Advanced Input Methods

Reading Data from Files

Advantages:

- separation between data and code
- faster replacement and sharing of data
- more flexible code
- clearer code

File Formats

- excel (pandas, xlrd)
- csv (csv)
- json (json)
- basically any text-file of some custom format (write parser)

The JSON File Format

```
{
    "oil_types":[
        "heavy","medium","light"
    ],
    "processes":[0,1],
    "production":{
        "heavy":[2,1],"medium":[2,2],"light":[1,4]
    },
    "demand":{
        "heavy":3,"medium":5,"light":4
    },
    "process_cost":[3,5]
}
```

Data Types

- Boolean
- Number (integer or floating point)
- String
- Array [] (ordered list of elements of arbitrary type)
- Object {} (unordered collection of name-value pairs)

Data Types

- Boolean
- Number (integer or floating point)
- String
- Array [] (ordered list of elements of arbitrary type)
- Object {} (unordered collection of name-value pairs)

Translates straight-forward to Python!

Reading JSON files in Python

```
import json

with open("data.json") as json_file:
data = json.load(json_file)
```

Read text files

- generally, input files not given as *json* or *csv* or *xslx*
- any custom format
- solution: read file line by line according to its syntax and create list/dictionary/object

Reading general textfiles in Python

```
filename = 'data.txt'
with open(filename, "r") as file:
    line = file.readline()
    while line:
        print(line.split("separator"))
```

Write text files

- generate different datasets to have stable collection of example data
- straightforward in python (very similar to *print*())

Writing textfiles in Python

```
filename = 'data.txt'
with open(filename, "w") as file:
    file.write("sometext\n")
    file.close()
```

Demo 3

Json