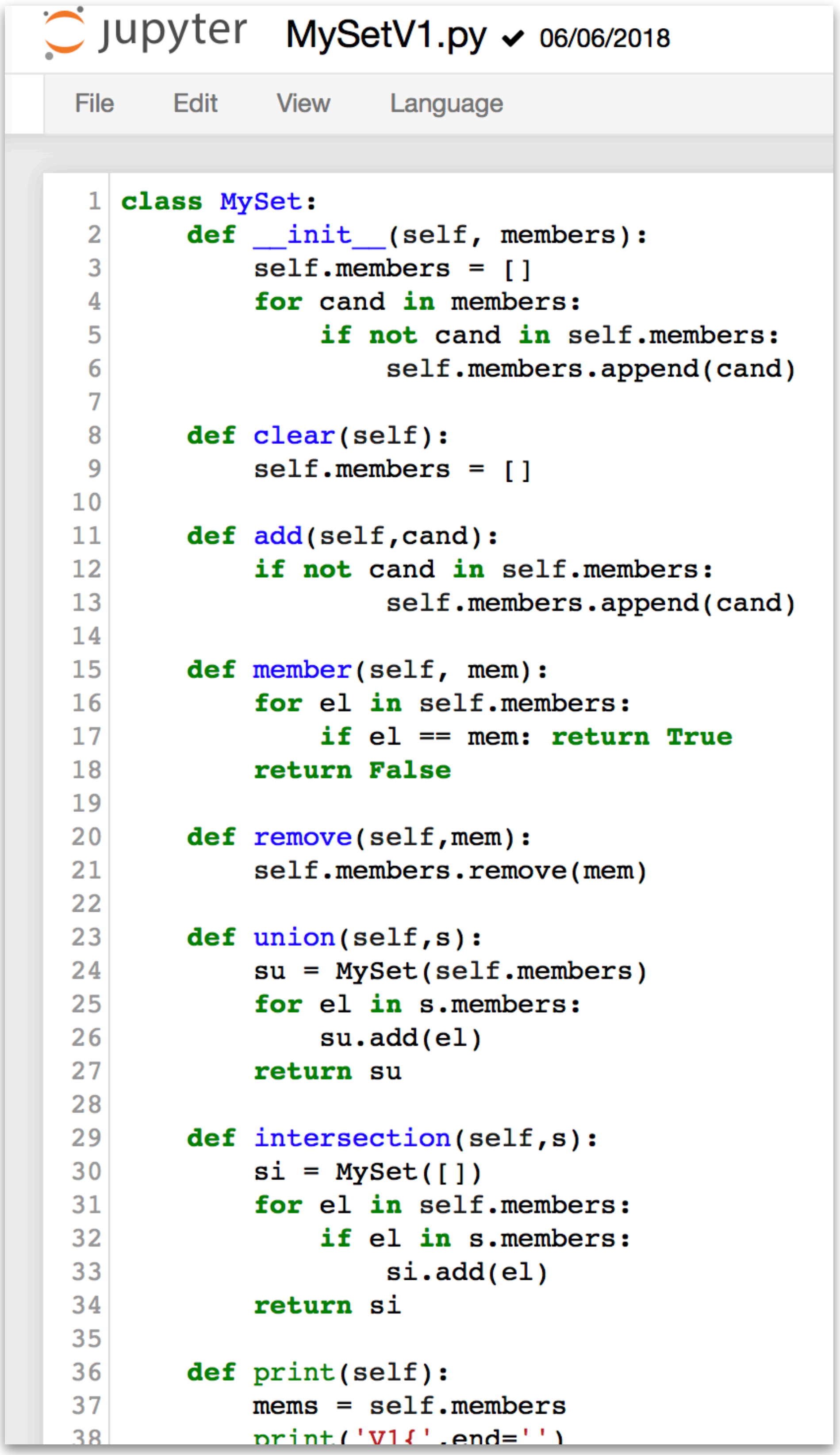
Creating Modules

- We can create modules with the MySet class definitions
  - MySetV1.py
  - MySetV2.py
- These modules can be imported (used) elsewhere
  - MySet_Cards notebook

MySetV1.py

- Store the MySet definitions in MySetV1.py

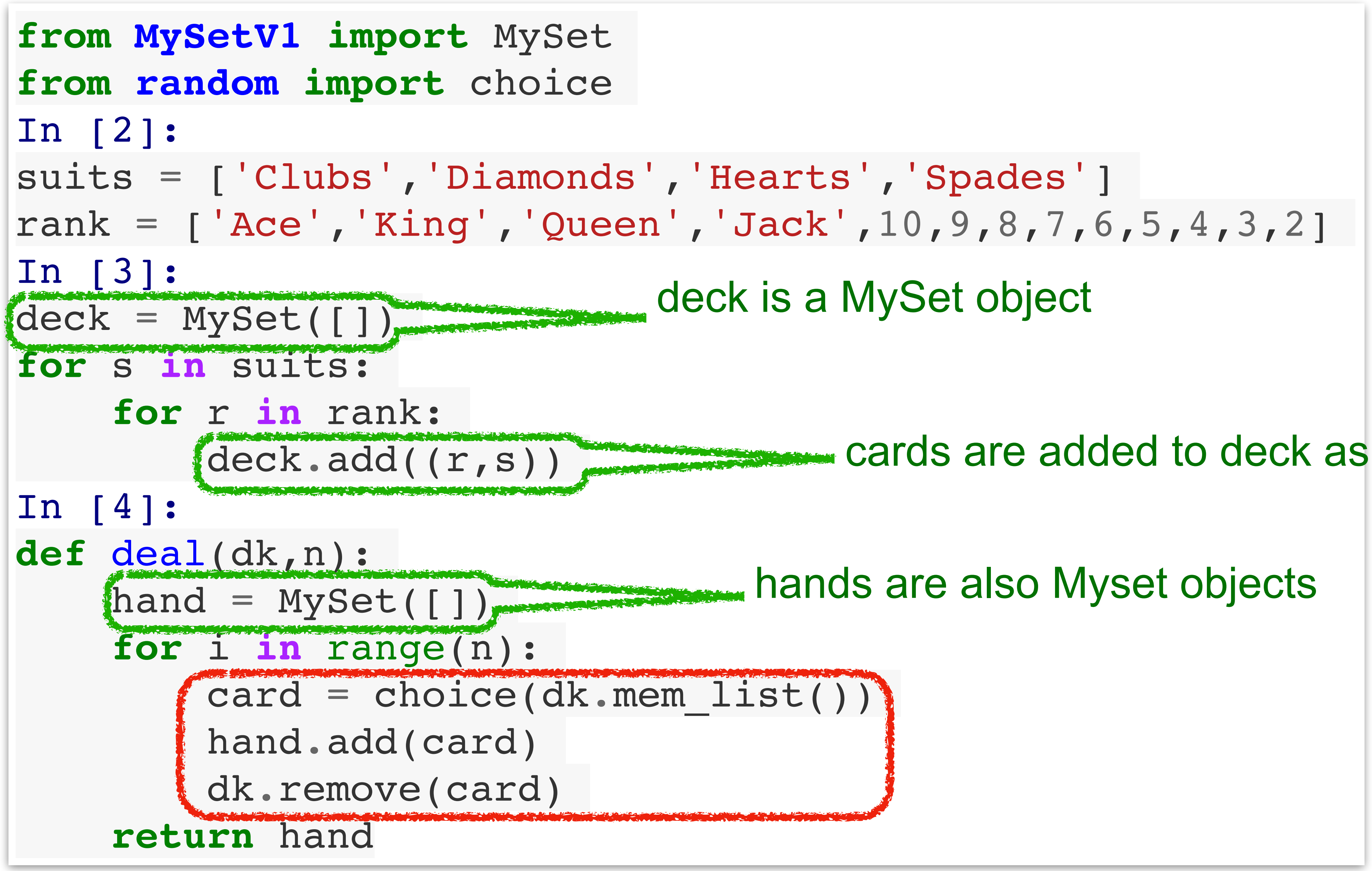
```python
from MySetV1 import MySet
```



```python
class MySet:
    def __init__(self, members):
        self.members = []
        for cand in members:
            if not cand in self.members:
                self.members.append(cand)

    def clear(self):
        self.members = []

    def add(self,cand):
        if not cand in self.members:
                self.members.append(cand)

    def member(self, mem):
        for el in self.members:
            if el == mem: return True
        return False

    def remove(self,mem):
        self.members.remove(mem)

    def union(self,s):
        su = MySet(self.members)
        for el in s.members:
            su.add(el)
        return su

    def intersection(self,s):
        si = MySet([])
        for el in self.members:
            if el in s.members:
                si.add(el)
        return si

    def print(self):
        mems = self.members
        print('V1{',end='')
```

MySet_Cards

- A notebook that uses the MySet class



```
from MySetV1 import MySet
from random import choice
In [2]:
suits = ['Clubs','Diamonds','Hearts','Spades']
rank = ['Ace','King','Queen','Jack',10,9,8,7,6,5,4,3,2]
In [3]:
deck = MySet([])
for s in suits:
    for r in rank:
        deck.add((r,s))
In [4]:
def deal(dk,n):
    hand = MySet([])
    for i in range(n):
        card = choice(dk.mem_list())
        hand.add(card)
        dk.remove(card)
    return hand
```

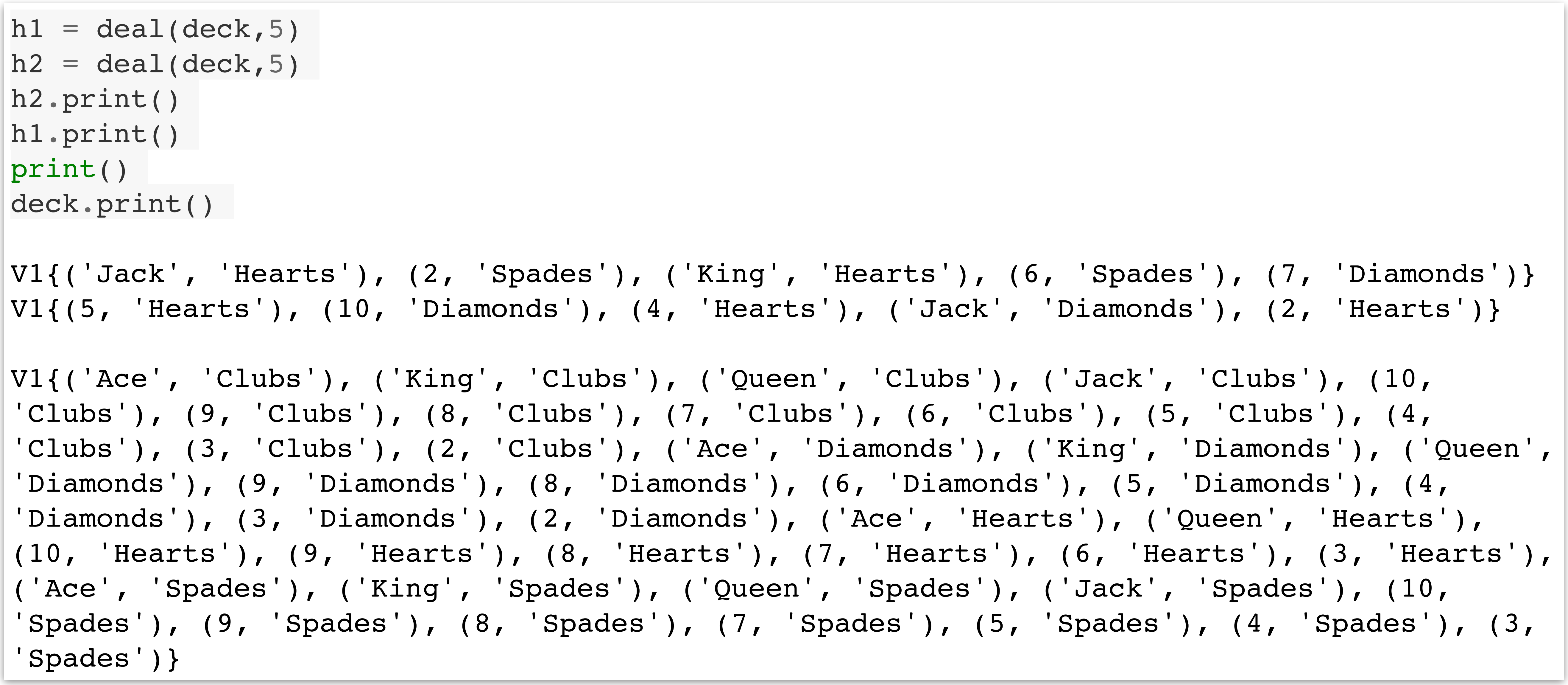
deck is a MySet object

cards are added to deck as tuples

hands are also Myset objects

MySet_Cards notebook

- Deal two hands from deck
  - dealt cards removed from deck

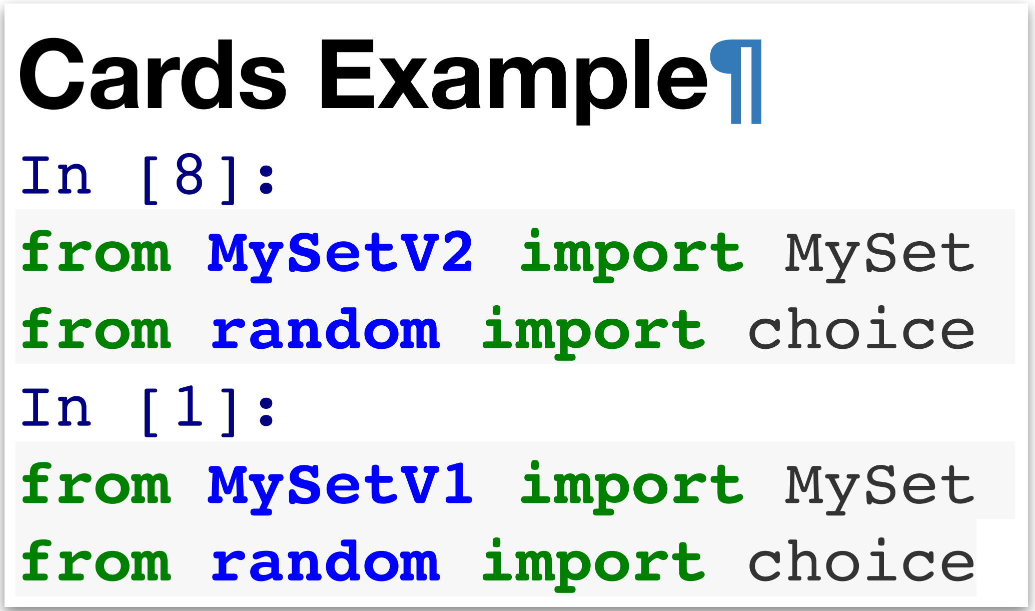


```
h1 = deal(deck,5)
h2 = deal(deck,5)
h2.print()
h1.print()
print()
deck.print()
```

```
V1{('Jack', 'Hearts'), (2, 'Spades'), ('King', 'Hearts'), (6, 'Spades'), (7, 'Diamonds')}
V1{(5, 'Hearts'), (10, 'Diamonds'), (4, 'Hearts'), ('Jack', 'Diamonds'), (2, 'Hearts')}

V1{('Ace', 'Clubs'), ('King', 'Clubs'), ('Queen', 'Clubs'), ('Jack', 'Clubs'), (10,
'Clubs'), (9, 'Clubs'), (8, 'Clubs'), (7, 'Clubs'), (6, 'Clubs'), (5, 'Clubs'), (4,
'Clubs'), (3, 'Clubs'), (2, 'Clubs'), ('Ace', 'Diamonds'), ('King', 'Diamonds'), ('Queen',
'Diamonds'), (9, 'Diamonds'), (8, 'Diamonds'), (6, 'Diamonds'), (5, 'Diamonds'), (4,
'Diamonds'), (3, 'Diamonds'), (2, 'Diamonds'), ('Ace', 'Hearts'), ('Queen', 'Hearts'),
(10, 'Hearts'), (9, 'Hearts'), (8, 'Hearts'), (7, 'Hearts'), (6, 'Hearts'), (3, 'Hearts'),
('Ace', 'Spades'), ('King', 'Spades'), ('Queen', 'Spades'), ('Jack', 'Spades'), (10,
'Spades'), (9, 'Spades'), (8, 'Spades'), (7, 'Spades'), (5, 'Spades'), (4, 'Spades'), (3,
'Spades')}
```

Back to MySet_Cards

- Two options, import V1 or V2
- `deck.print()` method reveals which option is in use
  - (This is only to let us see what is going on.)



Cards Example¶

```
In [8]:
from MySetV2 import MySet
from random import choice
In [1]:
from MySetV1 import MySet
from random import choice
```

Summary:

- We can swap in a new implementation of MySet because implementation details are hidden (i.e. ***encapsulated***) in the class definition.

L07 MySet_Cards notebook

Exercise

- What happens when we try to deal from an empty deck?
  - i.e. deal more cards than are in the deck
- Include some exception handling in the deal() function to cover this.