1.7. Scope of Variables

Locale scope: variables inside a function can only be used in the function

You can can have functions in functions like

```
def myfunc():
    x = 300
    def myinnerfunc():
        print(x)
    myinnerfunc()
```

Global scope: Variables from outside of the function

```
X=300
def myfunc():
    print(x);
myfunc()
```

```
x=200; print(x)
```

inserting these two statements will cause an error to avoid it use

```
global x;
```

1.8. Expanding Python

Proverb: Don't reinvent the wheel

Example: We need π. Leibniz formula:

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$ or faster

```
import math
print(math.pi)
```

← library, contains also functions like sin, cos, ...

```
print ( math.sin (math.pi))
```

for more information check pydoc math

to spare the math. you can use

```
from math import pi, sin, cos      or
from math import *
```

10.9. Handy libraries

- numpy i.e. for matrix products (and much more)

```
import numpy as np
a = np.array ([[1,2],[3,4]])
b = np.array ([[5,6],[7,8]])
print ( np.matmul (a, b))
```

- SciPy for algorisms, like numerical integration, solving ODEs, Fourier transformation ...

- matplot for visualization

```
import matplotlib.pyplot as plt
import numpy as np
t = np.array (0.0, 2.0, 0.01)
s = 1 + np.sin ( 2 * np.pi * t )
fig, ax = plt.subplots ()
ax.plot (t, s)
ax.grid ()
plt.show()
```

2. Representation of numbers

Problem: continuous real numbers have to be discretized by

2.1. Fixed and floating point

(i) fixed point: integer, multiplied with a fixed scale factor b^E

consisting of a basis b and exponent E

example: $9.99 = 999 \cdot 10^{-2}$

properties:
- rounding errors
- overflows

} on small / large scales

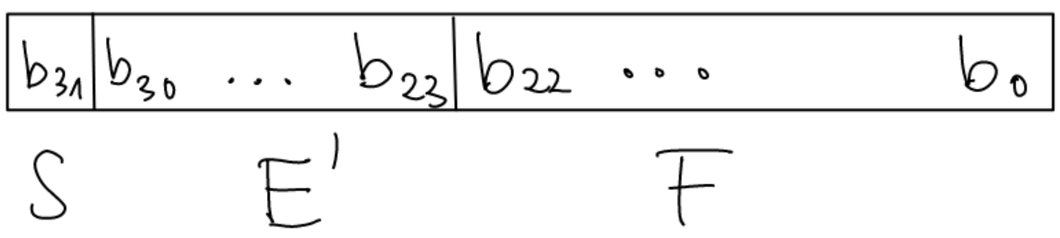
important for physics

(ii) floating point to cope with this problem represent numbers as

$$X = \underbrace{-}_{\text{sign } S} \quad \underbrace{0.1234567}_{\text{fraction } F} \cdot \underbrace{10^2}_{\text{scale factor } b^E}$$

$$X = S \cdot F \cdot b^E$$

assume we have 32 bits with $b = 2$ (IEEE 754)



$$X = (-1)^{b_{31}} \times 2^{(E'-127)} \times \left(1 + \sum_{i=1}^{23} b_{23-i}\, 2^{-i}\right)$$

regular range of exponent $-126 \leq E < 127$

E	F = 1	F ≠ 1
0	± 0	
255	± ∞	NaN = Not a Number

properties: • distance between neighbors increases with exponent

→ absolute rounding error $= |rd(x) - x| = 2^{E - \ell_F}$

($rd(x)$: rounding of x; ℓ_F: length of F in bits)

better use relative error:

$$\varepsilon = \frac{|rd(x) - x|}{x} \sim 2^{-\ell_F}$$

2.2. Errors

two major sources of numerical errors are

a) rounding with $rd(x) = x(1 + \varepsilon)$

(ε: required by discretization of x)

↳ Propagate, for example

$$f(x(1+\varepsilon)) \approx f(x) + f'(x) \times \varepsilon$$

$$\varepsilon_f = \left| \frac{f'(x)\, x}{f(x)} \right| \varepsilon_x$$

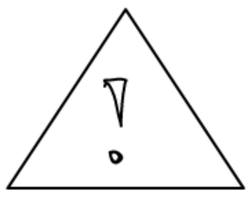
particularly bad when $f(x) \approx 0$

b) truncation iterations and recursions

example: $e^x \approx \sum_{n=0}^{N-1} \frac{1}{n!} x^n$

$$\text{error} = \sum_{n=N}^{\infty} \frac{1}{n!} X^n \leq \frac{1}{N!} X^N e^X$$

$\longrightarrow$ choose N appropriately



Numerical methods have to be stable with respect to these errors, implying that small, unavoidable numerical errors will not be amplified.