Data processing

Fundamental programming blocks

TU Delft

Compound statements

Condition: If else/elif

- Syntax

```
if condition:
  statements
[elif:
  statements]
else:
  Statements
```

- Example

```
if a > 0:
  magnitude = a
else:
  magnitude = -a
```

Loop

- For
- While
- Zip, enumerate
- Some commonly-used keywords:
  - Continue
  - Break

While loop

- Use while loop to repeat doing something as long as a condition holds
- Syntax

```
while condition:
    statements
```

While loop

- <ipynb>

For loop

- Use for loop when iterat over a collection
- The number of iterations are known
- Syntax:

```
for val in object:

    statements
```

For loop

- List
- Tuple
- Str
- Range
- Dict

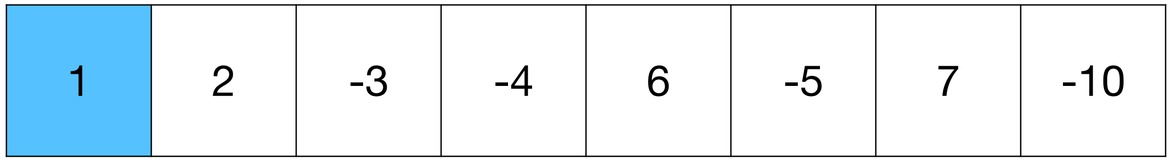
- <ipynb>
  - List
  - Tuple
  - Str
  - Range

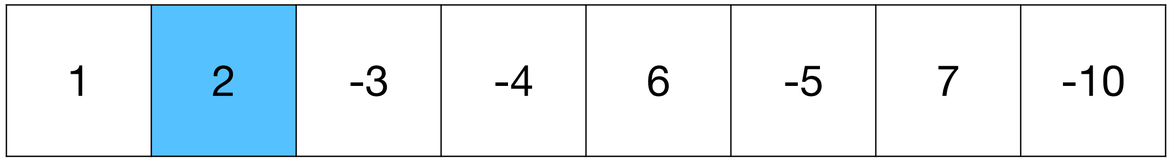
For loop

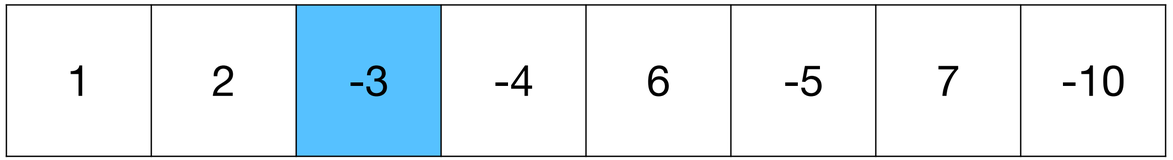
Dict

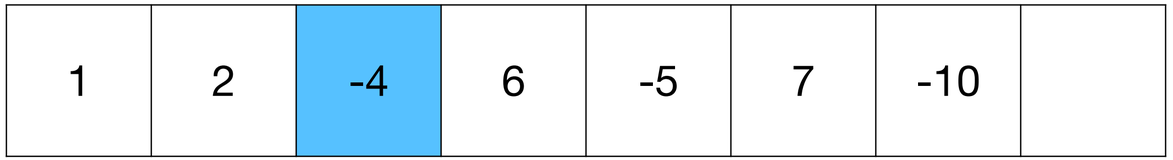
```
numbers = [1, 2, -3, -4, 6, -5, 7, -10]

for x in numbers:
   if x < 0:
      numbers.remove(x)

print(numbers)
```

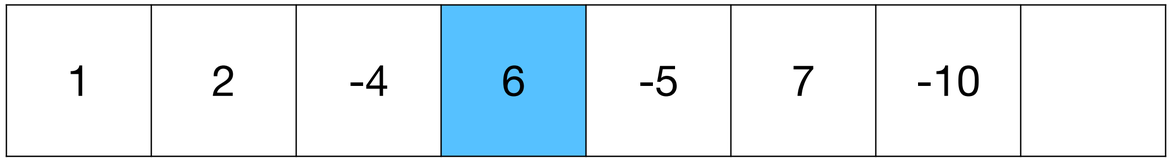
A	B	C
[1, 2, 6, 7]	[1, 2, -4, 6, 7]	Error

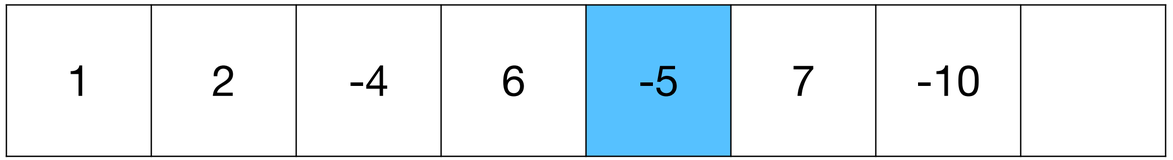

1
2
-3
-4
6
-5
7
-10

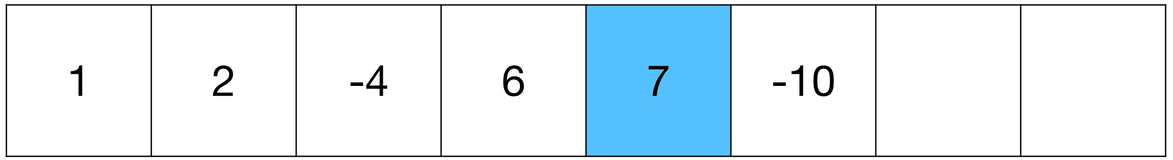

1
2
-3
-4
6
-5
7
-10

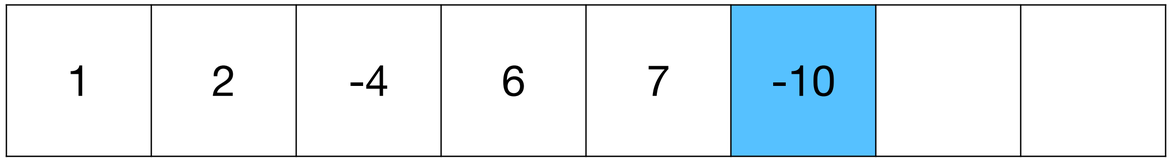

1
2
-3
-4
6
-5
7
-10

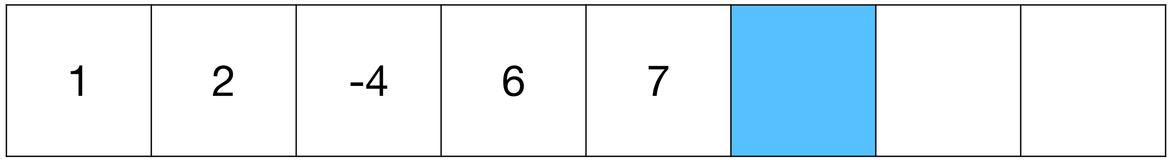

1
2
-4
6
-5
7
-10


1
2
-4
6
-5
7
-10


1
2
-4
6
-5
7
-10


1
2
-4
6
7
-10


1
2
-4
6
7
-10


1
2
-4
6
7

For loop

Dict

```
personal_info = {
   ‘name’: ‘Jone’,
   ‘age’: 25,
   ‘height’: 175,
   ‘weight’: 60
}

for element in personal_info:
   print(element)
```

A	B	C	D
{‘name’: ‘Jone’} {‘age’: 25} {‘height’: 175} {‘weight’: 60}	name age height weight	Jone 25 175 60	Error

For loop
Dict

- Three kinds of looping over a dict object

```
personal_info = {
   ‘name’: ‘Jone’,
   ‘age’: 25,
   ‘height’: 175,
   ‘weight’: 60
}

for key in personal_info:
   print(key)
```

```
personal_info = {
   ‘name’: ‘Jone’,
   ‘age’: 25,
   ‘height’: 175,
   ‘weight’: 60
}

for val in personal_info.values():
   print(val)
```

```
personal_info = {
   ‘name’: ‘Jone’,
   ‘age’: 25,
   ‘height’: 175,
   ‘weight’: 60
}

for key,value in personal_info.items():
   print(key)
   print(value)
```

Indexing in loop

- To update element while looping
- <ipynb>

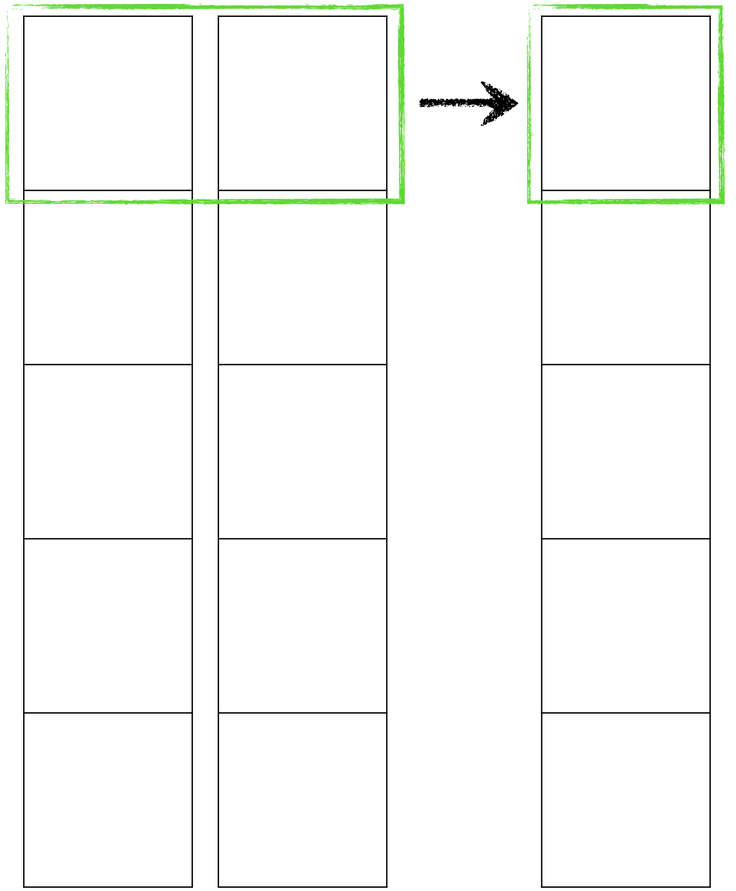
Enumerate

- Associate a count (index) with each element of iterable objects
- Use
-

```
for ind, val in enumerate(object):

    statements
```

Zip

- Use to associate two iterable objects value-by-value
- This makes concurrently looping through these objects more conveniently



Special keywords

Continue & Break

- Keyword continue is used in loop to ignore the rest of an iteration
  - Continue with the next iteration of the loop
- Keyword break is used to stop a loop

List comprehension

- <ipynb>

Function

Function

- To bind a procedure to a name for future references
- Syntax
-

```
def func_name(arguments):
    statements
```

Function

- Mandatory arguments
- Optional arguments

Function

Calling functions

- Provide mandatory arguments
- Arguments are specified by either
  - positions
  - names, using key=value

Function

<ipynb>

Function

*args, **kwargs

- *args:
  - Unlimited number of positional arguments
- **kwargs:
  - Unlimited list of key-value arguments

Class

Class

- Represent objects
  - Multiple individual objects
  - Attributes
  - Methods

Class inheritance

- A class can inherit attributes and methods another class
  - Parent/child class

Repository structure

Module

- Codes are separated into multiple files for easy management
- A module
  - One single file
  - Definitions and statements
- Definitions from one module can be imported into another
  - No redundancy

Module

my_functions.py

```python
def unravel_list(nested_list):
    flat_list = []
    for element in nested_list:
        if isinstance(element, list):
            flat_list += unravel_list(element)
        else:
            flat_list.append(element)
    return flat_list
```

main.py

```python
from my_functions import unravel_list

if __name__=='__main__':
    flat_list = unravel_list([
        1,
        [2, 3],
        'b',
        [
            'x', 'y', ['c', 2]
        ],
        5
    ])
    print(flat_list)
```

Import module

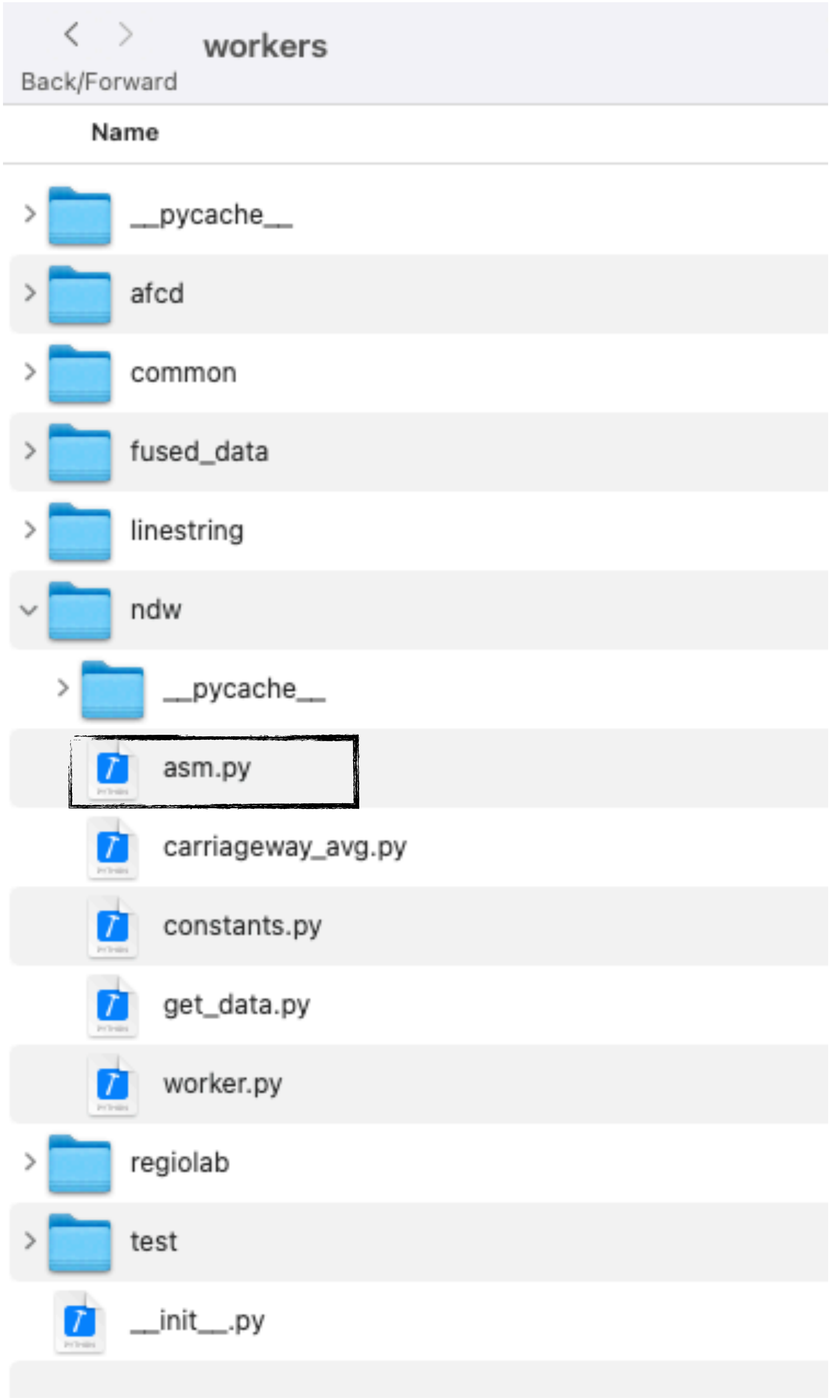
- When import a module
  - First, search built-in module
  - Then, files seen by sys.path

Package

- Collection of modules
- Can be organised as a hierarchical structure
- Use the __init__.py
  - A directory is a package

Package

- Use . to navigate sub-module
  - workers.ndw.asm



Intra-package reference

- A module can import from another module of sibling packages
- Absolute path
- Relative path
  - In the file worker/ndw/worker.py
  - from . import get_data
  - from ..regiolab import asm