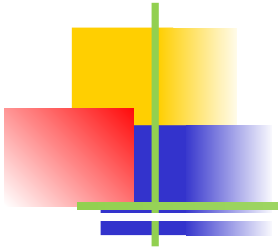
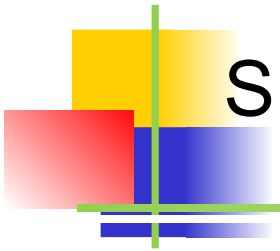


Fundamentos de Programação

João Rodrigues

Departamento de Electrónica, Telecomunicações e Informática

Universidade de Aveiro

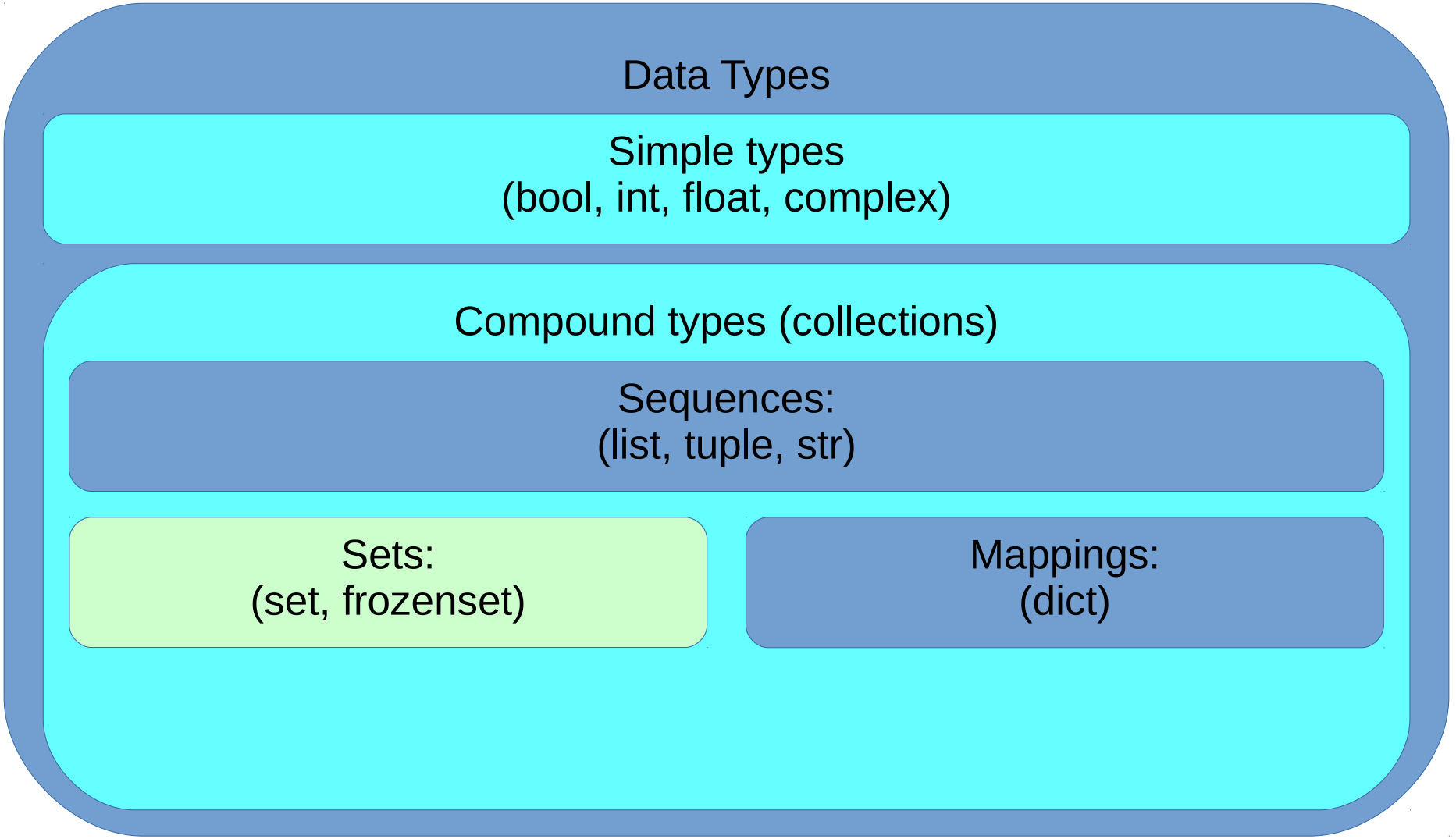


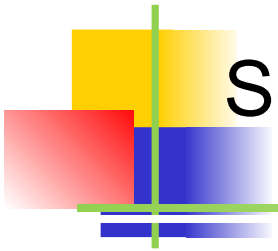
Summary

- Sets

Sets

- Python data types





Set types

- A **set** is an *unordered collection* of *unique* elements.
  - A **collection** because they may contain zero or more elements.
  - Elements are **unique**: they may not be repeated!
  - **Unordered** because elements are not in sequence.
- Sets are a fundamental data type in Math and computing.
- In Python, a set may be created using braces.

```
fruits = {'pear', 'apple', 'banana', 'orange'}
S = { x for x in fruits if x<'c' }  # (by comprehension)
```

- The `set` constructor converts from other types.

```
numbers = set([3, 1, 3])   #→ {1, 3}
```

- The empty set must be created with `set()`, because `{}` is a dictionary.

```
empty = set()
```

Elements in a set are unique

- An object either is or is not in a set. It cannot be in the set more than once!

```
{1, 2, 1} == {1, 2}

len({4, 5, 4, 5, 5})   #→ 2
```

- Like *keys* in a dictionary.
- Unlike sequences. `[1, 2, 1] != [1, 2]`.
- A common application of sets is for eliminating duplicated elements in sequences.

```
set([1, 2, 2, 2, 1])    #→ {1, 2}

set('banana')    #→ {'a', 'b', 'n'}
```

- This eliminates order, too.

Elements in a set are unordered

- Sets don't recall the position or order of entry of elements.

```
s = {3, 1, 2}      #→ {1, 2, 3}
s == {2 , 3, 1} == {1, 2, 3}  #→ True
```

- So, indexing, slicing, concatenation are not allowed!

```
s[0]       # TypeError
s[0:2]     # TypeError
s + {4}    # TypeError
```

Elements in a set must be hashable

- A set may contain elements of various types, but only *hashable* types are allowed.
- Just like dictionary *keys*.
- Simple immutable types (like numbers) are OK.
- Strings and tuples (*if their elements are hashable*) are OK, too.

```
{ 23, 'eggs', (1997,10,23) }
```

- Lists, dictionaries, sets and other mutable types are not allowed!

```
{ [1,2] }       # TypeError
{ {1}, {1,2} } # TypeError
```

- What are hashable types? (Read this description.)

Operations on sets

- Sets have a length and support the membership operator.

```
S = {23, 5, 12}
len(S)     # 3
5 in S     # True (This is a fast operation!)
```

- Sets can be intersected, united, subtracted.

```
{3,4,5} & {1,2,3}   #-> {3}
{3,4,5} | {1,2,3}   #-> {1,2,3,4,5}
{3,4,5} - {1,2,3}   #-> {4,5}
{3,4,5} ^ {1,2,3}   #-> {1,2,4,5}
```

Operations on sets (2)

- Sets may be compared for equality.

```
{2,2,1} == {1,2}  # True
```

- We may test subset or superset relations.

```
S = {1,2}
S <= {1,2,3}    # True = S.issubset({1,2,3})
S < {1,2}       # False
S >= {2}        # True = S.issuperset({2})
S > {2}         # True
```

- But this is *not a total ordering* relation! You can have two sets A and B such that:

```
A<B, A==B, A>B  # All are False!
```

Sets are mutable

- We can add or remove elements in sets.

```
S = {1,2,3}
S.add(4)          # S → {1,2,3,4}
S.remove(2)       # S → {1,3,4}
```

- We can update the set by union, intersection or differences.

```
S |= {3,5,7}          # S → {1,3,4,5,7}
S.update({3,5,7})     # Same thing
S &= {1,2,3,4}        # S → {1,3,4}
S -= {4,5,6}          # S → {1,3}
S ^= {1,2,4}          # S → {2,3,4}
```

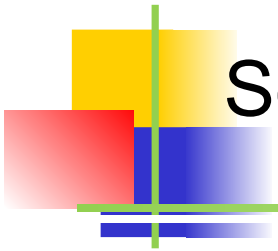
- Python also has immutable sets: the `frozenset` type.

```
T = frozenset({1,2,3})
```

How to select the proper data type?

- Choosing the right type to store your data is very important.
- First, consider the different characteristics of the types.

Type	Type identifier	Collection?	Sequence?	Mutable?	Element type
Simple types	`bool, int, float, complex`	no (simple)	---	no	---
Strings	`str`	yes	yes	no	(characters)
Tuples	`tuple`	yes	yes	no	any type
Lists	`list`	yes	yes	yes	any type*
Dictionaries	`dict`	yes	No (unnorderable)	yes	keys: hashable, values: any type*
Sets	`set`	yes	No (unnorderable)	yes	hashable
Immutable sets	`frozenset`	yes	No (unnorderable)	no	hashable



Some questions to help deciding

- Are the data simple or compound (several elements)?
- Does element order/position matter? Yes => sequence.
- Will the contents grow, shrink or change? Yes => mutable.
- Need to quickly map a key to a value? Yes => dictionary.