Forritunarmálið Python

Day 2

Data structures

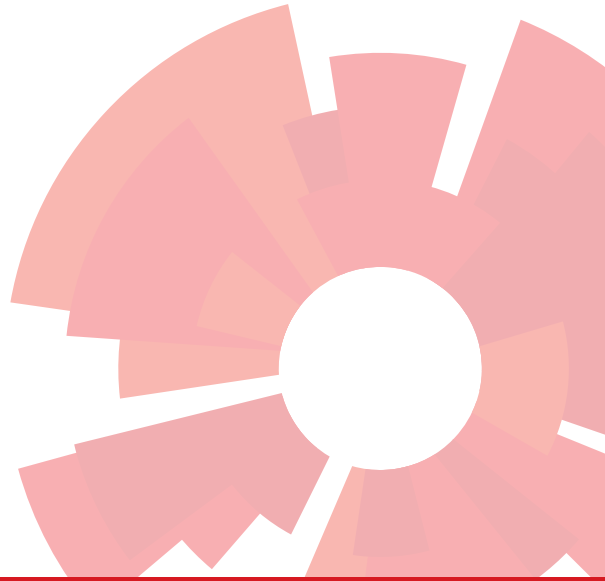
Hjalti Magnússon

28. nóvember 2017

Lists

Properties

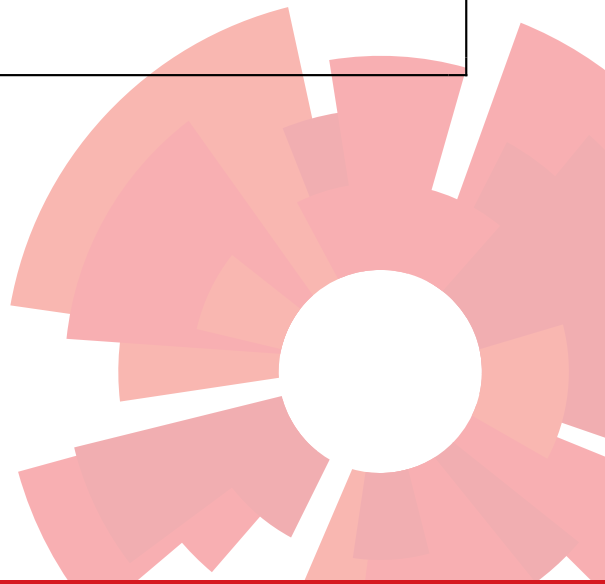
- Resizable (mutable) array
  - Constant time lookup
  - (Amortized) constant time append
  - Linear insert
- One of Python's *iterables*



Constructor

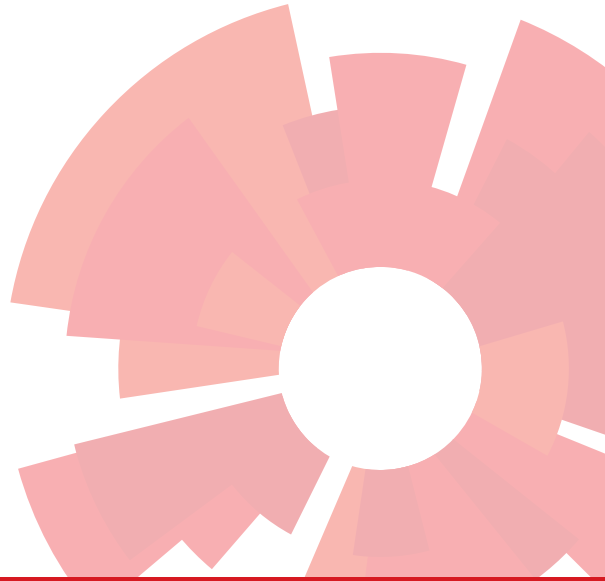
- We can convert any iterable to a list

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list('string')
['s', 't', 'r', 'i', 'n', 'g']
```



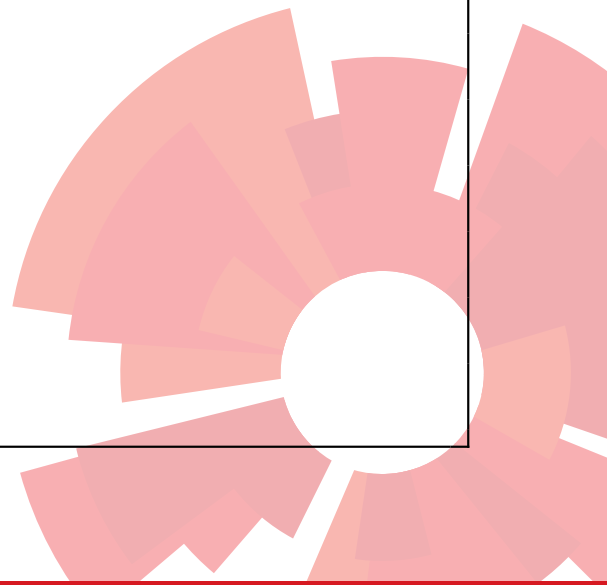
Builtin functions/keywords

- These work on most or all iterables
  - `in`
  - `len`
  - `max, min`
  - `sum`
  - `del`
  - `reversed`
  - `sorted`



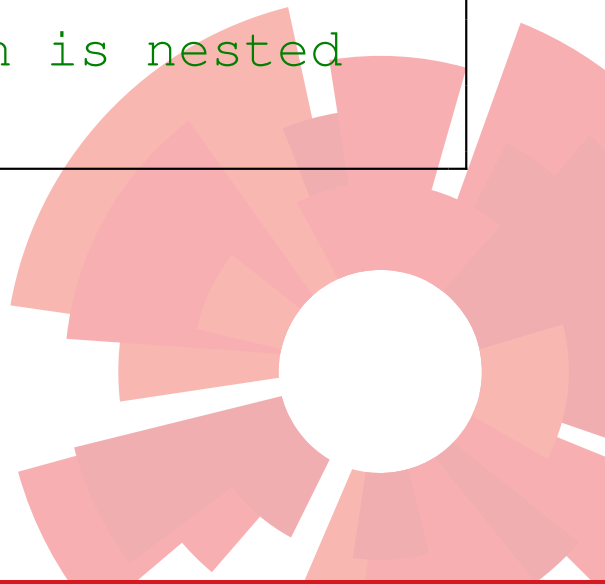
Slicing

```
# index      0,  1,  2,  3,  4,  5,  6,  7
>>> lis =  [1,  2,  3,  4,  5,  6,  7,  8]
# rev ind  -8, -7, -6, -5, -4, -3, -2, -1

>>> lis[2:4]
[3, 4]
>>> lis[:4]
[1, 2, 3, 4]
>>> lis[2:]
[3, 4, 5, 6, 7, 8]
>>> lis[::2]
[1, 3, 5, 7]
>>> lis[5:2:-1]
[6, 5, 4]
>>> lis[::-1]
[8, 7, 6, 5, 4, 3, 2, 1]
```



Comparison

```
>>> a = [1, 2, 3]
>>> b = [3, 2, 1]
>>> a == b[::-1]
True

>>> c = [1, 2, [5, 6, 7]]
>>> d = [[5, 6, 7], 2, 1]
>>> c == d[::-1]              # Comparison is nested
True
```



Comparison

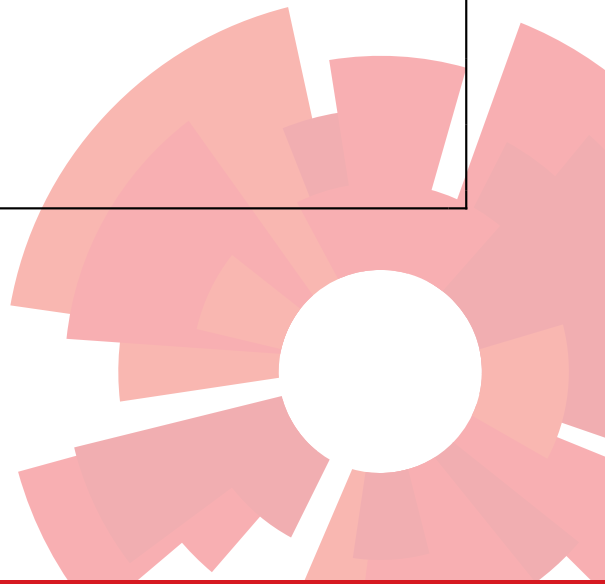
For iterables, Python uses lexicographical ordering to determine which is lesser or greater

Lexicographical order

- Suppose we have two lists A and B
- Check if $A < B$
- For each element $a \in A$ and $b \in B$
  - If $a < b$, return true
  - If $a > b$, return false
  - if $a == b$, compare next elements
- If there are more elements in B than A, return false; otherwise, return true

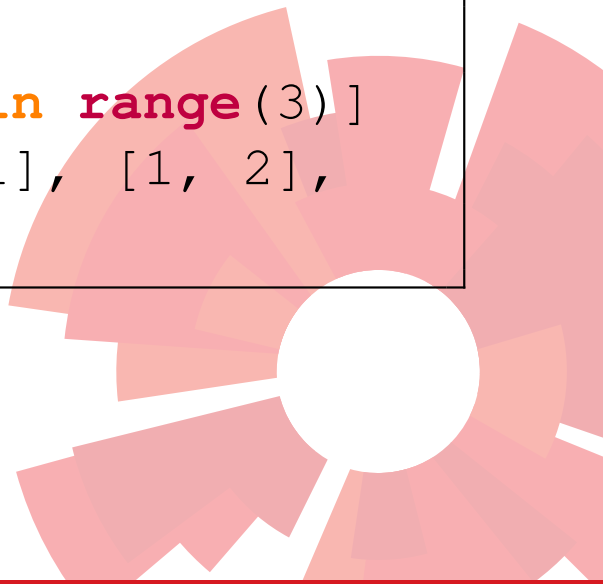
Comparison

```
>>> [1, 2, 3] < [1, 2, 8]
True
>>> [1, 2, 3] < [1, 2, 3, 4]
True
>>> [1, 3, 2] < [1, 2, 3]
False
>>> [1, 2, 3] < [1, 2, 3]
False
>>> [] < [1, 2, 3]
True
```



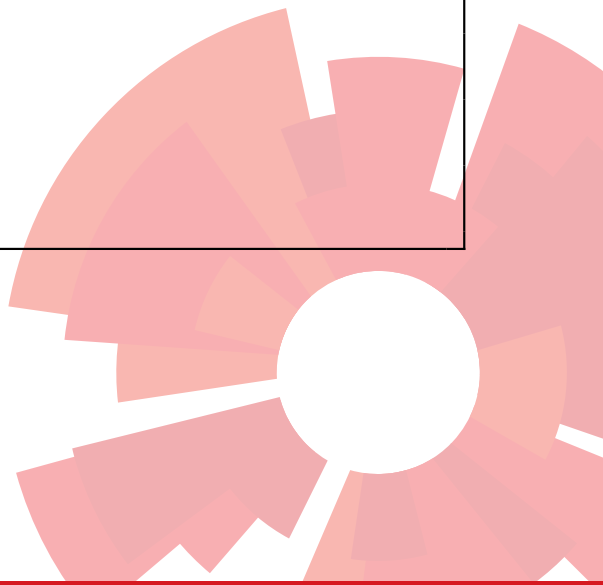
List comprehension

```
>>> [n for n in range(10)]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> [n ** 2 for n in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

>>> [a * b for a in range(3) for b in range(4)]
[0, 0, 0, 0, 0, 1, 2, 3, 0, 2, 4, 6]

>>> [[a,  b] for a in range(3) for b in range(3)]
[[0, 0], [0, 1], [0, 2], [1, 0], [1, 1], [1, 2],
 [2, 0], [2, 1], [2, 2]]
```



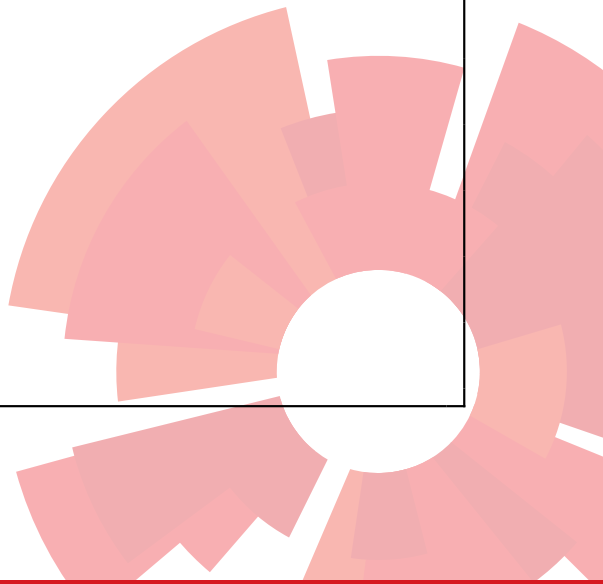
List comprehension

```
>>> [n for n in range(100) if n % 10 == 3]
[3, 13, 23, 33, 43, 53, 63, 73, 83, 93]

>>> [n for n in range(10) if n % 10 not in [5, 6]]
[0, 1, 2, 3, 4, 7, 8, 9]

>>> [[a for a in range(3)] for b in range(3)]
[[0, 1, 2], [0, 1, 2], [0, 1, 2]]

>>> [0 for _ in range(10)]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```



Gotchas

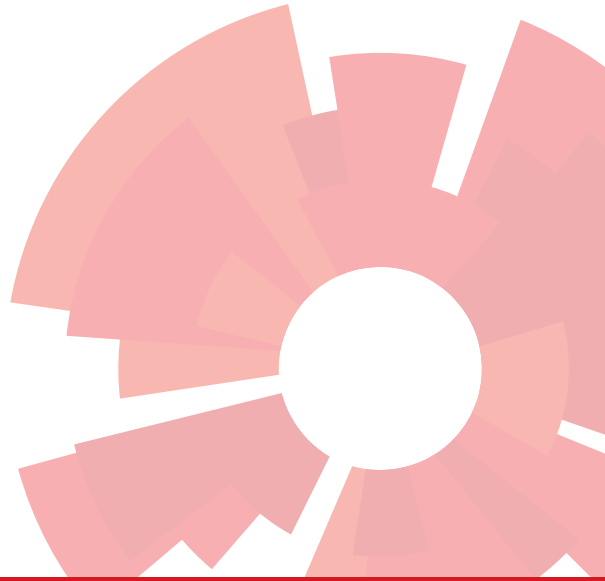
```
>>> lis = 3 * [ [1, 2, 3] ]
>>> lis
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]

>>> lis[0][1] = 5
>>> lis
[[1, 5, 3], [1, 5, 3], [1, 5, 3]]

>>> lis = [[1, 2, 3] for _ in range(3)]
>>> lis
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]

>>> lis[0][1] = 5
>>> lis
[[1, 5, 3], [1, 2, 3], [1, 2, 3]]
```



Tuples

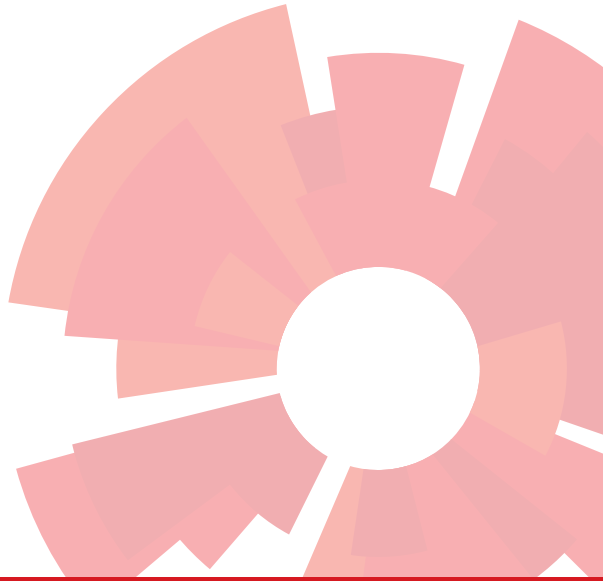
What are they?

- Read only lists
- Usages
  - Multiple return values (`divmod`)
  - Denote non-mutable list of items, e.g. points in a grid
- Tuple assignment



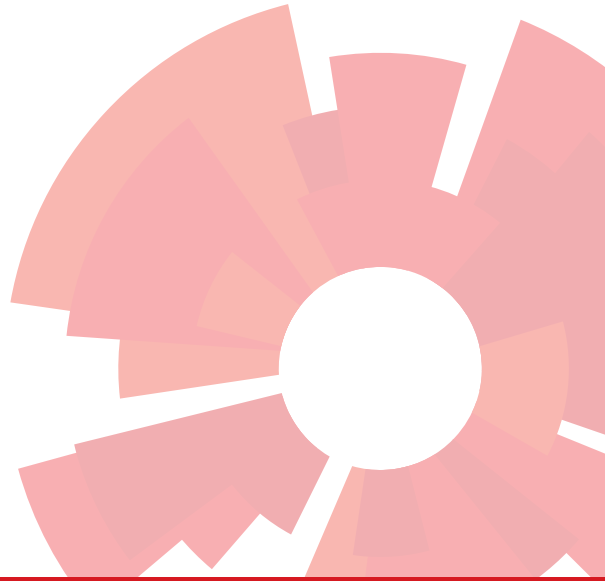
Hashing

- Maybe the most important reason for their existence
  - Tuples provide hashable lists
  - `hash`



Sequence unpacking

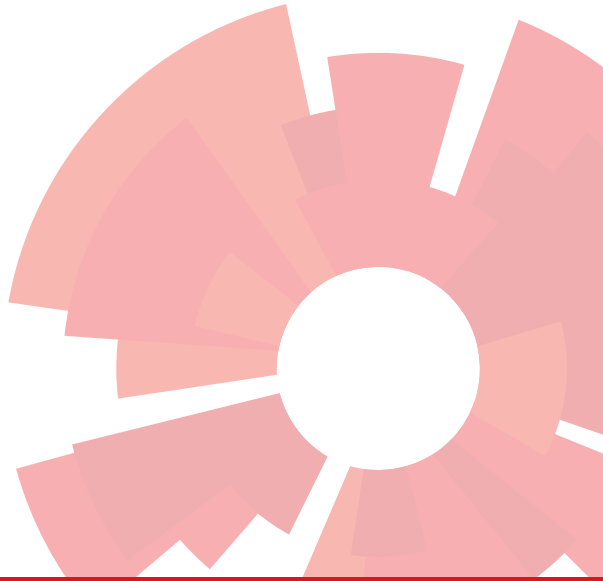
- Unpack sequence
- Unpack into sequence



Set

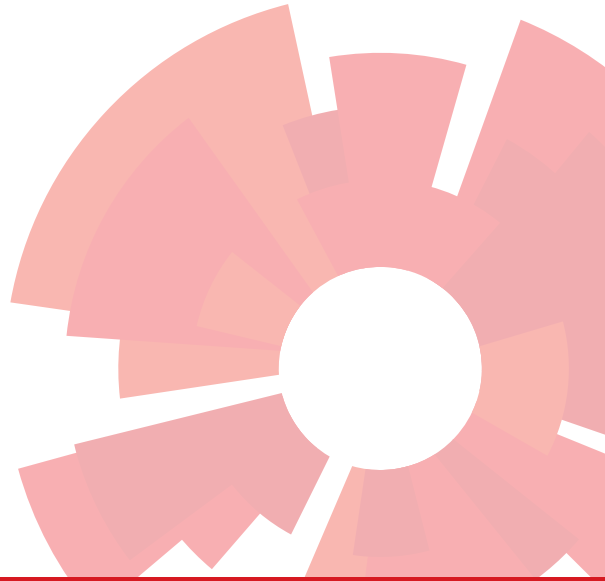
Sets

- Sets
  - Similar to a mathematical set
  - Collection of where we cant to check if an item is present
- *Hash* set
  - Constant time containment
  - Constant time add
  - Constant time remove



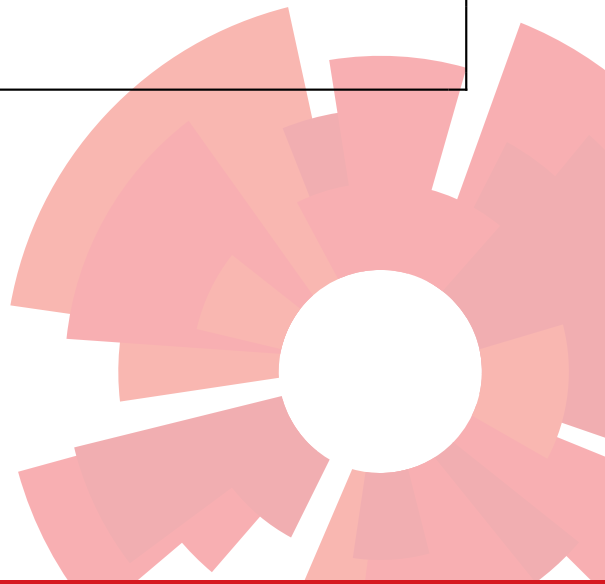
Set operations

- Builtin functions
  - `A | B`: Union ($A \cup B$)
  - `A & B`: Intersection ($A \cap B$)
  - `A ^ B`: Symmetric difference ($A \oplus B$)
  - `A - B`: Set minus ($A - B$)
  - `A < B`: Proper subset ($A \subset B$)
  - `A <= B`: Subset ($A \subseteq B$)
  - `A > B`: Proper superset ($A \supset B$)
  - `A >= B`: Superset ($A \supseteq B$)
  - `A == B`: Equality ($A == B$)



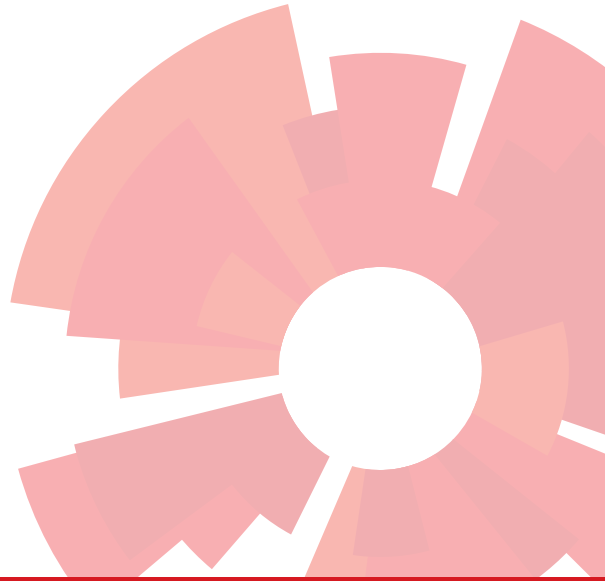
What's the difference?

```
>>> a = list(range(10_000_000))
>>> 877990 in a
True

>>> b = set(range(10_000_000))
>>> 877990 in b
True
```



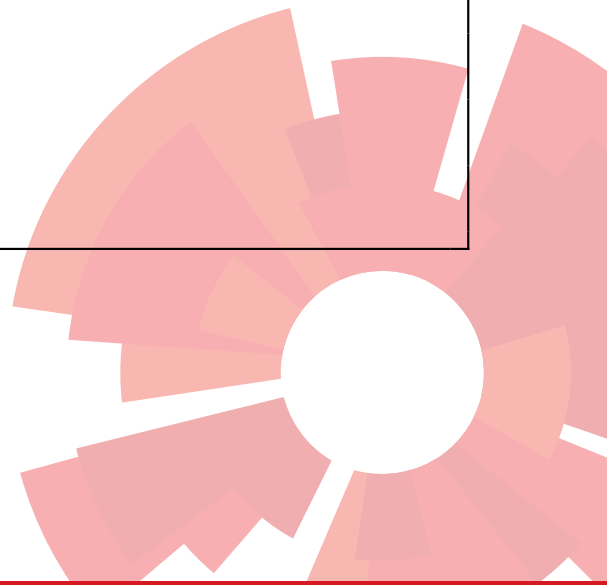
What can be in a set?

- Hash sets can only contain hashable types
  - tuples
  - strings
  - numbers (integers, floats, etc)



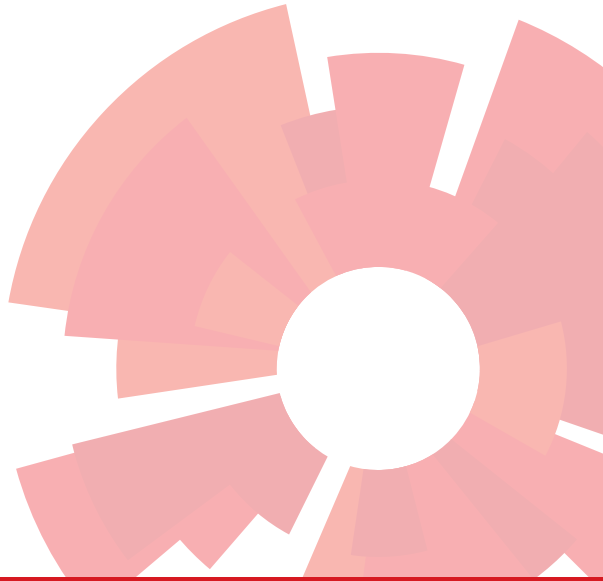
Set comprehension and iteration

```
>>> s = {i for i in range(5)}
>>> s
{0, 1, 2, 3, 4}

>>> for i in s:
...     print(i)
0
1
2
3
4
```



Hashable sets

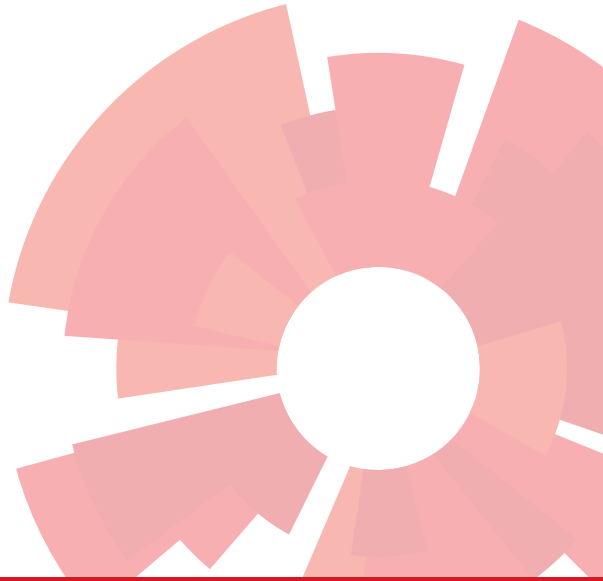
`frozenset` is a read only (and hashable) version of set



Strings

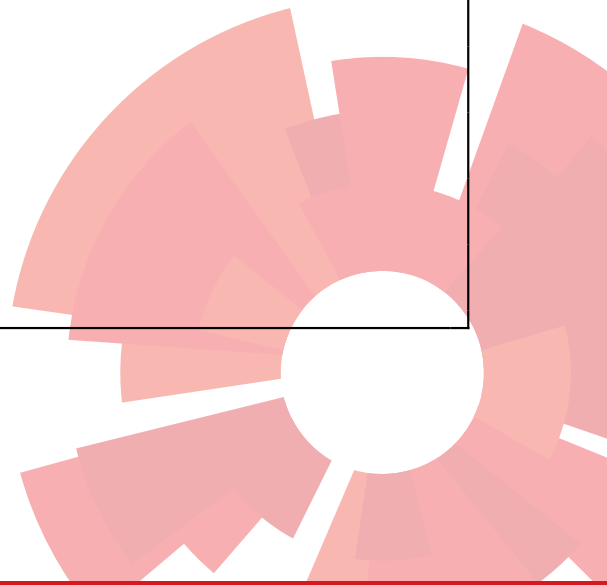
Common string functions

- `in` (substring check)
- Modification
  - `center, ljust, rjust`
  - `capitalize, title, swapcase, upper, lower`
- Checks
  - `endswith, startswith, isalnum, isdigit, is ...`
- Very important
  - `split, splitlines, join, strip`
- Other
  - `find, index`



String formatting

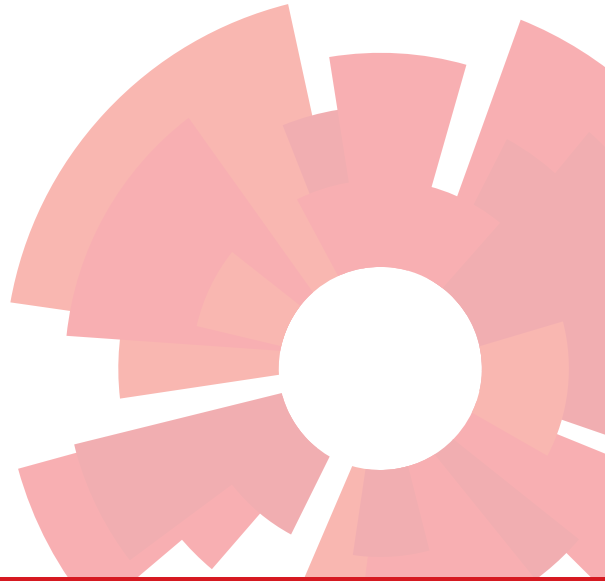
```
# Very old way
>>> '%s-%d' % ('test', 4)
'test-4'

# Old "new" way
>>> '{}-{}'.format('test', 4)
'test-4'

# Actual new way
>>> test = 'test'
>>> number = 4
>>> f'{test}-{number}'
'test-4'
```



Dictionaries

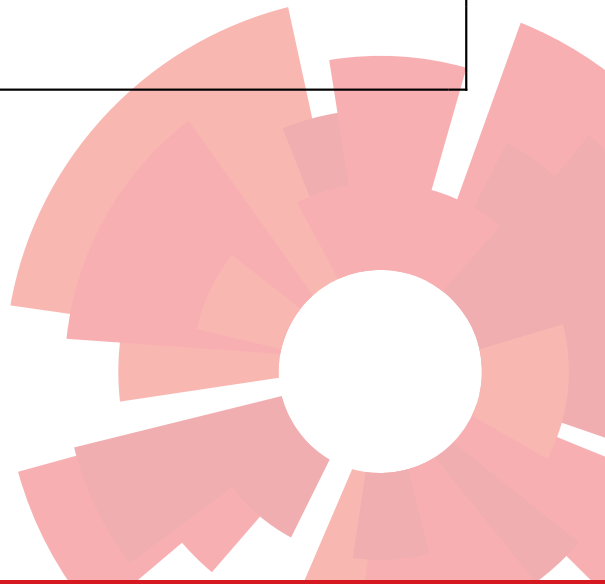
Implementation

- Store key-value pairs
  - keys are unique
- *Hash* map
  - Constant time lookup (value for key)
  - Constant time add
  - Constant time remove



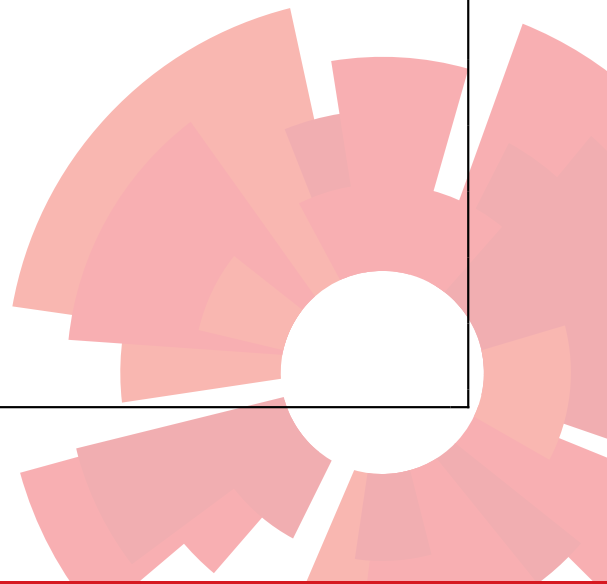
Syntax

```
>>> populations = {
...     'iceland': 300_000,
...     'italy': 60_000_000,
...     'china': 1_400_000_000,
...     'uk': 66_000_000,
...     'greece': 11_000_000,
... }
```



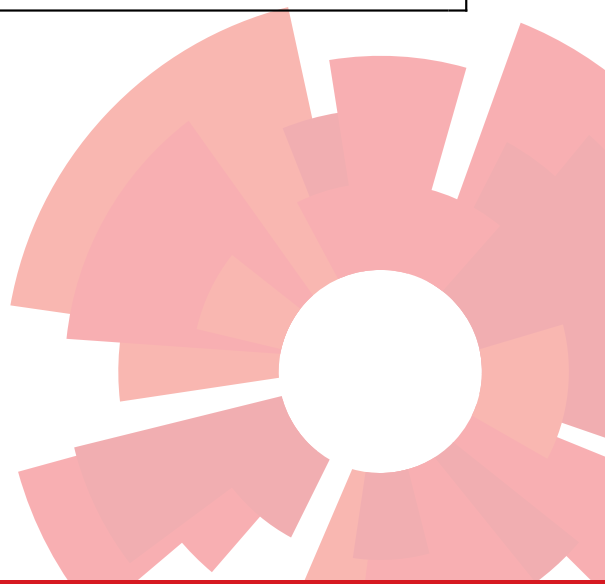
Iteration

```
>>> for k in populations:
...     print(k)
iceland
italy
china
uk
greece

>>> for k, v in populations.items():
...     print(k, v)
iceland 300000
italy 60000000
china 1400000000
uk 66000000
greece 11000000
```



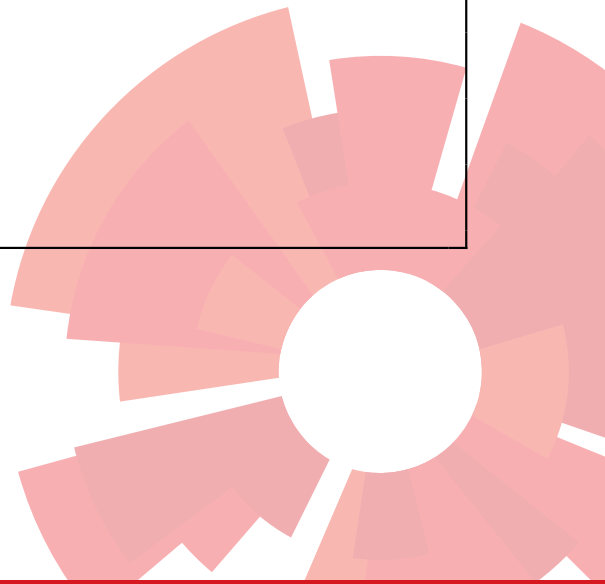
Constructor

```
>>> la = [1, 2, 3, 4, 5]
>>> lb = ['a', 'b', 'c', 'd', 'e']

>>> dict(zip(la, lb))
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
```



Dict comprehension

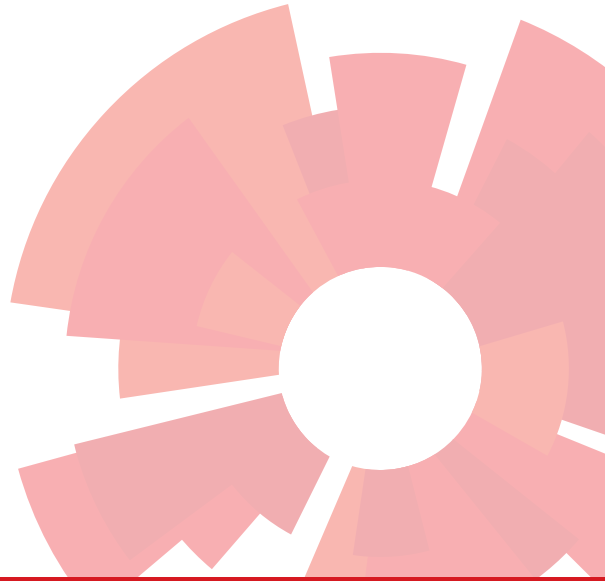
```
>>> { i: i**3 for i in range(10) }
{0: 0,
 1: 1,
 2: 8,
 3: 27,
 4: 64,
 5: 125,
 6: 216,
 7: 343,
 8: 512,
 9: 729}
```



Problems

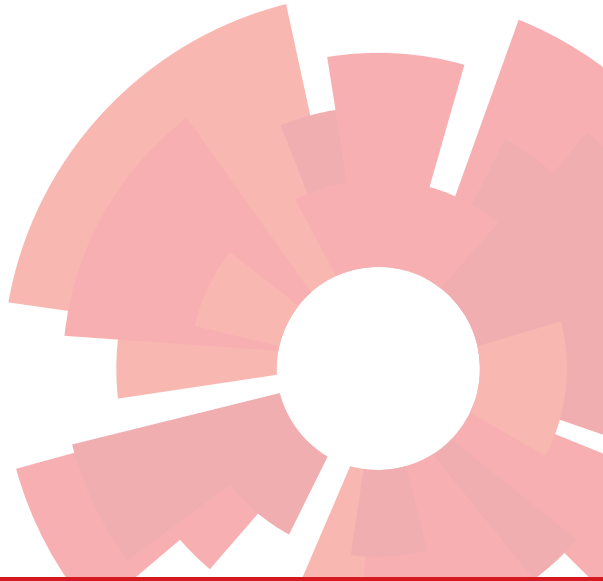
Problem 1

Find all countries that start with the letter D



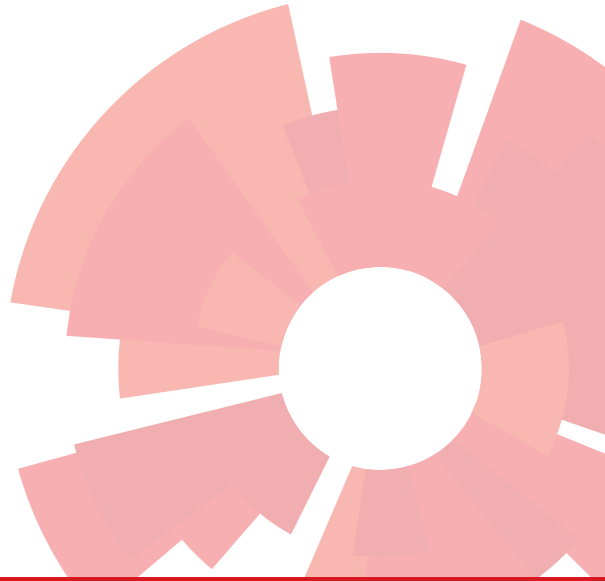
Problem 2

Find all countries with names longer than 10 letters



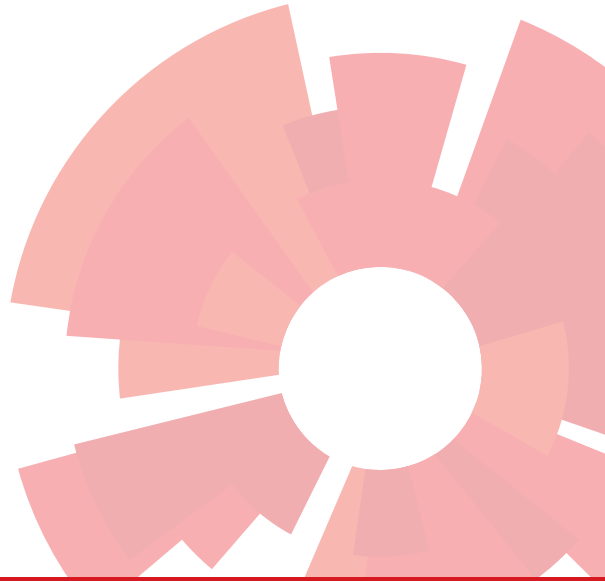
Problem 3

Find the country with the most unique letters



Problem 4

Find the country in Europe with the highest population



Problem 5

Group countries together by continent