Dictionaries

CLARUSWAY©
WAY TO REINVENT YOURSELF

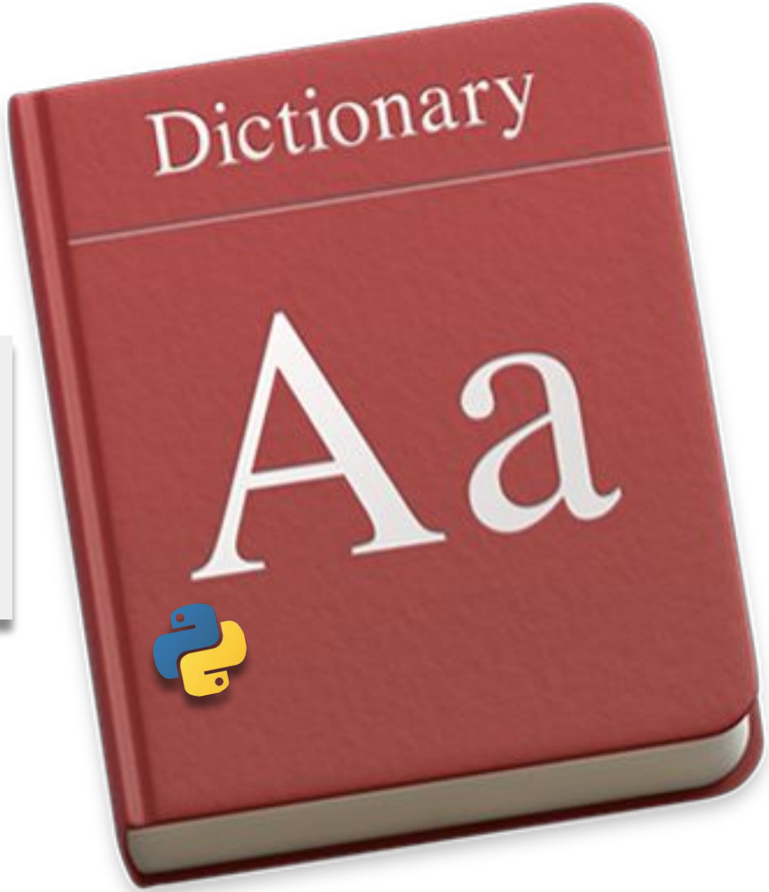
Table of Contents

CLARUSWAY©
WAY TO REINVENT YOURSELF

Definitions

- Dictionaries

```
{key1 : value1,
 key2 : value2}
```



Creating a `dict`

CLARUSWAY©
WAY TO REINVENT YOURSELF

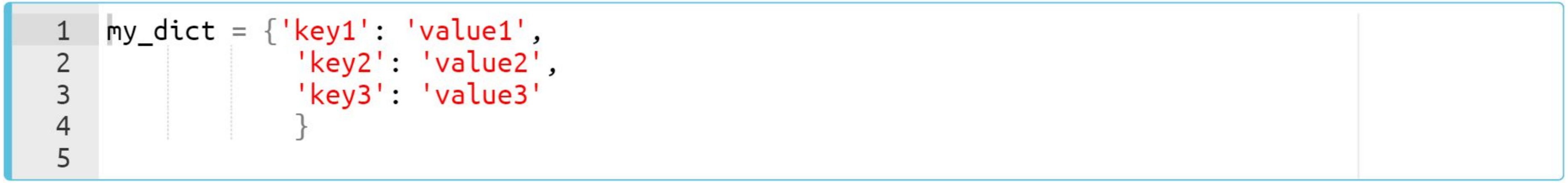
Creating a `dict` (review)

- We have two basic ways to create a dictionary.

- `{}`
- `dict()`

Creating a `dict`(review pre-class)

- Here is an example of simple structure of a `dict`:



```
my_dict = {'key1': 'value1',
           'key2': 'value2',
           'key3': 'value3'
          }
```

Creating a `dict` (review)

- A `dict` can be created by enclosing pairs, separated by commas, in curly-braces 👉 `{}`.
- Another way to create a `dict` is to call the `dict()` function.

```
grocer1 = {'fruit':'apple', 'drink':'water'}
grocer2 = dict(fruit='apple', drink='water')
print(grocer1)
print(grocer2)
```

- `{}`
- `dict()`

What is the output? Try to figure out in your mind...

CLARUSWAY©
WAY TO REINVENT YOURSELF

Creating a `dict` (review)

- A `dict` can be created by enclosing pairs, separated by commas, in curly-braces 👉 `{}`.
- Another way to create a `dict` is to call the `dict()` function.

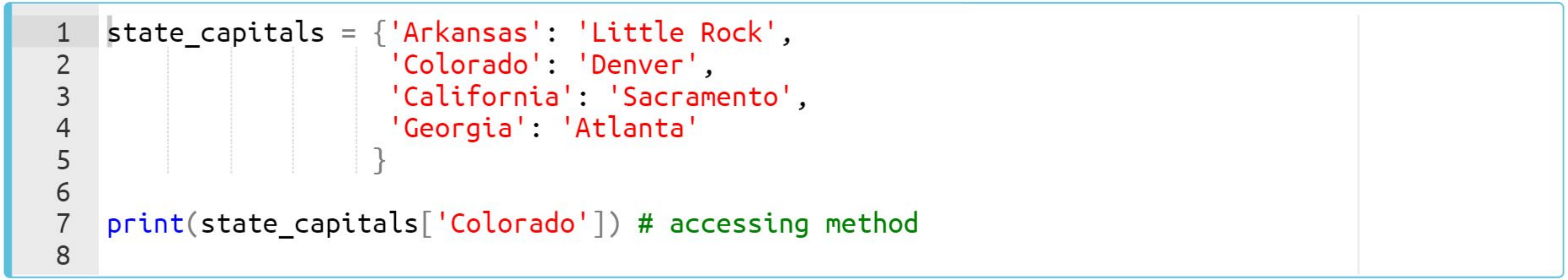
```
grocer1 = {'fruit':'apple', 'drink':'water'}
grocer2 = dict(fruit='apple', drink='water')
print(grocer1)
print(grocer2)
```

```
{'fruit': 'apple', 'drink': 'water'}
{'fruit': 'apple', 'drink': 'water'}
```

- `{}`
- `dict()`

Creating a `dict`(review pre-class)

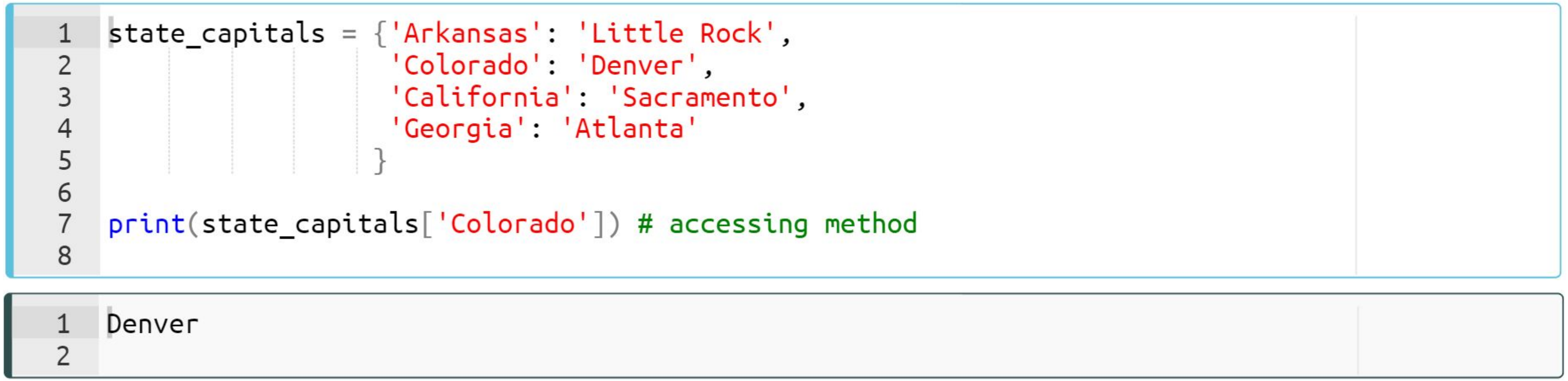
- Accessing and assigning an item.



```python
state_capitals = {'Arkansas': 'Little Rock',
                  'Colorado': 'Denver',
                  'California': 'Sacramento',
                  'Georgia': 'Atlanta'
                 }

print(state_capitals['Colorado']) # accessing method

```

Creating a `dict`(review pre-class)

- Assigning a value to a key

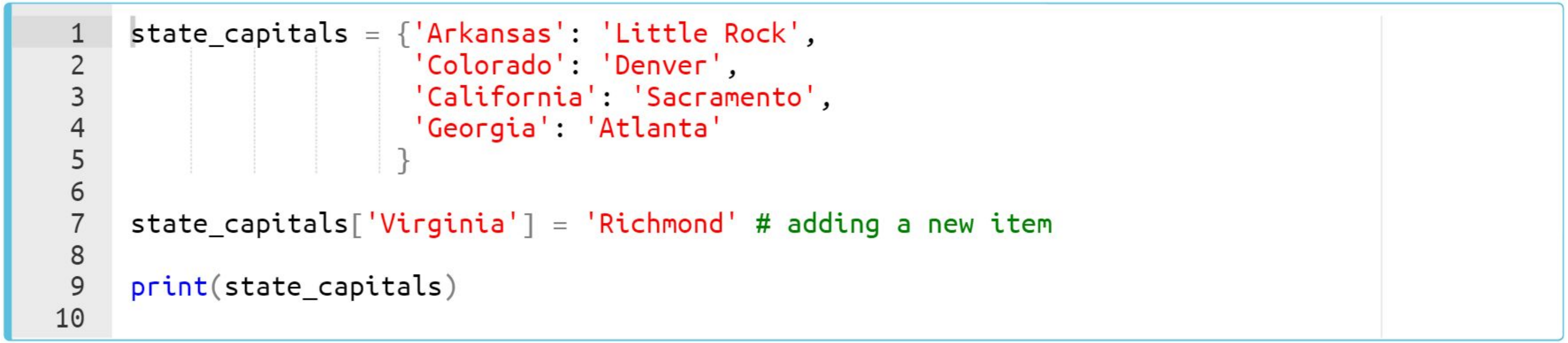


```
state_capitals = {'Arkansas': 'Little Rock',
                  'Colorado': 'Denver',
                  'California': 'Sacramento',
                  'Georgia': 'Atlanta'
                 }

print(state_capitals['Colorado']) # accessing method

```

```
Denver

```

Creating a `dict`(review pre-class)

- Let's add a new item into the `dict`.



```
state_capitals = {'Arkansas': 'Little Rock',
                  'Colorado': 'Denver',
                  'California': 'Sacramento',
                  'Georgia': 'Atlanta'
                 }

state_capitals['Virginia'] = 'Richmond' # adding a new item

print(state_capitals)
```

Creating a `dict`(review pre-class)

- Let’s add a new item into the `dict`.

```python
state_capitals = {'Arkansas': 'Little Rock',
                  'Colorado': 'Denver',
                  'California': 'Sacramento',
                  'Georgia': 'Atlanta'
                 }

state_capitals['Virginia'] = 'Richmond' # adding a new item

print(state_capitals)

```

```
{'Arkansas': 'Little Rock',
'Colorado': 'Denver',
'California': 'Sacramento',
'Georgia': 'Atlanta',
'Virginia': 'Richmond'}

```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Creating a `dict`(review pre-class)

Tips:

- Note that keys and values can be of different types.

```python
mix_values = {'animal': ('dog', 'cat'),  # tuple type
              'planet': ['Neptun', 'Saturn', 'Jupiter'],  # list type
              'number': 40,  # int type
              'pi': 3.14,  # float type
              'is_good': True}  # bool type

mix_keys = {22 : "integer",
            1.2 : "float",
            True : "boolean",
            "key" : "string"}
```

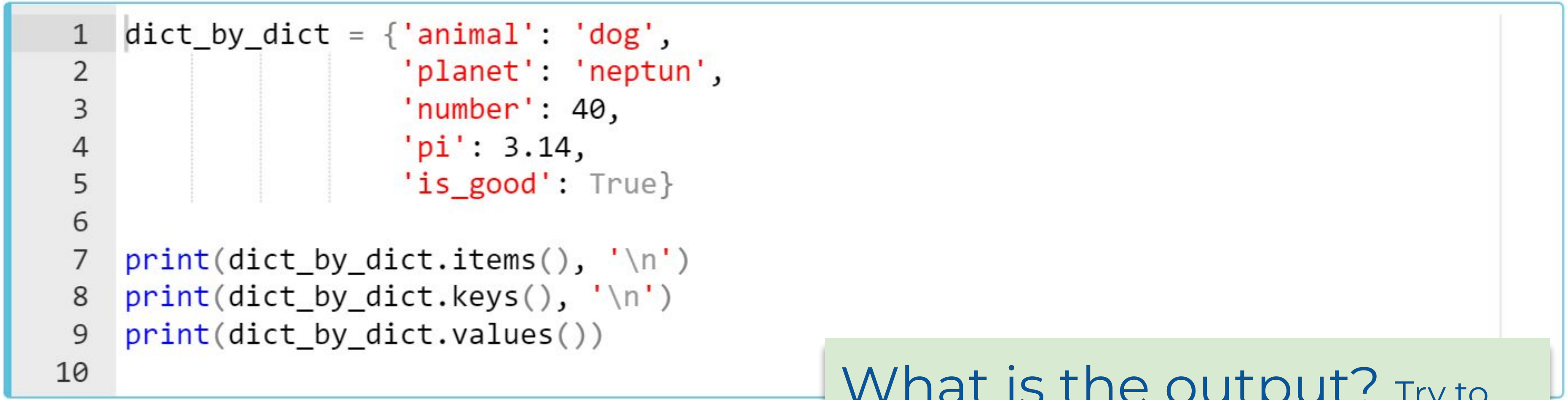
CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with Dictionaries

CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with `dicts` (review)

- Let’s take a look at this example :

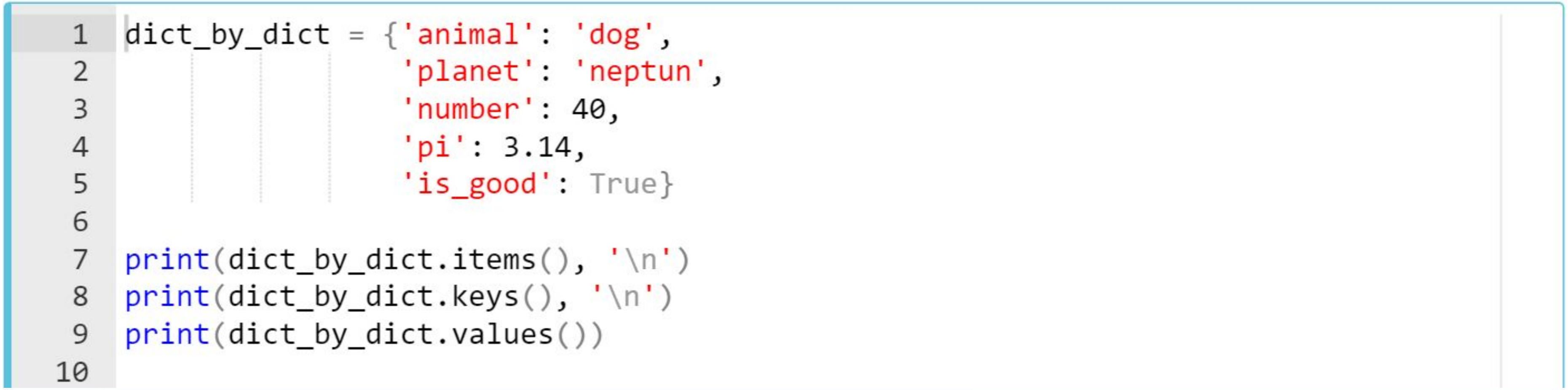


```python
dict_by_dict = {'animal': 'dog',
                'planet': 'neptun',
                'number': 40,
                'pi': 3.14,
                'is_good': True}

print(dict_by_dict.items(), '\n')
print(dict_by_dict.keys(), '\n')
print(dict_by_dict.values())
```

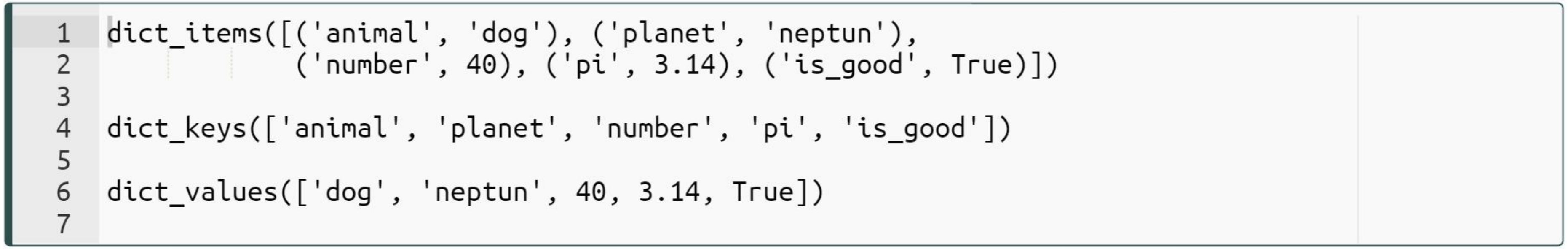
What is the output? Try to figure out in your mind...

Main Operations with `dicts` (review)

- Let's take a look at this example :



```python
dict_by_dict = {'animal': 'dog',
                'planet': 'neptun',
                'number': 40,
                'pi': 3.14,
                'is_good': True}

print(dict_by_dict.items(), '\n')
print(dict_by_dict.keys(), '\n')
print(dict_by_dict.values())
```

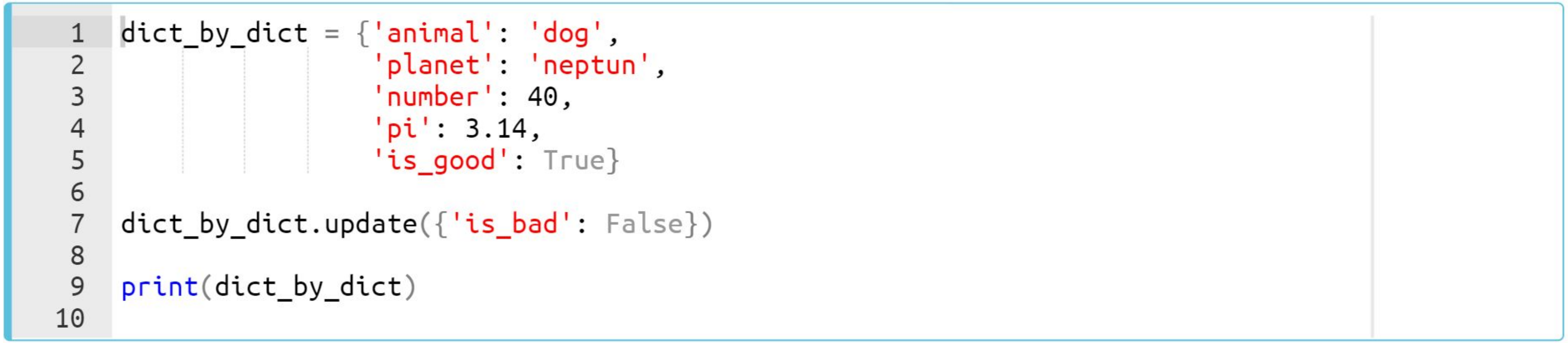


```
dict_items([('animal', 'dog'), ('planet', 'neptun'),
            ('number', 40), ('pi', 3.14), ('is_good', True)])

dict_keys(['animal', 'planet', 'number', 'pi', 'is_good'])

dict_values(['dog', 'neptun', 40, 3.14, True])
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with `dicts` (review)

- `.update()` method :



```python
dict_by_dict = {'animal': 'dog',
                'planet': 'neptun',
                'number': 40,
                'pi': 3.14,
                'is_good': True}

dict_by_dict.update({'is_bad': False})

print(dict_by_dict)
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with `dicts` (review)

- Another way to add a new **item** into a `dict` is the `.update()` method.

```python
dict_by_dict = {'animal': 'dog',
                'planet': 'neptun',
                'number': 40,
                'pi': 3.14,
                'is_good': True}

dict_by_dict.update({'is_bad': False})

print(dict_by_dict)
```

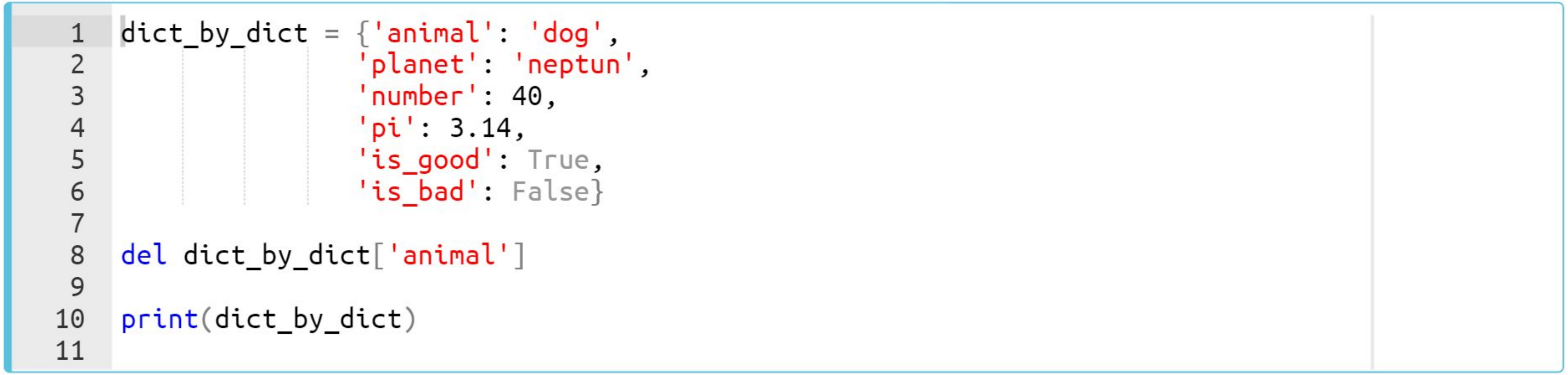
```
{'animal': 'dog',
'planet': 'neptun',
'number': 40,
'pi': 3.14,
'is_good': True,
'is_bad': False}
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with dicts (review)

- Python allows us to remove an item from a dict using the del function.

The formula syntax is : `del dictionary_name['key']`

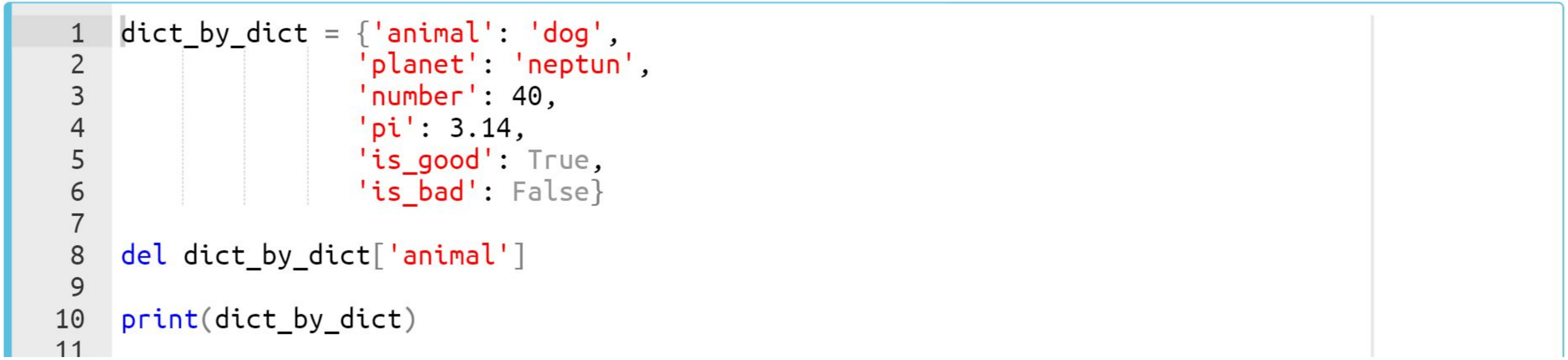


```
dict_by_dict = {'animal': 'dog',
                'planet': 'neptun',
                'number': 40,
                'pi': 3.14,
                'is_good': True,
                'is_bad': False}

del dict_by_dict['animal']

print(dict_by_dict)
```

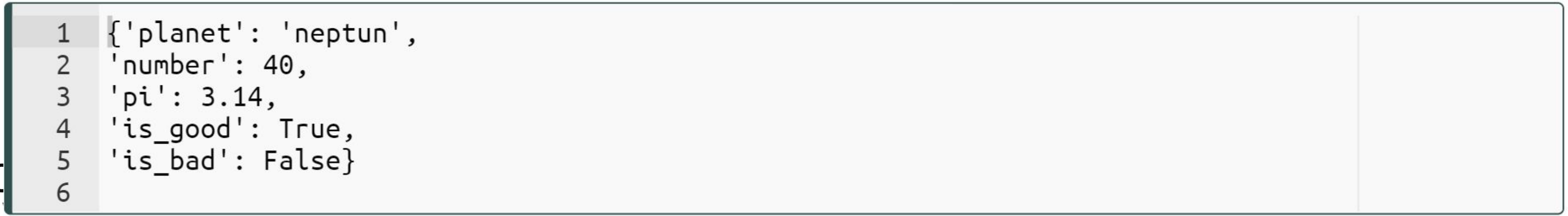
Main Operations with `dicts` (review)

- Python allows us to remove an item from a `dict` using the `del` function.

The formula syntax is : `del dictionary_name['key']`



```python
dict_by_dict = {'animal': 'dog',
                'planet': 'neptun',
                'number': 40,
                'pi': 3.14,
                'is_good': True,
                'is_bad': False}

del dict_by_dict['animal']

print(dict_by_dict)
```



```
{'planet': 'neptun',
'number': 40,
'pi': 3.14,
'is_good': True,
'is_bad': False}
```

Main Operations with `dicts` (review)

Using the `in` and the `not in` operator, you can check if the `key` is in the `dict`ionary.

> - When we use the `in` operator; if the `key` is in the dictionary, the result will be `True` otherwise `False`.
>
> - When we use the `not in`; if the `key` is not in the dictionary, the result will be `True` otherwise `False`.

CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with `dicts` (review)

Using the `in` and the `not in` operator, you can check if the `key` is in the `dict`ionary.

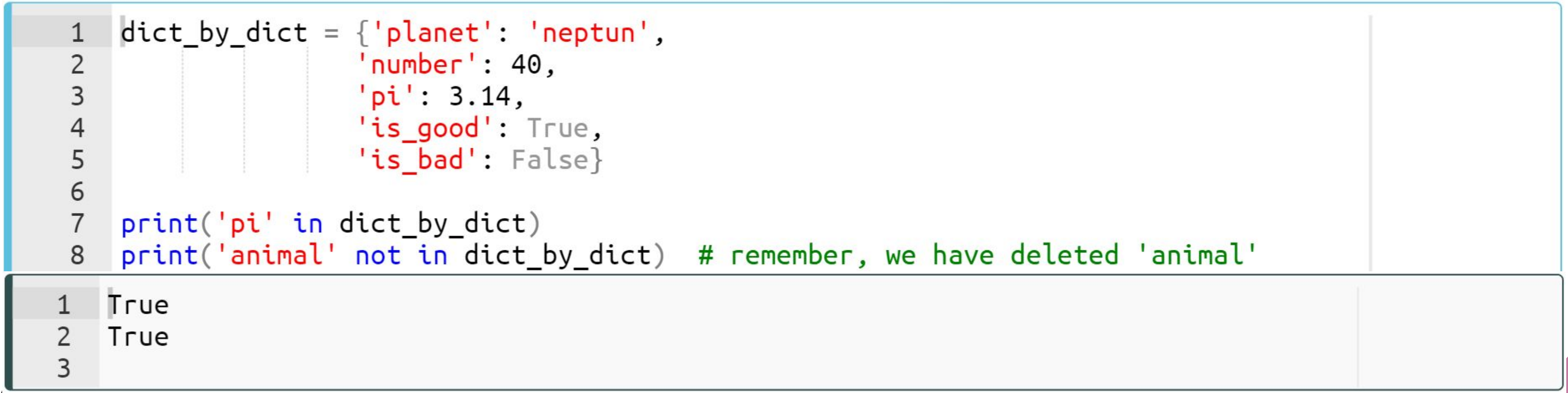
- When we use the `in` operator; if the `key` is in the dictionary, the result will be `True` otherwise `False`.
- When we use the `not in`; if the `key` is not in the dictionary, the result will be `True` otherwise `False`.

```python
dict_by_dict = {'planet': 'neptun',
                'number': 40,
                'pi': 3.14,
                'is_good': True,
                'is_bad': False}

print('pi' in dict_by_dict)
print('animal' not in dict_by_dict)  # remember, we have deleted 'animal'

```

Main Operations with `dicts` (review)

Using the `in` and the `not in` operator, you can check if the `key` is in the `dict`ionary.

- When we use the `in` operator; if the `key` is in the dictionary, the result will be `True` otherwise `False`.
- When we use the `not in`; if the `key` is not in the dictionary, the result will be `True` otherwise `False`.



```python
dict_by_dict = {'planet': 'neptun',
                'number': 40,
                'pi': 3.14,
                'is_good': True,
                'is_bad': False}

print('pi' in dict_by_dict)
print('animal' not in dict_by_dict)  # remember, we have deleted 'animal'
```

```
True
True
```

Other Operations with `dicts`

- `clear()`; **Remove** all items from the dictionary.
- `pop(key[, default])`; If *key* is in the dictionary, **remove** it and **return its value**, else return *default*. If *default* is not given and *key* is not in the dictionary, a KeyError is raised.
- `popitem()`; **Remove** and **return a** `(key, value)` **pair** from the dictionary. Pairs are returned in **LIFO** order.
- `copy()`; Return a shallow **copy** of the dictionary.
- `get(key[, default])`; **Return the value** for *key* if *key* is in the dictionary, else *default*. If *default* is not given, it defaults to `None`, so that this method never raises a `KeyError`.

Main Operations with `dicts`

- pop()

```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha',
          'name4': 'Tom'
         }
# using pop to return and remove key-value pair.
pop_ele = family.pop('name1')
print("deleted..:", pop_ele)
print(family)
```

Option-3

```
deleted..: Joseph
{'name2': 'Bella', 'name3': 'Aisha', 'name4': 'Tom'}
```

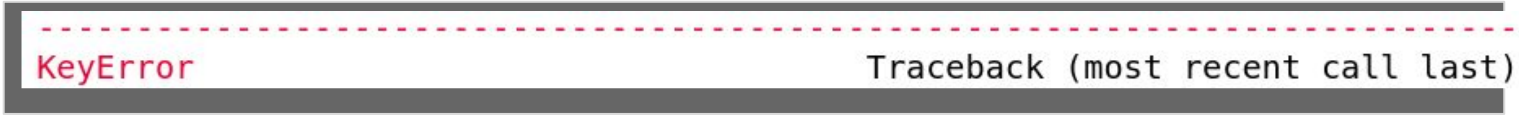
CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with `dicts`

- **If the key is not present in the dictionary, it raises a KeyError.**

KeyError

```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha',
          'name4': 'Tom'
          }
>>> family.pop('name5')
## or
>>> del family['name5']
```



```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with `dicts`

- If the key is **not present** in the dictionary, it raises a **KeyError.**

KeyError Solution?

```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha',
          'name4': 'Tom'
         }
>>> family.pop('name5', 'absent in the dict.')
```

message

```
'absent in the dict.'
```

Main Operations with `dicts`

`popitem`(): Remove and return a `(key, value)` pair from the dictionary. Pairs are returned in **LIFO** order.

```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha',
          'name4': 'Tom'
         }
print(family.popitem() )
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Main Operations with `dicts`

Remove and return a `(key, value)` pair from the dictionary. Pairs are returned in **LIFO** order.

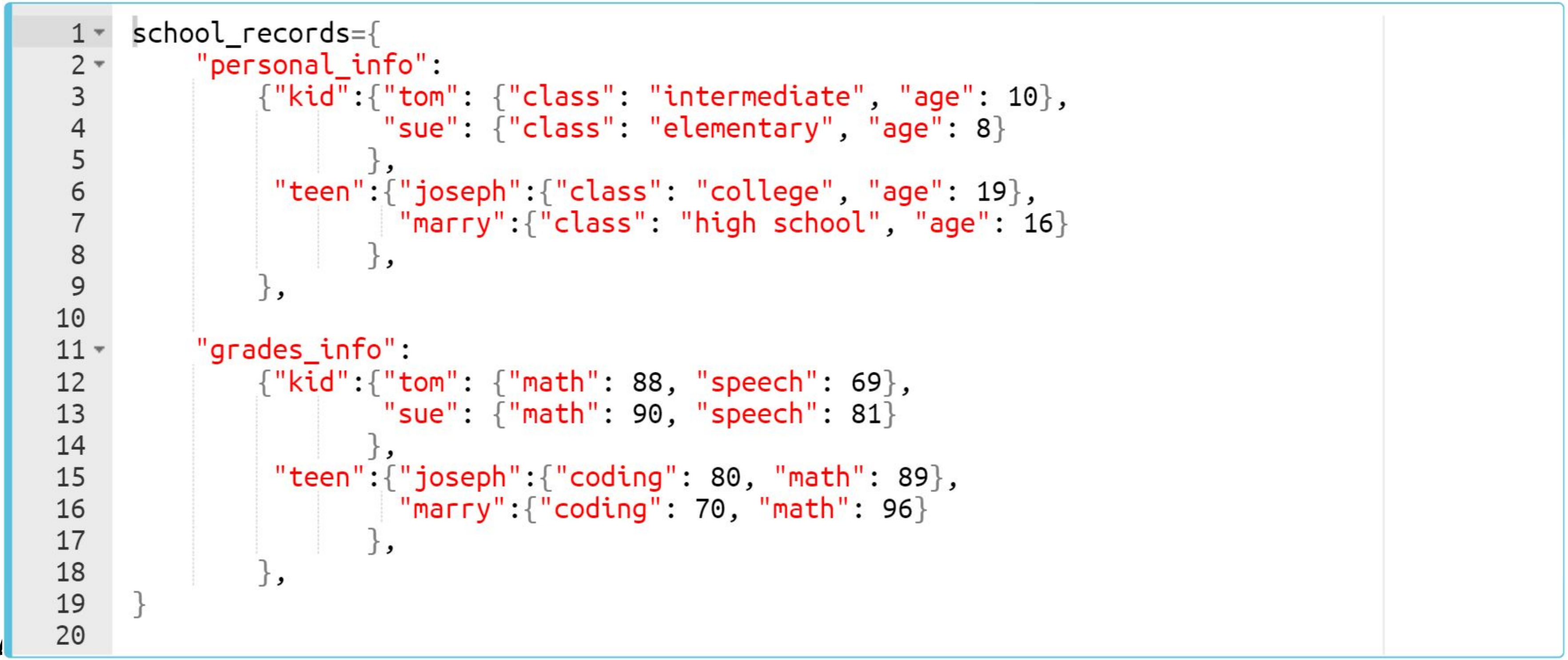
```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha',
          'name4': 'Tom'
         }
print(family.popitem() )
```

```
('name4', 'Tom')
```

Nested Dictionaries

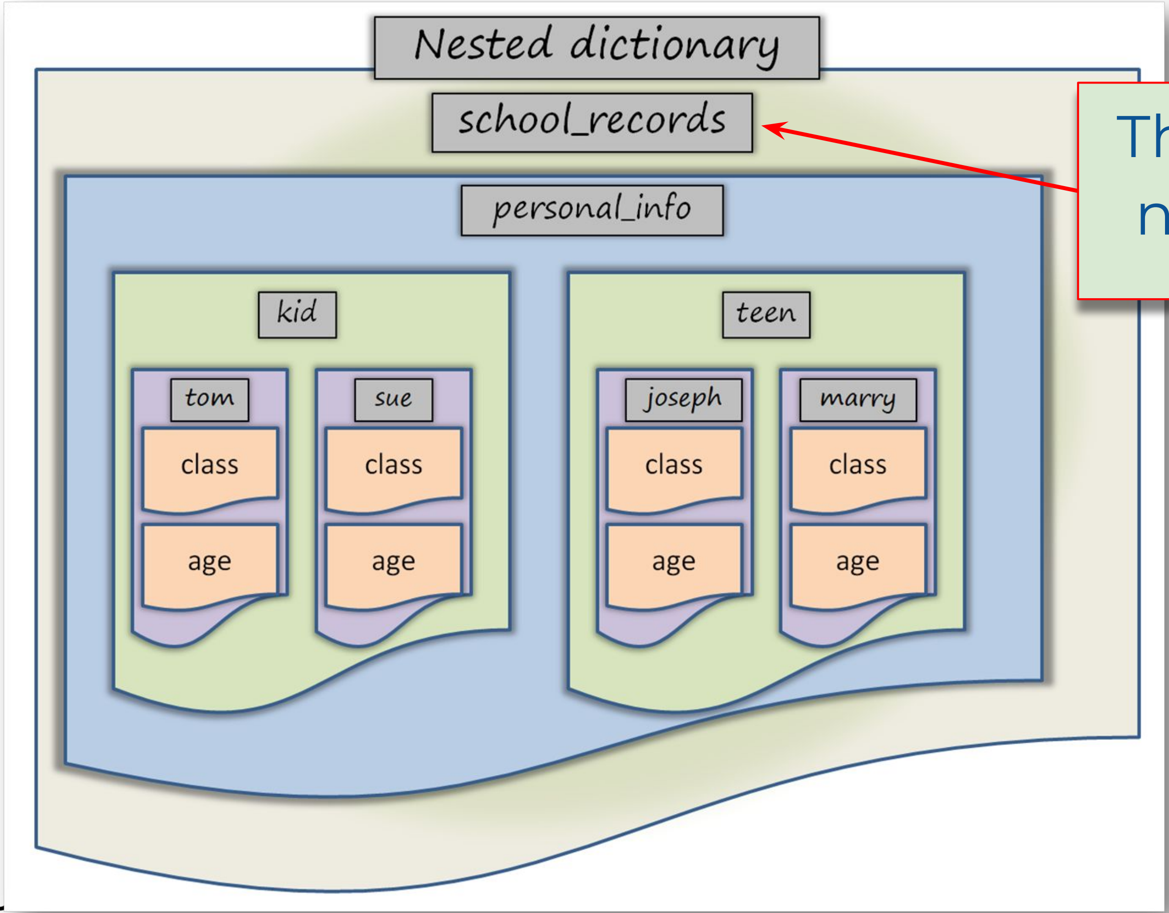
Nested `dicts` (review pre-class)

- In some cases you need to work with nested `dict`. Consider the following pre-class example :



```
school_records={
        "personal_info":
                {"kid":{"tom": {"class": "intermediate", "age": 10},
                        "sue": {"class": "elementary", "age": 8}
                       },
                 "teen":{"joseph":{"class": "college", "age": 19},
                         "marry":{"class": "high school", "age": 16}
                        },
                },

        "grades_info":
                {"kid":{"tom": {"math": 88, "speech": 69},
                        "sue": {"math": 90, "speech": 81}
                       },
                 "teen":{"joseph":{"coding": 80, "math": 89},
                         "marry":{"coding": 70, "math": 96}
                        },
                },
}
```

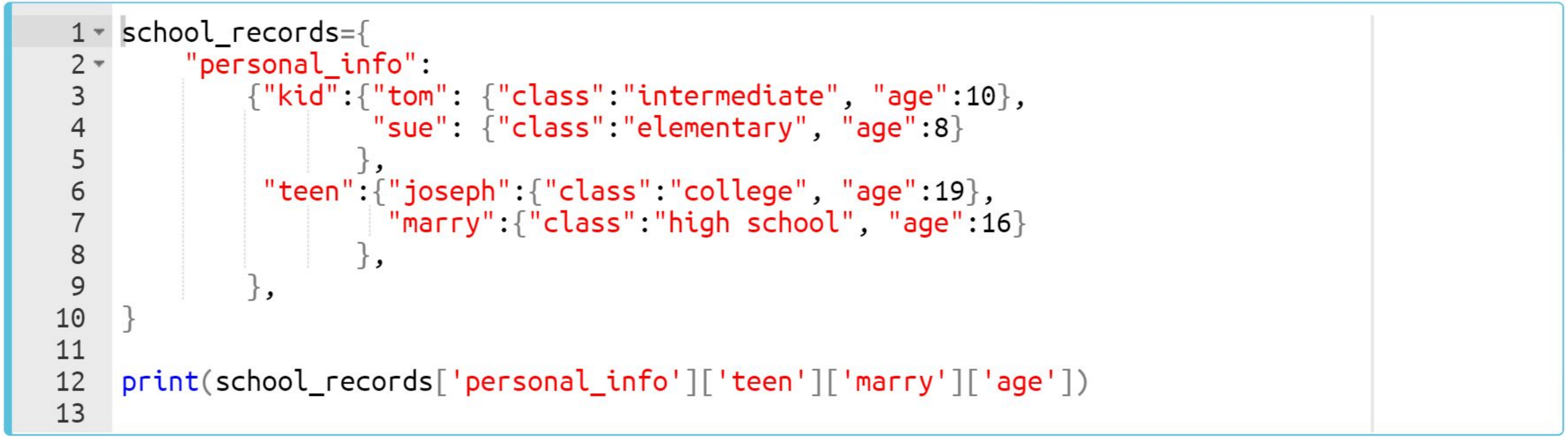
Nested `dicts` (review pre-class)



The first part of the nested dictionary.

Nested `dicts` (review pre-class)

- You can use traditional accessing method - square brackets - also in the nested dictionaries.



```
school_records={
    "personal_info":
        {"kid":{"tom": {"class":"intermediate", "age":10},
                "sue": {"class":"elementary", "age":8}
               },
         "teen":{"joseph":{"class":"college", "age":19},
                 "marry":{"class":"high school", "age":16}
                },
        },
}

print(school_records['personal_info']['teen']['marry']['age'])

```

Nested `dicts` (review pre-class)

- You can use traditional accessing method **-** square brackets **-** also in the nested dictionaries.

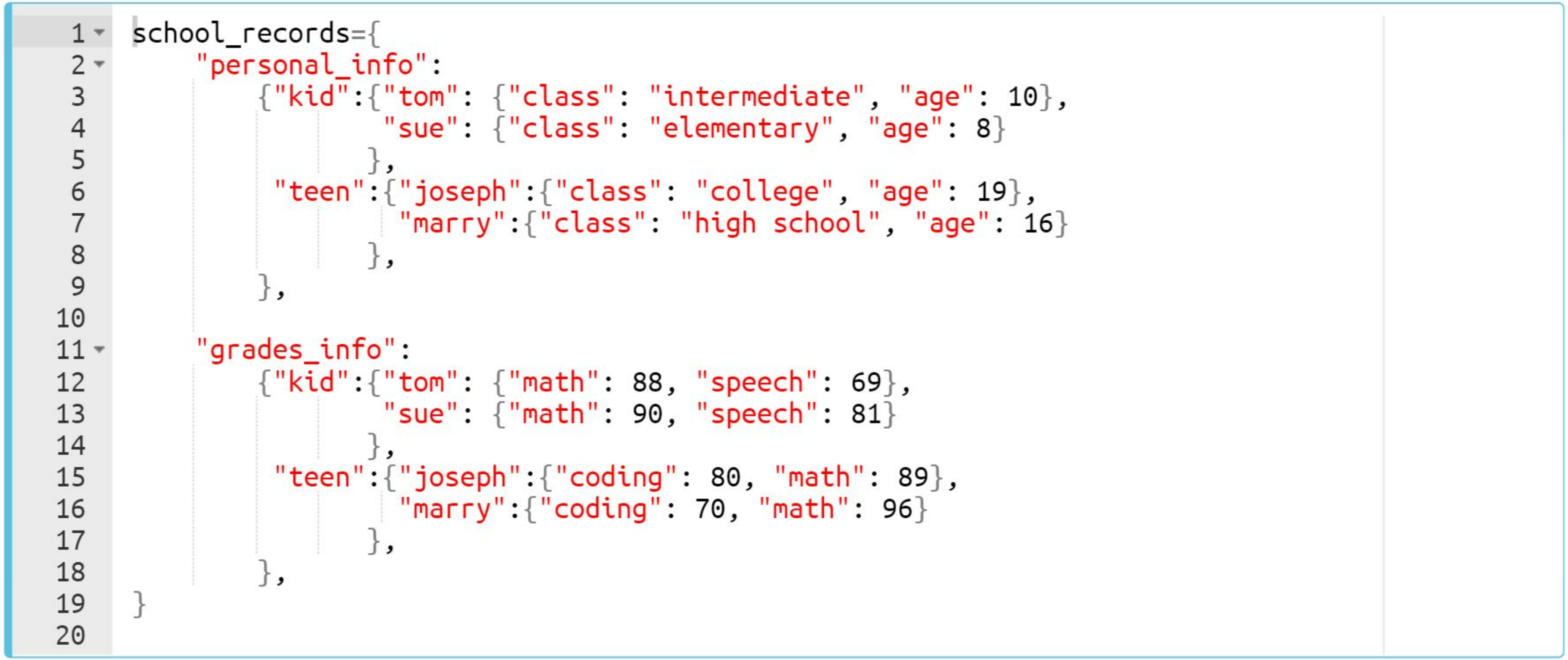
```
school_records={
        "personal_info":
                {"kid":{"tom": {"class":"intermediate", "age":10},
                        "sue": {"class":"elementary", "age":8}
                       },
                  "teen":{"joseph":{"class":"college", "age":19},
                          "marry":{"class":"high school", "age":16}
                        },
                },
}

print(school_records['personal_info']['teen']['marry']['age'])
```

```
16
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Nested `dicts`

- **Task** : Access and print the **exams** and their **grades** of **Joseph**

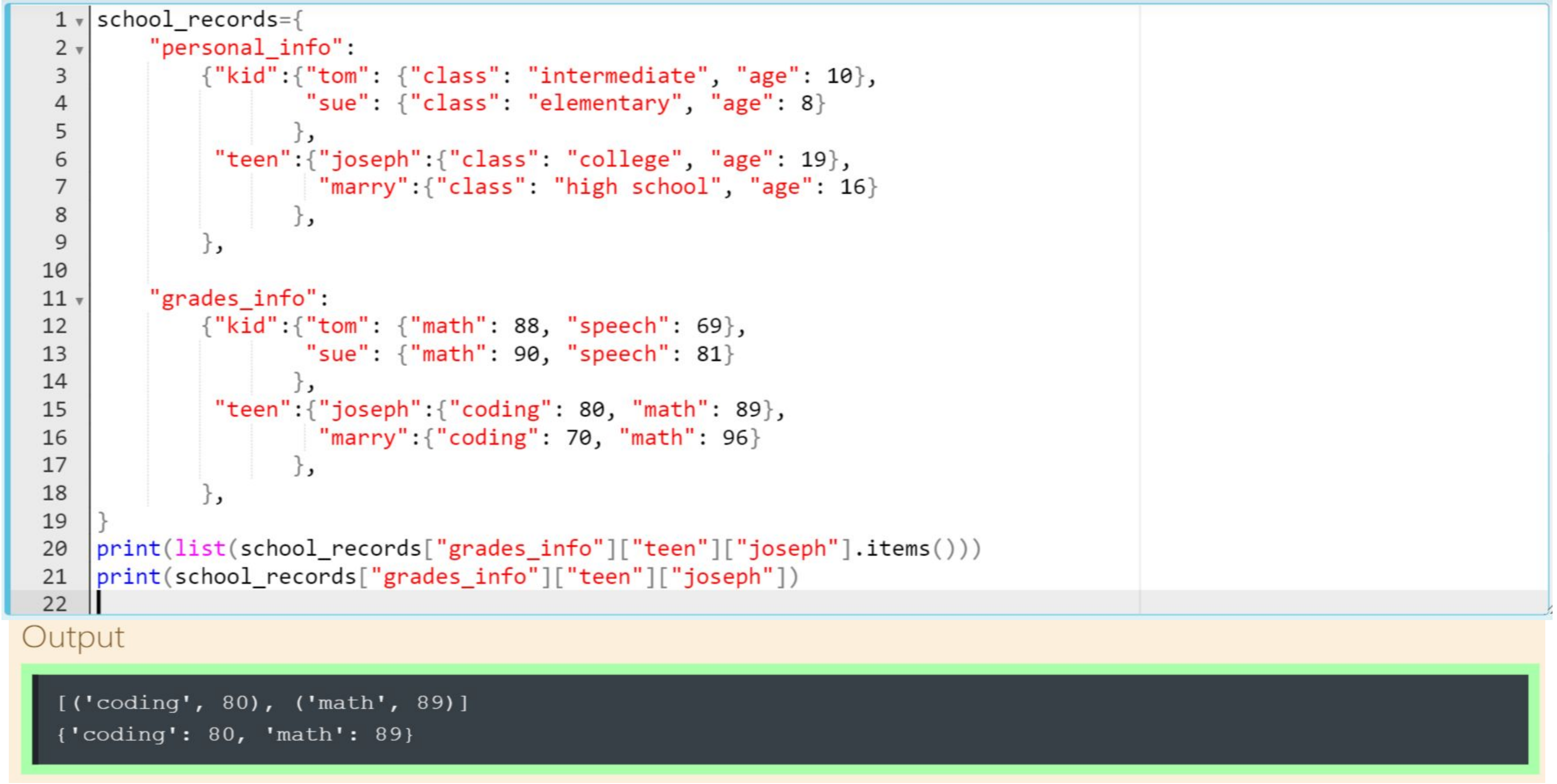


```
school_records={
    "personal_info":
        {"kid":{"tom": {"class": "intermediate", "age": 10},
                "sue": {"class": "elementary", "age": 8}
               },
         "teen":{"joseph":{"class": "college", "age": 19},
                 "marry":{"class": "high school", "age": 16}
                },
        },

    "grades_info":
        {"kid":{"tom": {"math": 88, "speech": 69},
                "sue": {"math": 90, "speech": 81}
               },
         "teen":{"joseph":{"coding": 80, "math": 89},
                 "marry":{"coding": 70, "math": 96}
                },
        },
}
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Nested `dicts`

- The code can be like :



```
school_records={
    "personal_info":
        {"kid":{"tom": {"class": "intermediate", "age": 10},
                "sue": {"class": "elementary", "age": 8}
                },
         "teen":{"joseph":{"class": "college", "age": 19},
                 "marry":{"class": "high school", "age": 16}
                },
        },

    "grades_info":
        {"kid":{"tom": {"math": 88, "speech": 69},
                "sue": {"math": 90, "speech": 81}
                },
         "teen":{"joseph":{"coding": 80, "math": 89},
                 "marry":{"coding": 70, "math": 96}
                },
        },
}
print(list(school_records["grades_info"]["teen"]["joseph"].items()))
print(school_records["grades_info"]["teen"]["joseph"])
```

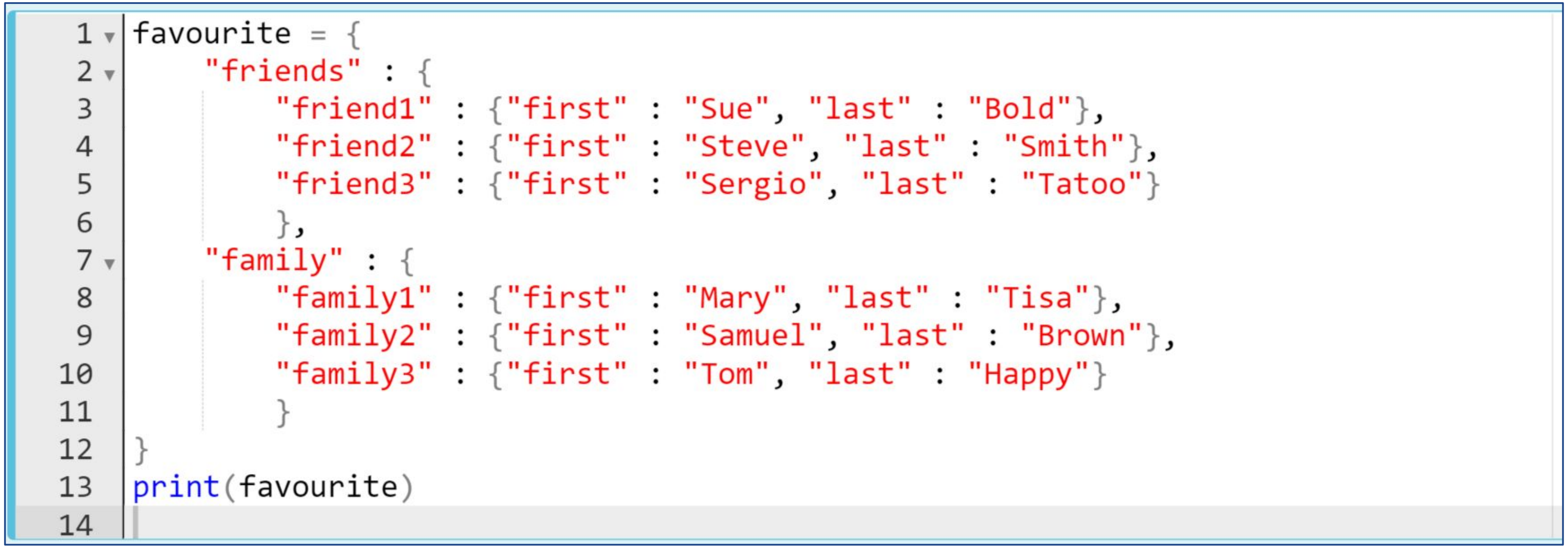
Output

```
[('coding', 80), ('math', 89)]
{'coding': 80, 'math': 89}
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Nested `dicts`

- What *statement* will **remove** the entry in the dictionary for key `'family3'`?

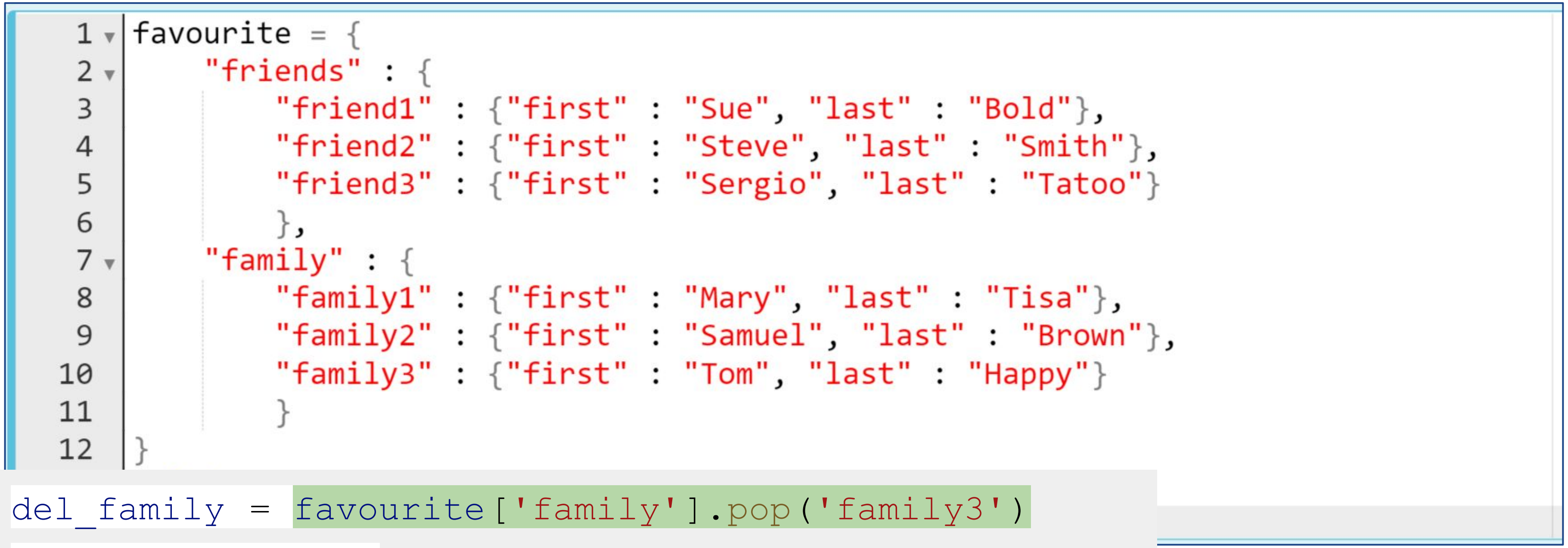


```
favourite = {
        "friends" : {
                "friend1" : {"first" : "Sue", "last" : "Bold"},
                "friend2" : {"first" : "Steve", "last" : "Smith"},
                "friend3" : {"first" : "Sergio", "last" : "Tatoo"}
                },
        "family" : {
                "family1" : {"first" : "Mary", "last" : "Tisa"},
                "family2" : {"first" : "Samuel", "last" : "Brown"},
                "family3" : {"first" : "Tom", "last" : "Happy"}
                }
}
print(favourite)
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Nested `dicts`

- What *statement* will **remove** the entry in the dictionary for key `'family3'`?



```
favourite = {
        "friends" : {
            "friend1" : {"first" : "Sue", "last" : "Bold"},
            "friend2" : {"first" : "Steve", "last" : "Smith"},
            "friend3" : {"first" : "Sergio", "last" : "Tatoo"}
            },
        "family" : {
            "family1" : {"first" : "Mary", "last" : "Tisa"},
            "family2" : {"first" : "Samuel", "last" : "Brown"},
            "family3" : {"first" : "Tom", "last" : "Happy"}
            }
}
```

```
del_family = favourite['family'].pop('family3')
print(del_family)
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

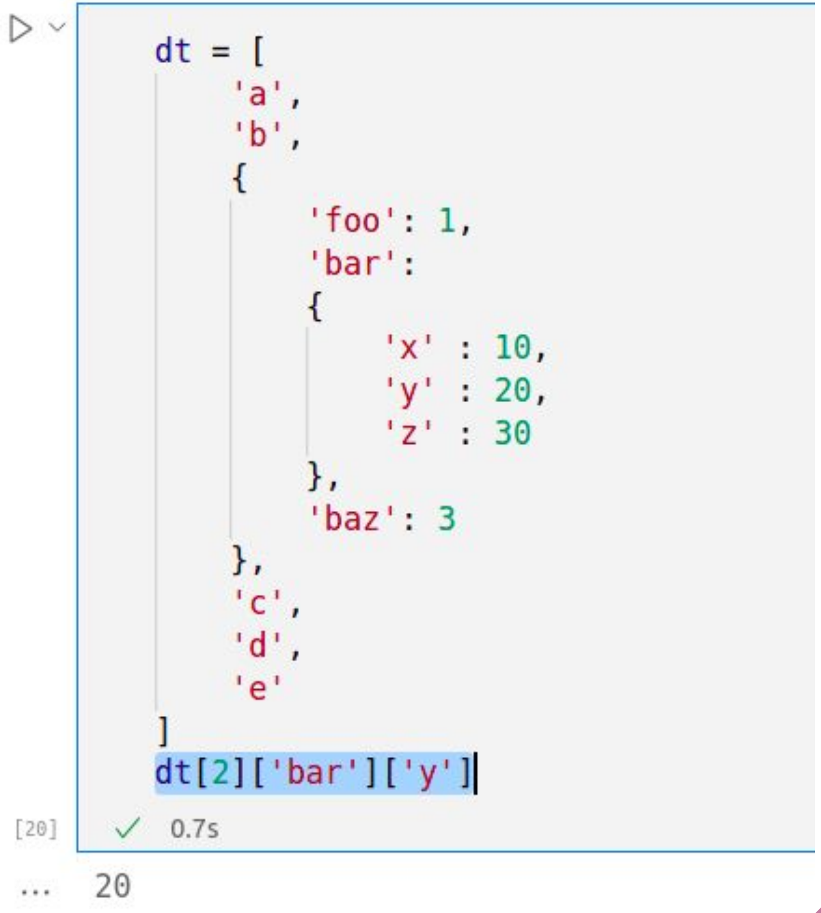
Nested `collections`

- What is the expression involving **`y`** that **access**es the value `20`?

```
dt = [
    'a',
    'b',
    {
        'foo': 1,
        'bar':
        {
            'x' : 10,
            'y' : 20,
            'z' : 30
        },
        'baz': 3
    },
    'c',
    'd',
    'e'
]
```

Nested `collections`

- What is the expression involving `y` that accesses the value `20`?



```
dt = [
    'a',
    'b',
    {
        'foo': 1,
        'bar':
        {
            'x' : 10,
            'y' : 20,
            'z' : 30
        },
        'baz': 3
    },
    'c',
    'd',
    'e'
]
dt[2]['bar']['y']
```

```
[20]  ✓  0.7s
...   20
```

CLARUSWAY©
WAY TO REINVENT YOURSELF