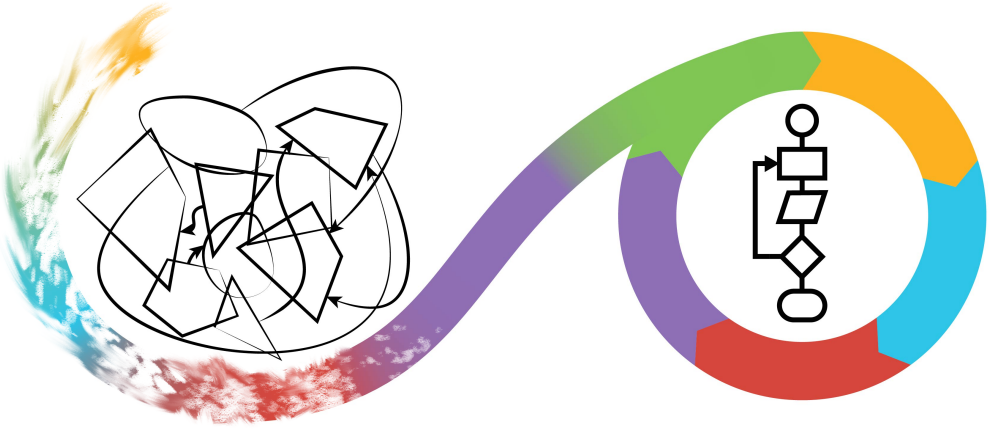
Data structures

Richèl Bilderbeek



Problem

Are there classes that can help me solve problems more elegantly?

How do I write these myself?

Python classes

- (`list`: heterogeneous container)
- (`tuple`: immutable `list`)
- `set`: sets
- `dict`: dictionary
- Regular expressions: text patterns

From Python 'Data Structures' documentation

`set`

Sorted collection of unique elements.

```
data = [3, 1, 4, 1, 5]
s = set(data)
assert 3 in s
assert list(s) == [1, 3, 4, 5]
```

- No need to check for elements existing twice

`set` example for `are_primes`

- `are_primes` determines of each value in a list if it is prime yes/no
- Problem: values in that list can occur multiple times, e.g. `[42, 42, 42]`
- Solution with a `set`:
  - Collect all unique values in the input
  - Put the unique values that are prime in a set
  - Check each input value to be in the set of primes

`dict`

A dictionary:

```
periodic_table = dict({1: "Hydrogen", 2: "Helium", 3: "Lithium"})
periodic_table[2] = "helium"
assert periodic_table[2] == "helium"
```

- Commonly uses as a look-up table
- A look-up table can store the results of earlier calculations

`dict` example for `are_primes`

- `are_primes` determines of each value in a list if it is prime yes/no
- Problem: values in that list can occur multiple times, e.g. `[42, 42, 42]`
- Solution with a `set`:
  - Collect all unique values in the input
  - Store for each unique values if it is prime yes/no
  - Look up each input value in the prime look-table

Regular expressions

A state-machine for a pattern in text

```
import re
dna_regex = re.compile("^[ACGT]*$")
assert dna_regex.match("")
assert dna_regex.match("A")
assert dna_regex.match("CA")
assert dna_regex.match("GCA")
assert dna_regex.match("TGCA")
assert dna_regex.match("TGCAAAAAA")
assert not dna_regex.match("nonsense")
```

- https://docs.python.org/3/library/re.html

Our own class

- https://docs.python.org/3/tutorial/classes.html

`DnaSequence`