values))
print(newDict)

keys = ('Ten', 'Twenty', 'Thirty')
values = [10, 20, 30]
newDict = dict(zip(keys, values))
print(newDict)
```



```
{'Ten': 10, 'Twenty': 20, 'Thirty': 30}
{'Ten': 10, 'Twenty': 20, 'Thirty': 30}
```

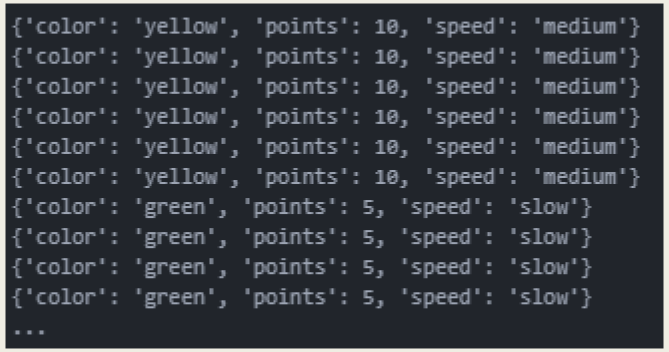
PE8_4 & PE8_5

- Write your codes and run

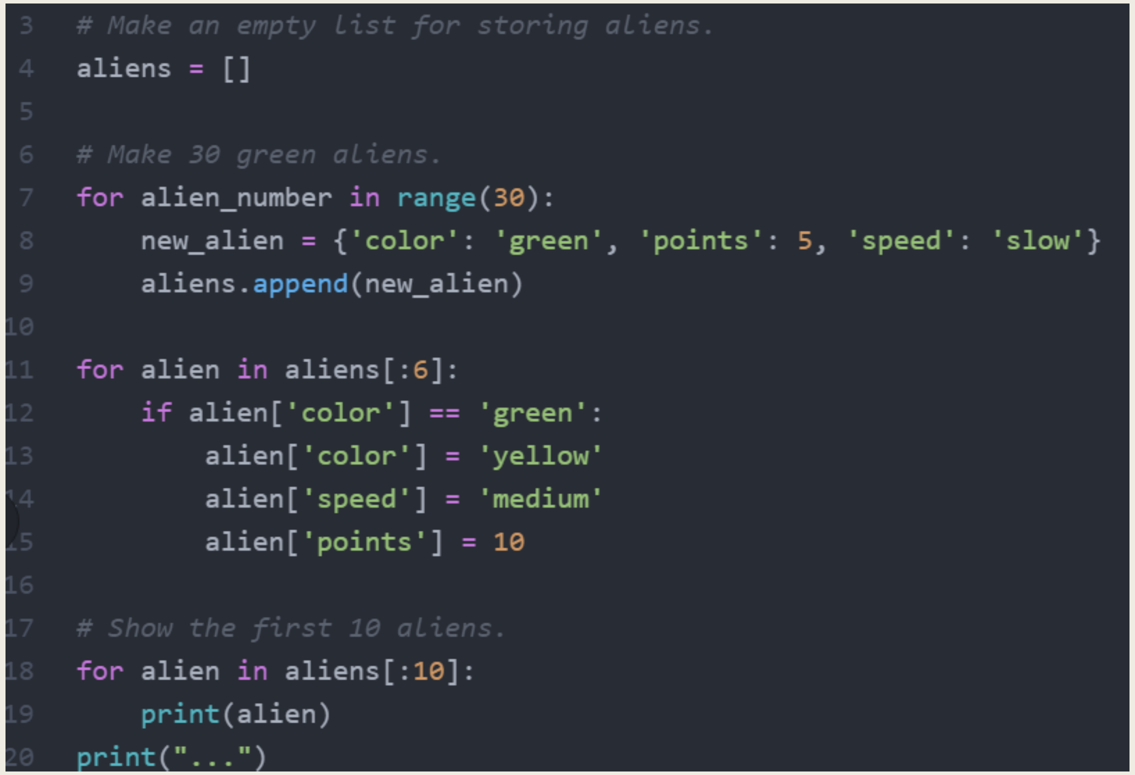
Nested Dictionaries (1 of 4)

Ex4_aliens.py

- A set of dictionaries inside a list
- Example



```
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
...
```

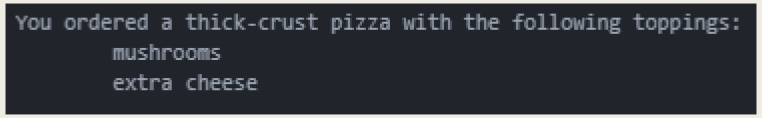


```python
# Make an empty list for storing aliens.
aliens = []

# Make 30 green aliens.
for alien_number in range(30):
    new_alien = {'color': 'green', 'points': 5, 'speed': 'slow'}
    aliens.append(new_alien)

for alien in aliens[:6]:
    if alien['color'] == 'green':
        alien['color'] = 'yellow'
        alien['speed'] = 'medium'
        alien['points'] = 10

# Show the first 10 aliens.
for alien in aliens[:10]:
    print(alien)
print("...")
```

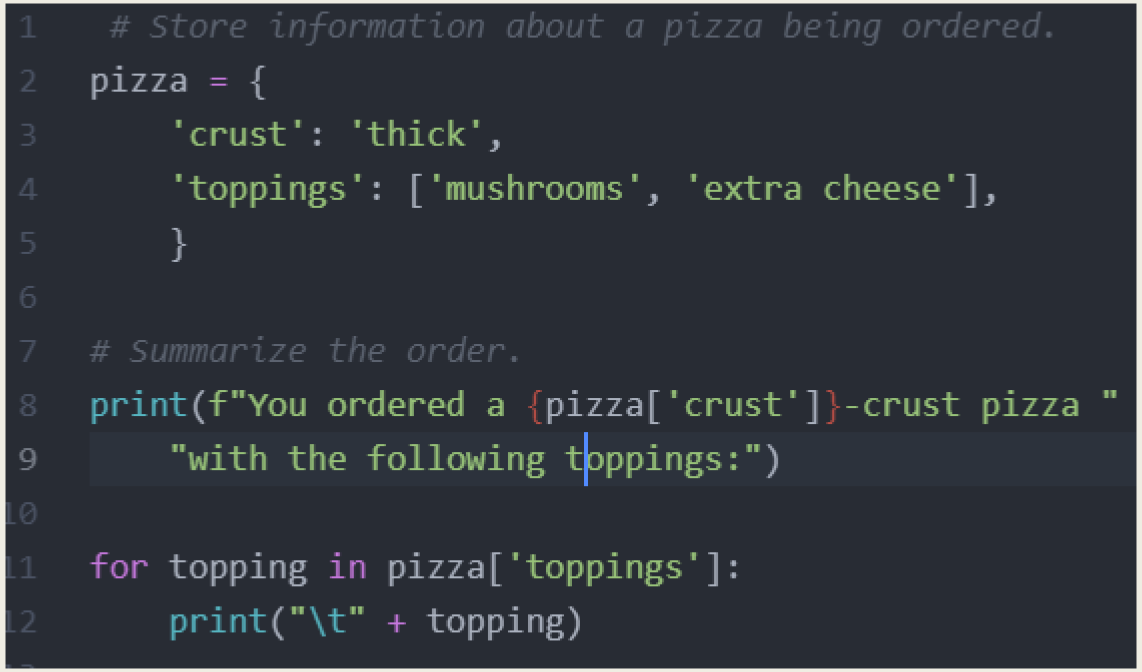
Nested Dictionaries (2 of 4)

Ex5_pizza.py

- A list of items as a value in a dictionary
- Example



```
You ordered a thick-crust pizza with the following toppings:
	mushrooms
	extra cheese
```

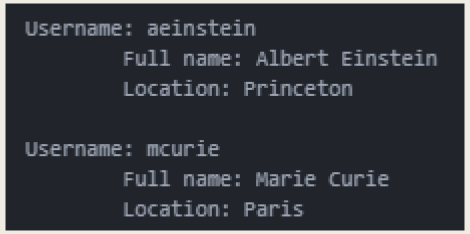


```python
 # Store information about a pizza being ordered.
pizza = {
    'crust': 'thick',
    'toppings': ['mushrooms', 'extra cheese'],
    }

# Summarize the order.
print(f"You ordered a {pizza['crust']}-crust pizza "
    "with the following toppings:")

for topping in pizza['toppings']:
    print("\t" + topping)
```

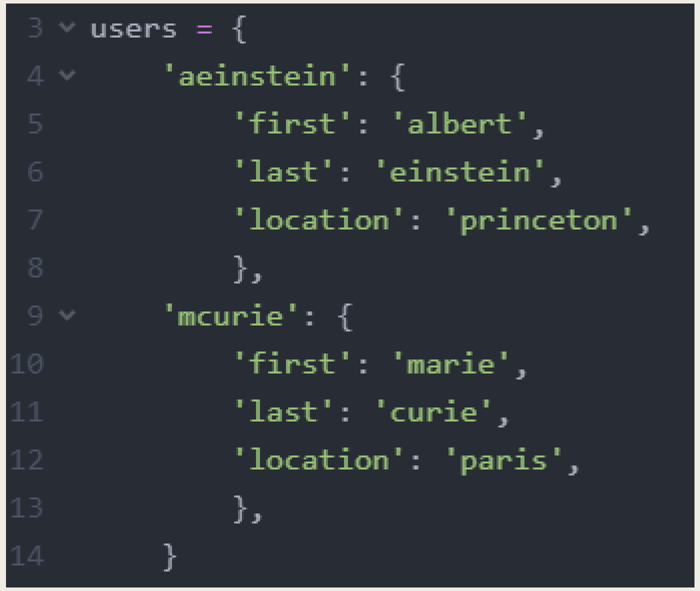
Nested Dictionaries (3 of 4)

Ex6_many_users.py

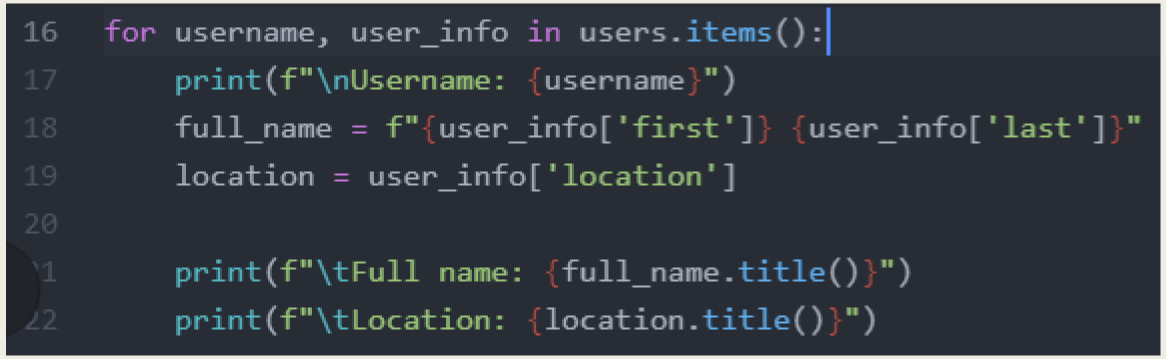
- A dictionary inside another dictionaries
- Example



```
Username: aeinstein
        Full name: Albert Einstein
        Location: Princeton

Username: mcurie
        Full name: Marie Curie
        Location: Paris
```



```python
users = {
    'aeinstein': {
        'first': 'albert',
        'last': 'einstein',
        'location': 'princeton',
        },
    'mcurie': {
        'first': 'marie',
        'last': 'curie',
        'location': 'paris',
        },
    }
```

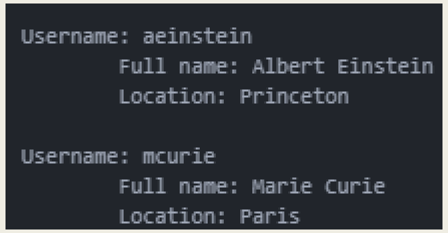


```python
for username, user_info in users.items():
    print(f"\nUsername: {username}")
    full_name = f"{user_info['first']} {user_info['last']}"
    location = user_info['location']

    print(f"\tFull name: {full_name.title()}")
    print(f"\tLocation: {location.title()}")
```

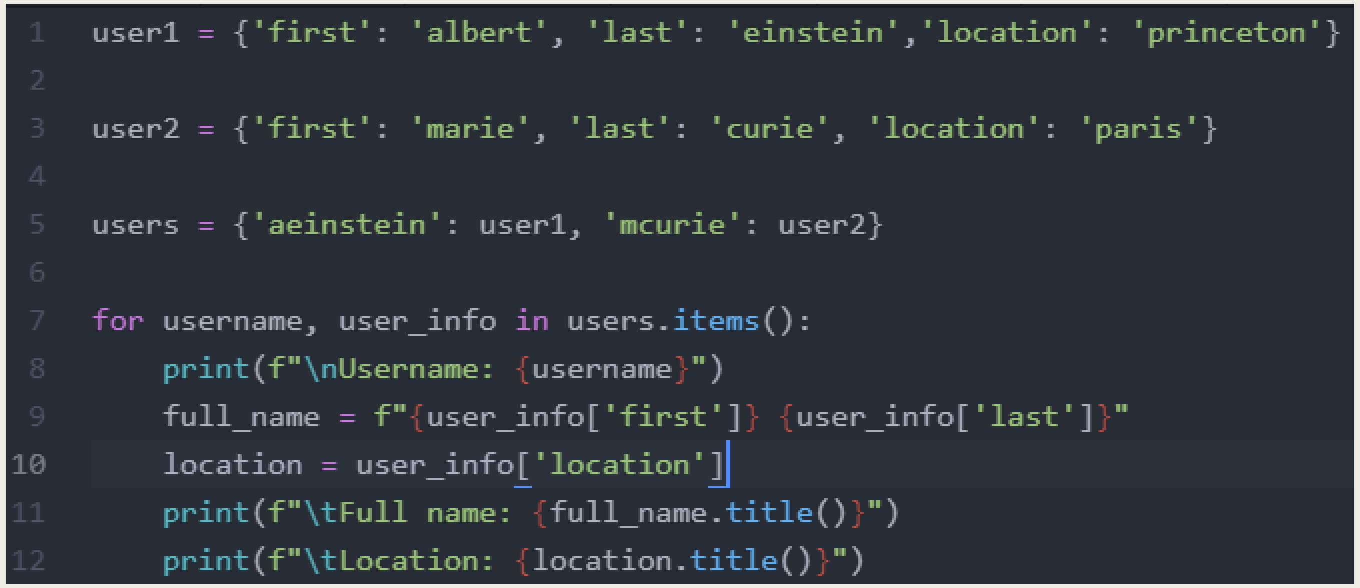
Nested Dictionaries (4 of 4)

Ex7_nest_users.py

- A dictionary inside another dictionaries
- Example



```
Username: aeinstein
        Full name: Albert Einstein
        Location: Princeton

Username: mcurie
        Full name: Marie Curie
        Location: Paris
```



```python
user1 = {'first': 'albert', 'last': 'einstein','location': 'princeton'}

user2 = {'first': 'marie', 'last': 'curie', 'location': 'paris'}

users = {'aeinstein': user1, 'mcurie': user2}

for username, user_info in users.items():
    print(f"\nUsername: {username}")
    full_name = f"{user_info['first']} {user_info['last']}"
    location = user_info['location']
    print(f"\tFull name: {full_name.title()}")
    print(f"\tLocation: {location.title()}")
```

PE8_6 & PE8_7

- Write your codes and run

Summary (1 of 2)

- A dictionary is an unordered collection of *key:value* pairs that **map** each key into its value.
- One way to create a dictionary is to place its *key:value* pairs (separated by commas) inside curly braces.
- `dictionaryName[key]` returns the value associated with the key.
- Dictionary keys must be immutable object. Numbers, strings, and tuples (but not lists) can serve as **keys** and all types of Python objects can serve as **values**.

```
d = { ("Blue", "Green"): "Cyan"}  valid
d = { ["Blue", "Green"]: "Cyan"}  invalid
```

Summary (2 of 2)

- Dictionary operations:
  `len, max, min, dict, list, tuple, set, zip`
  `in, not in, del,`
  `get, keys, values, items, copy, clear, pop, popitem,`
  `update, setdefault`

Dictionaries Terminologies

- ❑ 1 Curly brackets
- ❑ 2 Dictionary
- ❑ 3 Keys
- ❑ 4 Key-Value pairs
- ❑ 5 Mapping
- ❑ 6 Nested dictionary
- ❑ 7 Values
- ❑ 8 Unordered
- ❑ 9 Changeable

- ❑ 10 clear()
- ❑ 11 copy()
- ❑ 12 del
- ❑ 13 dict()
- ❑ 14 get()
- ❑ 15 items()
- ❑ 16 keys()
- ❑ 17 pop()
- ❑ 18 popitem()

- ❑ 19 set()
- ❑ 20 setdefault()
- ❑ 21 values()
- ❑ 22 update()
- ❑ 23 zip()

Quiz 8

- Quiz 8A has 10 questions in 15 minutes, 10 pts
  - *10 multiple choice/true or false questions, 1 pt. for each question*
  - *Quiz 8A has two attempt, the higher grade will be selected*
  - *Submit Quiz 8A (at least 1-minute)* ***before*** *the due time to Blackboard*

- Quiz 8B has 2 code questions, 15 pts
  - *Write the Python code based on the given question*
  - *Each question will be given during the last 10-minute of each session of week 8*
  - *Quiz 8B-1 on session A, and Quiz 8B-2 on session B*
  - *Quiz 8B has one attempt*

DB 8

- Instruction:
  1) Choose any **three** of the questions from PE8_1. Please avoid selecting the exact same questions. Make sure to indicate the **question #** you're working on in the thread title as soon as you open your thread. Then you can **explain and edit your questions** (1.2 pt).
  2) Indicate one **mistake** you have made and/or share one **tip** when you work on the lists and dictionaries (0.3 pt).
  3) Submit your posts before the due date. Let's learn from each other.