1 Reading and writing data

We use `open()` in Python to access **files** in the filesystem, to read text files, load and edit geometry files (e.g. in GeoJSON).

1.1 Accessing files – opening and closing

(C:/test.txt contains "text text text")

- loading file to variable:

```
aFile = open("C:/test.txt")
print aFile
# <open file 'C:/test.txt', mode 'r' at 0x7f4c34018270>
```

- okay, not really what we expect, reading content:

```
aFile = open("C:/test.txt")
content = aFile.read()
print content # text text text
```

- if you open a file, you should also **close** it!

```
aFile = open("C:/test.txt")
aFile.close()
content = aFile.read()
print content # ValueError: I/O operation on closed file
```

Closing prevents memory overflow and other issues.

1.2 Enconding

Usually you encounter Windows 1250 (that is cp-1250) encoding on Windows with Czech. Other frequently used encodings include *Windows 1252, ACII and Unicode (UTF-8).*
If text is loaded with a **wrong encoding**, things start to break – accented letters are displayed incorrectly and errors arise:

```
aFile = open("C:/test.txt", encoding="ascii")
print aFile.read()
# UnicodeDecodeError: 'ascii' codec can't decode byte 0x9a in position 2:
    ↪  ordinal not in range(128)

aFile = open("C:/test.txt", encoding="cp1250")
print aFile.read()
# ctštžcštýté=
```

1.3 Mode – reading, writing, appending

Character	Meaning
`"r"`	open for reading (default)
`"w"`	open for writing (erasing the file content first)
`"a"`	open for writing, appending to the end of the file
`"+"`	add to the previous three to allow reading and writing at the same time

- writing to a file:

```
aFile = open("C:/test.txt", mode="w")
aFile.write("this is a new text") # replaces "text text text"
aFile.seek(0) # see below
print aFile.read() # IOError: File not open for reading
```

- reading and writing:

```
aFile = open("C:/test.txt", mode="r+")
aFile.write("this is new once again") # replaces "this is a new text"
aFile.seek(0)
print aFile.read() # "this is new once again"
```

- appending to a file – adds to an existing file or creates a new one if it doesn't exist (instead of erasing the content as with write):

```
# write mode
for i in range(5):
    aFile = open("C:/test.txt", mode="w")
    aFile.write(str(i))
    aFile.close()
# "4" is the content of test.txt

# append mode
for i in range(5):
aFile = open("C:/test.txt", mode="a")
aFile.write(str(i))
aFile.close()
# "01234" is the content of test.txt
```

1.4 With statement – better way to open files

With closes the stream (file) automatically – much safer.

```
with open("test.txt", mode="r+") as aFile:
    aFile.write("text text")
print aFile.read() # ValueError: I/O operation on closed file → the file is
    ↪ now closed
```

1.5 Cursor position

Where we are in the stream (what position in the file).

- **tell()** – returns position

```
with open("test.txt", "r+") as aFile:
    print aFile.tell() # 0
    aFile.write("text") # writing changes position
    print aFile.tell() # 4
```

- **seek(offset, whence)** – changes cursor position to offset (whence can be 0,1,2 – 0 means from the beginning of the file (default), 1 means from the current position, 2 means from the end of the file)

```
with open("test.txt", "r+") as aFile:
    aFile.write("text") # writing changes position
    print aFile.read() # "" ← we are at the end of file, nothing to
        ↪ read!
    aFile.seek(0)
    print aFile.read() # "text"
    print aFile.tell() # 4
    aFile.seek(-2, 2) # two positions "to the left" from the end of
        ↪ the file
    print aFile.read() # "xt"
```

- **read(size)** – size limits the number of characters that are read

1.6 Lines

Creating a new line in **write()** – use **\n** for that.

```
with open("test.txt", 'w+') as aFile:
    for i in range(5):
        aFile.write('line' + str(i) + ': blablabla\n')

    aFile.seek(0)
    print aFile.read()
```

```
# line0: blablabla
# line1: blablabla
# line2: blablabla
# line3: blablabla
# line4: blablabla
```

Reading a file **line by line** is trivial:

```
with open("test.txt") as aFile:
    for line in aFile:
        print line
        # line0: blablabla
        # line1: blablabla
        # line2: blablabla
        # line3: blablabla
        # line4: blablabla
```

- **readline(limit)** – reads only until the end of a line (\n)

```
with open("test.txt") as aFile:
    print aFile.readline()
    # line0: blablabla
    print aFile.readline()
    # line1: blablabla
```

- **readlines(limit)** – read and *return* a list of lines

```
with open("test.txt") as aFile:
    print aFile.readlines()
    # ['line0: blablabla\n', 'line1: blablabla\n', 'line2:
        ↪  blablabla\n', 'line3: blablabla\n', 'line4: blablabla\n',
        ↪  'line5: blablabla\n', 'line6: blablabla\n', 'line7:
        ↪  blablabla\n', 'line8: blablabla\n', 'line9: blablabla\n']
```

2 Import

Import is easy, we use either:

- `import module`:

```
import random
print random.random()
```

- `from module import function1, function2, ...`:

```
from random import random, randint
print random(), randint(1,10)
```

or like this:

```
from random import *
# be very careful using this
```

We can also use this to give structure to our project. Let's save this code example as `lineTools.py`:

```
from random import random
def randomLine(length):
    line = []
    for i in range(length):
        line.append ([random() * 1000, random() * 1000])
    return line
def calculateLength(line):
    length = 0
    for i in range(len(line) - 1):
        length += ((line[i][0] - line[i+1][0]) ** 2 + (line[i][1] -
            ↪ line[i+1][1]) ** 2) ** 0.5
    return length
```

Now, you can import the content of this script to another script **saved in the same directory**: `script.py`

```
import lineTools
print lineTools.randomLine()
```

Important: In PyZo, you have to run the code **as a script**! That is, `Ctrl+Shift+E` instead of `Ctrl+E`.

2.1 Standard modules

There are some modules built directly into Python. These are some examples of useful **basic modules**. For a complete list, see `https://docs.python.org/2.7/library/`.

- **math** – mathematical functions
  `https://docs.python.org/2.7/library/math.html`

```
import math
circleRadius = 15
```

```
print math.pi * math.pow(circleRadius, 2)
print math.pi * circleRadius ** 2 # or like this
```

- **random** – pseudo-random number generators
  `https://docs.python.org/2.7/library/random.html`

```
import random
random.random() # random float 0..1
random.inform(1, 10) # random float 1.0..10.0
random.randrange(10) # integer 0..9
random.randrange(0,101,2) # even integer 0..100
random.choice("abcde") # random element from a string / list
```

Note

Pseudo-random? Why not random? See `https://www.random.org/`.
By the way, aside from atmospheric noise, people have used lava lamps and fish tanks to generate random numbers with webcams!

- **csv** – read and write tabular data in CSV format (you can save tables as CSV in Excel, LibreOffice, etc.)
  `https://docs.python.org/2.7/library/csv.html`

```
import csv
with open("users.csv", newline="") as f:
    reader = csv.reader(f, delimiter=":", quoting=csv.QUOTE_NONE)
    for row in reader:
        print row
```

- **urllib** – retrieve resources via URL
  `https://docs.python.org/2.7/library/urllib.html`

```
import urllib.request
with urllib.request.urlopen("http://python.org/") as response:
    html = response.read()
```

- **webbrowser** – an interface to display websites to users
  `https://docs.python.org/2.7/library/webbrowser.html`

```python
import webbrowser
webbrowser.open("http://python.org/", new=0, autoraise=True)
# new=0 to open in the same window if possible
# new=1 to open in a new window
# new=2 to open in a new tab
```

3 JSON, GeoJSON

3.1 JSON

JSON is a file format used frequently to send data over the internet (`https://en.wikipedia.org/wiki/JSON#Data_types.2C_syntax_and_example`). It uses **similar** data types as Python (numbers, strings, booleans, lists, dictionaries).

JSON example:

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
```

```
  "children": [],
  "spouse": null
}
```

In Python, the `json` module is useful for processing JSON data:

```
import json
data = {
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": True,
    "age": 25
}
jsonData = json.dumps(data) # convert to a string formatted as JSON
print jsonData
print json.loads(jsonData) # convert to data structure from JSON string
```

3.2 GeoJSON

GeoJSON format specifications: `https://tools.ietf.org/html/rfc7946`

Subset of JSON format used to handle spatial data, example:

```
{ "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [102.0, 0.5]
      },
      "properties": {
        "prop0": "value0"
      }
    },
    { "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]
        ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": 0.0
      }
    },
    { "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
          [100.0, 1.0], [100.0, 0.0] ]
        ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": {"this": "that"}
      }
    }
  ]
}
```

The root element is always a single GeoJSON object. It's type is one of the following:

- Point, MultiPoint
- LineString, MultiLineString
- Polygon, MultiPolygon
- GeometryCollection
- Feature
- FeatureCollection

Read through the specification and the examples at `https://en.wikipedia.org/wiki/GeoJSON`.

You can save data as GeoJSON from QGIS (right click on a layer → Save As) or using other tools (`https://mygeodata.cloud/converter/shp-to-geojson`).