1. Populating a dictionary

Create a dictionary by using all the given variables.

In [10]:
```
first_name = 'John'
last_name = 'Doe'
favorite_hobby = 'Python'
sports_hobby = 'gym'
age = 82
```

In [11]:
```
# Your implementation
my_dict = {'name': first_name + " " + last_name,
           "age": age,
           "hobbies": [favorite_hobby, sports_hobby]
           }
```

In [12]:
```
assert my_dict == {
        'name': 'John Doe',
        'age': 82,
        'hobbies': ['Python', 'gym']
    }
```

2. Accessing and merging dictionaries

Combine `dict1`, `dict2`, and `dict3` into `my_dict`. In addition, get the value of `special_key` from `my_dict` into a `special_value` variable. Note that original dictionaries should stay untouched and `special_key` should be removed from `my_dict`.

In [29]:
```
dict1 = dict(key1='This is not that hard', key2='Python is still cool')
dict2 = {'key1': 123, 'special_key': 'secret'}
# This is also a away to initialize a dict (list of tuples)
dict3 = dict([('key2', 456), ('keyX', 'X')])
```

In [30]:
```
# 'Your impelementation'
my_dict = dict1.copy()

my_dict.update(dict2)
my_dict.update(dict3)


special_value = my_dict['special_key']

my_dict.pop("special_key")
```

Out[30]:
```
'secret'
```

In [31]:
```
assert my_dict == {'key1': 123, 'key2': 456, 'keyX': 'X'}
assert special_value == 'secret'

# Let's check that the originals are untouched
```

```
assert dict1 == {
        'key1': 'This is not that hard',
        'key2': 'Python is still cool'
    }
assert dict2 == {'key1': 123, 'special_key': 'secret'}
assert dict3 == {'key2': 456, 'keyX': 'X'}
```