# Moduliai

# Moduliai

**Python** pats iš esmės neturi labai daug realizuotų funkcijų, galima sakyti, nedaug moka - su **Python** be matematinių gudrybių net šaknies ištraukt neišeina, nekalbant apie trigonometriją, *etc* ... 😎.
Visą **Python**'o funkcionalumą lemia iš esmės daugybė, daugybė, daugybė modulių (*modules*), dar kitaip vadinamų bibliotekų (*libraries*).
Laisvai prieinamų modulių rinkinys-svetainė: PyPi.

# pip, aplinkos

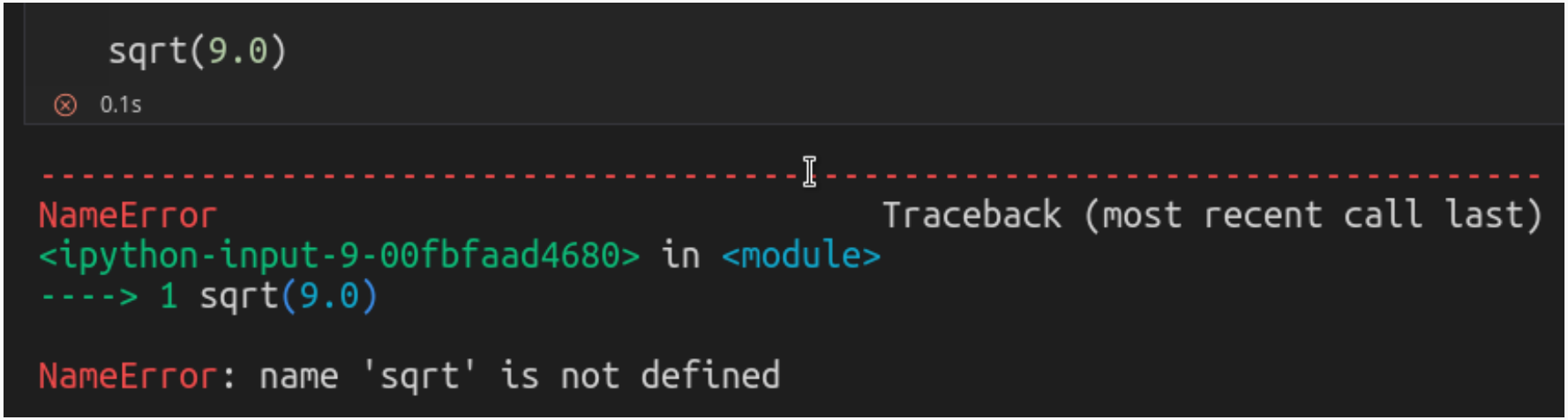
## Virtualios aplinkos

Virtuali aplinka - tai aplinka, kur Python bibliotekos yra atskirtos nuo sistemoje įdiegtų bibliotekų. Virtualių aplinkų galima turėti kiek norima. Tai reiškia, jog skirtingiems projektams galima naudoti skirtingas bibliotekų versijas, etc ...

## Bibliotekų /Modulių diegimas

```
pip install <library name>
pip install <library name>==version
```

# Moduliai ir jų importavimas




```python
import math # importuojamas visas modulis
math.sqrt(9.0) #pasiekiamos modulio funkcijos
from math import * #iš modulio importuojamos visos funkcijos
sqrt(9.0)
from math import sqrt #importuojama konkreti viena funkcija
sqrt(9.0)
```

# Uždaviniai

Susikurkite kelias skirtingas virtualias aplinkas, jose:

- Susidiekite šios modulius:
  *numpy*, *pandas*, *PyQt5*, *matplotlib*

Pašalinkite susikurtas virtualias aplinkas.

# numpy, random, dateutils

`random` - modulis, skirtas atsitiktinių skaičių generavimui.
Naudingos funkcijos:

```
import random
f = random.random() # ribos 0 - 1, float tipas
print(f)
i = random.randint(a, b) # vienas sveikasis skaičius iš intervalo a : b
print(i)
```

VILNIUS CODING SCHOOL

## Teksto konvertavimas į datą

```
from dateutil import parser  # konvertuoja tekstą į datas
td1 = "21-May-96"  #
data1 = parser.parse(td1).date()  # date() grąžina datos (date)
 ↪  objektą
td2 = "11-21-2022"
data2 = parser.parse(td2).date()
td3 = "2022-05-18"
data3 = parser.parse(td3).date()
td4 = "95-May-05"
data4 = parser.parse(td4).date()
```

## Atributai:

`.year` - grąžina metus

`.month` - grąžina mėnesį

`.day` - grąžina dieną.

```
print(data4)
print(data1.year)
print(data2.month)
print(data3.day)
```

`datetime` modulis skirtas dirbti su datomis, laikais.

```
import datetime
td0 = "21/May/00 17:45:07"
d0 = datetime.datetime.strptime(td0, "%d/%b/%y %H:%M:%S")
print(d0)
print(d0.time())
print(d0.hour)
print(d0.minute)
print(d0.date())
print(d0.year)
```

Plačiau čia apie pakaitos simbolius.

# Numpy modulis

`numpy` modulis yra skirtas dirbti su masyvais (*arrays*), stipriai optimizuotas. Numpy masyvai yra efektyvesni nei normalūs Python sąrašai, naudoja mažiau kompiuterio resursų.

## numpy importavimas

```
import numpy as np
#dabar visos numpy funkcijos yra pasiekiamos taip: np.funkcijos
 ↪  pavadinimas()
#pavyzdžiui, kubinės šaknies traukimas (cbrt() funkcija):
np.cbrt(64)
```

- `.linspace(start, stop)`
- `.asarray()`

### `.linspace()`

```
import numpy as np
print(np.linspace(0,10,10, endpoint=False))
```

$$step = \frac{STOP - START}{n-1}$$ - jei `endpoint=True`

$$step = \frac{STOP - START - 1}{n-1}$$ - jei `endpoint=False`

### `.asarray()`

```
lst = [1, 2, 3, 4, 5]
np_lst = np.asarray(lst)
print(type(np_lst))
```

- `.arange()`

```
array = np.arange(0, 100, 20) # skirtumas nuo linspace() -
 ↪  žingsnis, o ne taškų skaičius.
print(array)
```

**Masyvo generacija ir dauginimas iš skaičiaus**

```
array = (np.linspace(1,100,100, endpoint=True))*3.14
print(array)
```

`np.linspace(1,100,100, endpoint=True)` sugeneruoja masyvą, o visi sugeneruoto masyvo elementai padauginami iš $3.14$, lyg iš paprasto skaičiaus.

# Numpy kai kurios funkcijos

```
b = [3,4,7,8,9]
b_np = np.asarray(b)
np.average(b_np) # svorinis vidurkis (weighted)
np.mean(b_np) # aritmetinis vidurkis
np.min(b_np)
np.max(b_np)
np.abs(b_np) # modulis
np.cbrt() # kubinė šaknis
np.sqrt() # kv. šaknis
#trigonometrinės, eksponentinės ...
np.random.rand(shape) # 0-1 ribos
np.random.randint(low, high, size) #nurodytos ribos, kiekis
```

`.average()`: $A = \frac{\Sigma(a \cdot w)}{\Sigma w}$, čia $w$ svorių masyvas. Jei nenurodyta $w$, tai visi $w$ masyvo nariai laikomi lygūs $1$, ir tada $\Sigma w$ lygus $a$ masyvo elementų skaičiui, ir galutinis rezultatas gaunasi toks pats, kaip ir aritmetinio vidurkio.

# Užduotis

Parašyti skaičių spėliojimo žaidimą. Programa sugeneruoja skaičių nuo 1 iki 10 ir vartotojas spėlioja skaičius, programa pasako ar spėjimas mažesnis ar didesnis nei sugeneruotas skaičius. Programa baigia veikti, kai vartotojas atspėja skaičių. Naudojama `random` biblioteka.