1장 데이터 모델 및 데이터 구조체

- 데이터 모델
- 데이터 구조체(시퀀스)

1. Python Data Model

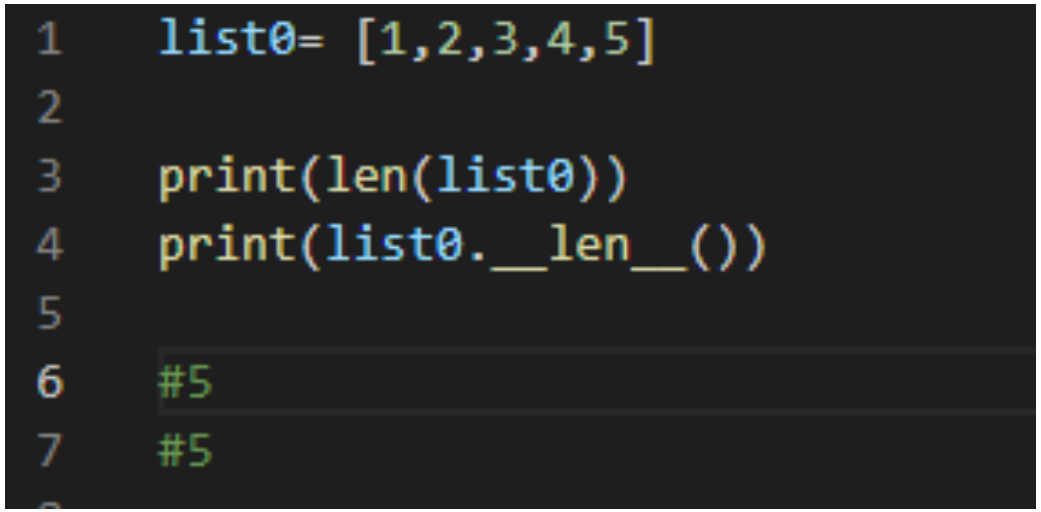
Guido's sense of the aesthetics of language design is amazing. I've met many fine language designers who could build theoretically beautiful languages that no one would ever use, but Guido is one of those rare people who can build a language that is just slightly less theoretically beautiful but thereby is a joy to write programs in[1].

— Jim Hugunin

creator of Jython, co-creator of AspectJ, architect of the .Net DLR

1. Python Data Model

- Pythonic
  - collection.len() : 다른 객체지향 언어 사용법(java, C++,…)
  - len(collection) : python에서 사용하는 사용법



1. Python Data Model

- 데이터모델은 일종의 프레임워크로서 구성 단위에 대한 인터페이스 정의
- 파이썬 데이터 모델 =
  객체 모델(다른 언어) or 파이썬 객체모델
  - 시퀀스
  - 반복자
  - 함수
  - 클래스
  - 콘텍스트 관리자 등

1.1 파이썬 카드 한벌

```
import collections

Card = collections.namedtuple('Card', ['rank', 'suit'])

class FrenchDeck:
    ranks = [str(n) for n in range(2, 11)] + list('JQKA')
    suits = 'spades diamonds clubs hearts'.split()

    def __init__(self):
        self._cards = [Card(rank, suit) for suit in self.suits
                                        for rank in self.ranks]

    def __len__(self):
        return len(self._cards)

    def __getitem__(self, position):
        return self._cards[position]
```

```
>>> beer_card = Card('7', 'diamonds')
>>> beer_card
Card(rank='7', suit='diamonds')
```

```
>>> deck = FrenchDeck()
>>> len(deck)
52
```

```
>>> deck[0]
Card(rank='2', suit='spades')
>>> deck[-1]
Card(rank='A', suit='hearts')
```

1.1 파이썬 카드 한벌

```
>>> from random import choice
>>> choice(deck)
Card(rank='3', suit='hearts')
>>> choice(deck)
Card(rank='K', suit='spades')
```

```
>>> choice(deck)
Card(rank='2', suit='clubs')
```

```
>>> deck[:3]
[Card(rank='2', suit='spades'), Card(rank='3', suit='spades'),
Card(rank='4', suit='spades')]
>>> deck[12::13]
[Card(rank='A', suit='spades'), Card(rank='A', suit='diamonds'),
Card(rank='A', suit='clubs'), Card(rank='A', suit='hearts')]
```

1.1 파이썬 카드 한벌

```
>>> for card in deck:  # doctest: +ELLIPSIS
...   print(card)
Card(rank='2', suit='spades')
Card(rank='3', suit='spades')
Card(rank='4', suit='spades')
```

```
>>> for card in reversed(deck):  # doctest: +ELLIPSIS
...   print(card)
Card(rank='A', suit='hearts')
Card(rank='K', suit='hearts')
Card(rank='Q', suit='hearts')
...
```

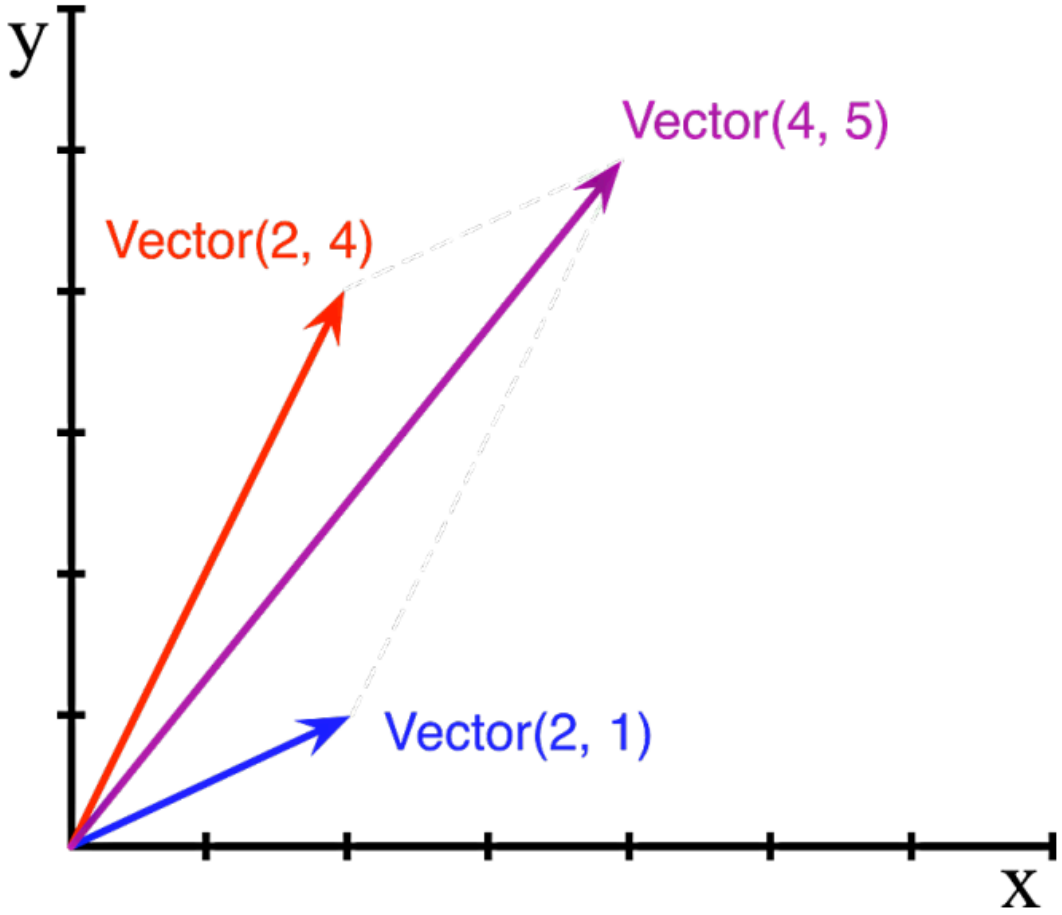
```
suit_values = dict(spades=3, hearts=2, diamonds=1, clubs=0)

def spades_high(card):
    rank_value = FrenchDeck.ranks.index(card.rank)
    return rank_value * len(suit_values) + suit_values[card.suit]
```

1.1 파이썬 카드 한벌

```
>>> for card in sorted(deck, key=spades_high):  # doctest: +ELLIPSIS
...       print(card)
Card(rank='2', suit='clubs')
Card(rank='2', suit='diamonds')
Card(rank='2', suit='hearts')
... (46 cards ommitted)
Card(rank='A', suit='diamonds')
Card(rank='A', suit='hearts')
Card(rank='A', suit='spades')
```

1.2 Emulating Numeric Type




```
>>> v1 = Vector(2, 4)
>>> v2 = Vector(2, 1)
>>> v1 + v2
Vector(4, 5)
```

```
>>> v = Vector(3, 4)
>>> abs(v)
5.0
```

```
>>> v * 3
Vector(9, 12)
>>> abs(v * 3)
15.0
```

1.2 Emulating Numeric Type

- __init__ : initialization
- __repr__ : representation
- __abs__ : Euclidean distance
  - hypot() : hypotenuse
- __add__ : addition
- __mul__ : multiplication

```python
from math import hypot

class Vector:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __repr__(self):
        return 'Vector(%r, %r)' % (self.x, self.y)

    def __abs__(self):
        return hypot(self.x, self.y)

    def __bool__(self):
        return bool(abs(self))

    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return Vector(x, y)

    def __mul__(self, scalar):
        return Vector(self.x * scalar, self.y * scalar)
```