OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

Übungen 11

Documentation

Links

- sequence operations
- mutable sequence operations
- string methods
- list methods
- dictionary methods

- NumPy Documentation

- Pseudo-random numbers
  - Python random module
  - NumPy Random Generator

Random Numbers

np.random.seed()

- Compare the output of the following scripts
  - use different cells in JupyterLab as indicated

- script 1:

```
[1]: import numpy as np
[2]: np.random.rand(3)
???
[3]: np.random.rand(3)
???
```

- script 2:

```
[4]: np.random.seed(1234)
     np.random.rand(3)
???
[6]: np.random.seed(1234)
     np.random.rand(3)
???
```

Random Numbers

np.random.seed()

- pseudo-random number generator (PRNG)
  - deterministic algorithm that generates numbers
  - numbers approximate properties of random numbers
  - sequence of numbers is determined by an initial value (seed)
- seed
  - number (vector) to initialize a pseudo-random number generator
  - completely determines number sequence generated by the PRNG
  - reinitialization with the same seed will produce the same number sequence
- purpose
  - enables us to replicate results and to compare different algorithms
- python

```
>>> np.random.seed(number)
>>> np.random.seed()                    # default seed uses system time
```

Random Numbers

np.random.default_rng()

- np.random.seed()
  - sets global random seed
  - affects all uses of the np.random module
  - fine for small projects (as ours)

- problem
  - larger projects with imports that could also set the seed
  - different seeds could result in non-reproducible results

- solution
  - use a local PRNG for your own code

Random Numbers

np.random.default_rng()

- Recommendation for larger projects
  - create one PRNG at the beginning of your script
  - use a seed if you want reproducibility
  - use the PRNG in the rest of your script

- Example

```
import numpy as np
seed = 12345                              # use your seed
rng = np.random.default_rng(seed)         # can be called without seed
rng.random(3)                             # generate random numbers with rng.function(args)
```

- Documentation [>]
  - simple random data
  - permutations
  - distributions (uniform, normal, ...)

Exercises

Fitting polynomials 1

- complete notebook `sample_ex.ipynb` as described in comments

- complete notebook `polyfit_ex.ipynb` as described in comments
  - fix (small) data set and observe errors and coefficients for least squares fits of various degrees
  - fix degree (large) and observe errors and coefficients for least squares fits on samples of varying size

- documentation
  - Polynomial [>]
  - Polynomial.fit [>]

Exercises

Fitting polynomials 2

- Generate
  - train set of size 20
  - validation set of size 10
  - test set of size 100

- select best best model with least squares fit on validation set
  - use degrees 1 .. 10
  - fit on train set
  - evaluate on validation set ()

- apply best fit on test set and compare error against error on validation set