Jazyky pre Dátovú Analytiku (JDA)

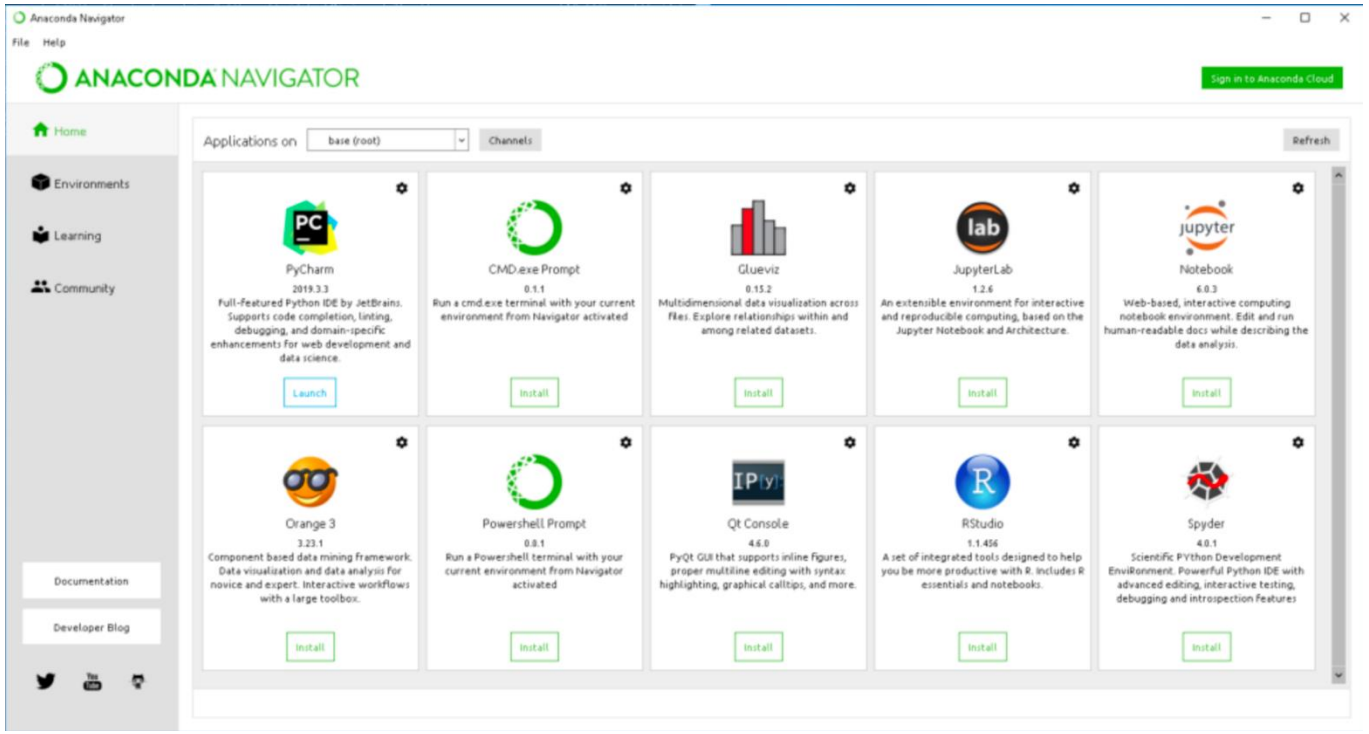
Prednáška č.4

Python

- Python – https://www.python.org/
  - Voľne dostupný programovací jazyk
  - Interpretovaný
  - Vysoko-úrovňový
- Stále populárnejší, s veľkým množstvom vývojárov a veľkým množstvom podporných balíkov
- Spolu s R sú najobľúbenejšími jazykmi pre analýzu dát
- Poskytuje takisto všetko pre dátovú analytiku
  - Prístup k dátam
  - Čistenie a transformácie dát
  - Analýzy
  - Reportovanie a vizualizácie
- Vývojové prostredie
  - Notebookovo orientované – Jupyter notebooky, JupyterLab
  - „Klasické“ IDE – PyCharm, Spyder, ...
- Rozsiahle balíky / distribúcie pre podporu (Python aj R) – často používaná v dátovej analytike – Anaconda (https://www.anaconda.com/)

Anaconda a rôzne IDE

- Poskytuje možnosť definovať a udržiavať si prostredie pre prácu, inštaláciu balíkov, nastavenia globálnych premenných prostredia, atď.
- Obsahuje aj rôzne typy rozhraní priamo v inštalácii (niektoré použiteľné pre Python aj R), viď. obrazovka Anaconda Navigator s rôznymi IDE pre inštaláciu / spustenie, ako Jupyter Notebook, JupyterLab, PyCharm, Spyder, Rstudio, ...
- Bolo by dobré aby ste si vyskúšali aj Notebookové rozhrania ako Jupyter, ale aj iné IDE ako napr. Spyder či PyCharm



Jupyter notebook / JupyterLab

- https://jupyter.org/
- Umožňuje vytvárať stránku popisujúcu jednotlivé analýzy s kódmi a ich evaluáciou (podobne ako Rmarkdown používa markdown na formátovanie textov) – veľmi dobré pre vysvetlenie postupov analýzy

jupyter Lecture-2B-Single-Atom-Lasing (unsaved changes)

File Edit View Insert Cell Kernel Help Python 3

Markdown CellToolbar

In [1]:
```
# setup the matplotlib graphics library and configure it to show
# figures inline in the notebook
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

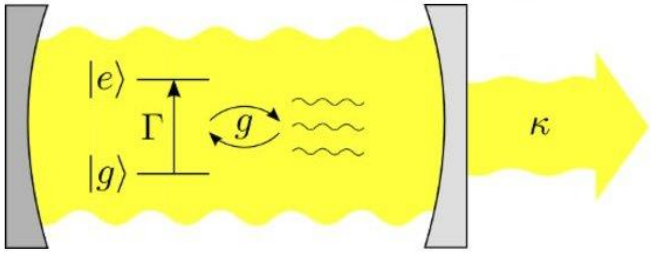
In [2]:
```
# make qutip available in the rest of the notebook
from qutip import *

from IPython.display import Image
```

Introduction and model

Consider a single atom coupled to a single cavity mode, as illustrated in the figure below. If there atom excitation rate Γ exceeds the relaxation rate, a population inversion can occur in the atom, and if coupled to the cavity the atom can then act as a photon pump on the cavity.

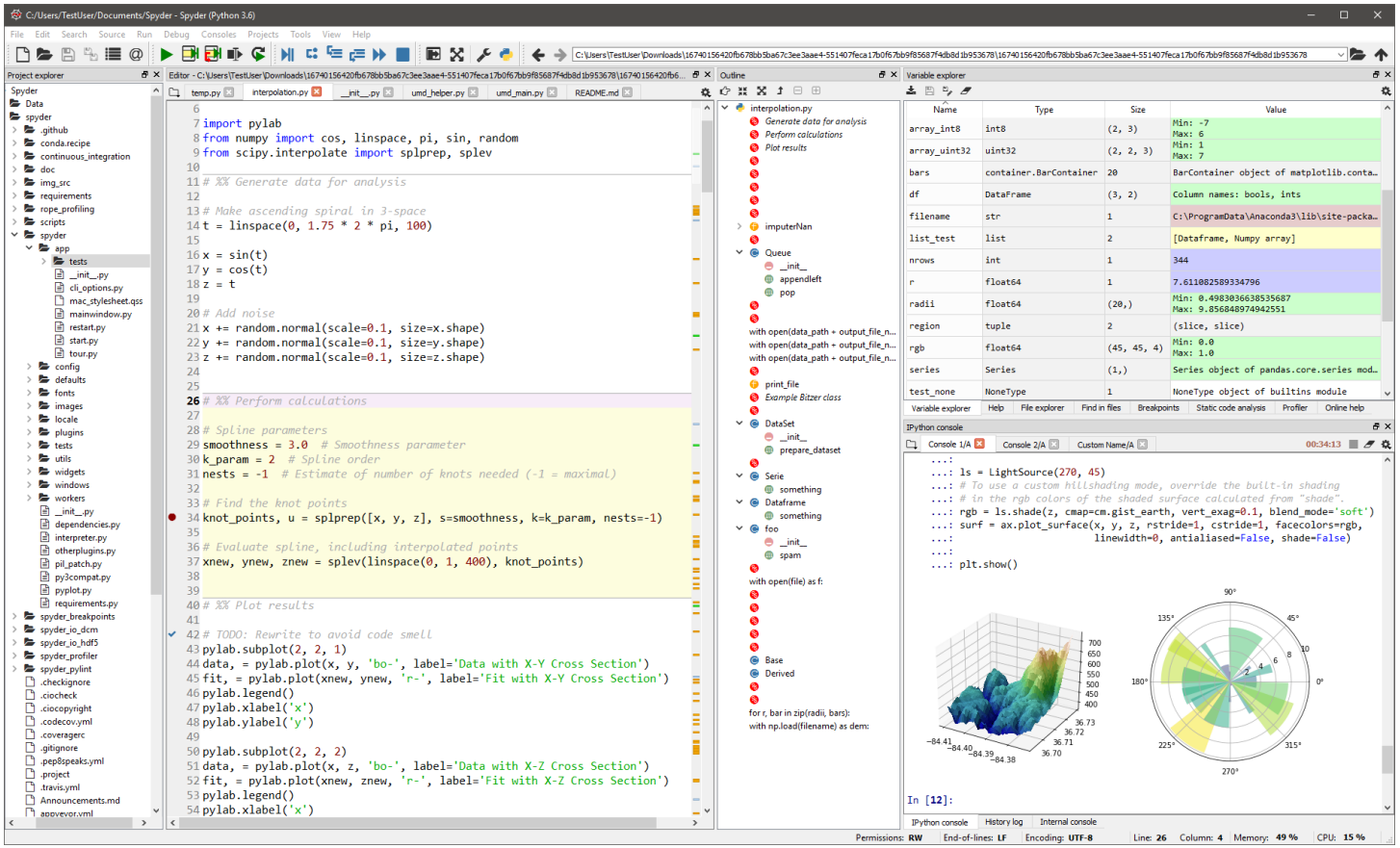
In [3]:
```
Image(filename='images/schematic-lasing-model.png')
```

Out[3]:



Spyder

- https://www.spyder-ide.org/
- Predstavuje bohatšie rozhranie IDE, trochu klasickejšie podané – príbuzné s Rstudiom či Matlabom



Python – konzola / interpreter

- Konzola / interpreter
  - Všetko čo zapíšeme do konzoly + ENTER … spustí vyhodnotenie a vygeneruje výstup
- Niekoľko jednoduchých prvkov a vlastností (totožné s R)
  - Premenné/priradzovací operátor: x = 8
  - Explicitný výpis: print(x) ..... vypíše 8
  - Existuje aj implicitný výpis: x ..... aj toto vypíše 8
  - Komentár: # toto je komentár
  - Reťazec: vždy v „“ ..... str = "hello"
  - Python takisto rozlišuje veľké a malé písmená (je case-sensitive) !

Dátové typy v jazyku Python

- Jednoduché
  - Znaky/Reťazce (**str**) ..... "hello world"
  - Numerické
    - Reálne hodnoty (**float**) 3
    - Celočíselné hodnoty (**int**) 1.3
    - Komplexné čísla (**complex**) ..... 3+2j
  - Logická/boolovská hodnota (**bool**) ..... True
  - Špeciálne binárne typy – bytes, bytearray, memoryview
- Zistenie typu (aj zložitejších, viď. další slajd)
  - Funkcia type() .... majme x = 3.5

```
type(x)
float
```

Dátové typy v jazyku Python (2)

- Zložené sekvenčné typy
  - Zoznam (**list**) ..... x = ["jano", "anna", "peter"]
  - N-tica (**tuple**) .... počet hodnôt je daný, ako pri vektore ..... x = (1.5, 2.3, 3.7)
  - Rozsahový typ (**range**) ..... x = range(4,7) ... obdoba m:n v R
- Množiny
  - Množina prvkov (obdoba listu, avšak nezáleží na poradí a nemôže obsahovať rovnaké prvky) .... (**set**) ..... x = {"a", "b", "c"}
  - Existuje aj typ **frozenset** pre množiny ktoré už nemôžeme meniť
- Mapovania / mapy
  - Mapa/Slovník (**dict**) .... množina párov kľúč:hodnota ..... x = {"meno" : "Martin", "age" : 25}

Stanovenie alebo úprava typu

- Pre vynútenie typu môžeme použiť konštruktor, napr.
  - int(), float(), str()
  - Napríklad pri funkcii int()
    - Ak vložíme celé číslo, výsledok bude toto číslo
    - Ak vložíme reálne číslo, výsledok úpravy bude iba celá časť čísla
    - Ak vložíme reťazec ktoré má vo vnútri celé číslo, výsledok je toto číslo
- Ak vstup do funkcie nie je presne literál toho typu, konštruktor sa snaží určiť hodnotu (a teda vykoná implicitnú úpravu)
- Môžeme si takto pomôť aj pre explicitnú úpravu hodnôt – čiže obdobne ako v R tieto metódy predstavujú explicitnú alebo implicitnú úpravu typov

Operátory pre prácu s premennými/hodnotami

- Veľa z nich je totožných ako v C, Java, R, ...
  - Aritmetické .... ako + - / * % ** //
    - % - modulo (zvyšok po delení) ... 5 % 2 dáva výsledok 1
    - ** - exponenciálna funkcia .... napr. 2^5 .... 2 ** 5 dáva výsledok 32
    - // - výsledok delenia zaokrúhlenie .... 9 // 4 dáva výsledok 2
  - Priradzovacie .... = += -= *= .... a ďalšie
  - Porovnávacie .... == != > < >= <=
  - Logické (pre kombináciu viacerých podmienok).... and or not

    (x <=5 and x > 3) or (x >10 and x <=15)

    not (x == 7 and y >= 10)
  - Zistenie identity objektov .... is is not
  - Zistenie výskytu v množine ..... in in not
  - Binárne .... & | .... a ďalšie

Výber častí zoznamov / indexovanie

- Prístup k elementom zoznamu / n-tice / ...
  - Jednotlivé prvky …. [index prvku]
    - V python je indexovanie na začiatku zoznamu od 0 !
    - z = [3.5, 1.2, 2.3, 4.5, 1.7] …. z[0] vráti 3.5, z[1] vráti 1.2, ...
    - Indexovanie môže byť negatívne – znamená to index od konca, pričom index -1 je posledný prvok, -2 predposledný, atď. .... z[-1] vráti 1.7
  - Viacero prvkov
    - Časť vyberieme pomocou [**od**:**do**], pričom **od** je vrátane, ale **do** bez tohto prvku, t.j. z[1:4] vráti [1.2, 2.3, 4.5]
    - Aj tu môžeme použiť negatívne indexovanie
  - Viacero prvkov s vynechaním jedného indexu
    - Keď vynecháme „od“ alebo „do“, daný index sa uvažuje buď ako začiatok alebo koniec zoznamu
    - Napríklad z[:3] vráti všetky prvky od začiatku po prvok s indexom 3 (avšak bez prvku s daným indexom), čiže [3.5, 1.2, 2.3]
    - Potom zase z[3:] vráti všetky prvky od indexu 3 po koniec (vrátane prvku s daným indexom), čiže [4.5, 1.7]

Kontrola behu programu

- Beh programu je možné kontrolovať pomocou nasledujúcich štruktúr :
  - **if, elif, else**: testovacia podmienka

```
if x > y:
    print("x je väčšie ako y")
elif x == y:
    print("x je rovnaké ako y")
else:
    print("x je menšie ako y")
```

  - **while**: cyklus bežiaci kým je platná podmienka

```
i = 1
while i < 10:
  print(i)
  i += 1
```

  - **for:** cyklus cez zoznam elementov alebo pre daný rozsah – používa sa in operátor (ako v R), range sa používa na vytvorenie rozsahu pre vopred definovaný počet cyklov

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

```
for x in range(6):
    print(x)
```

  - **Zmeny chovania v cykle**
    - **break**: ukončuje beh cyklu (rovnako ako break v C, Java)
    - **continue**: preskočí interakciu v cykle (rovnako ako continue v C, Java)
    - **else -** nie v cykle samotnom - ak použijeme na konci po cykle, môžeme tak určiť čo sa má udiať ak prejdeme všetky kroky cyklu (nevykoná sa napríklad ak sme z cyklu vypadli cez break)

- elif – ďalšia podmienka pre tie prípady, ktoré nesplnili predchádzajúce podmienky (else if)
- else – časť pre všetky ostatné prípady
- Jednoduchšie podmienky samozrejme môžeme kombinovať pomocou and, or, not

Môžeme použiť špecifickejší rozsah od:do, či pridať aj veľkosť kroku podobne ako v ďalších jazykoch range(2,6) alebo range(2,6,3)

Častý rozdiel pri volaní funkcií s objektmi v Python vs R

- Veľa funkcionalít v Pythone je riešených volaním funkcie daného objektu cez bodkovú notáciu (ako klasickej metódy objektu pri OOP)

  objekt.funkcia(parametre)

- V R sa používa výsledkovo ekvivalentná funkcia (nemusí mať rovnaký názov) ktorá má väčšinou ako prvý parameter daný objekt a následne ďalšie parametre

  funkcia(objekt, parametre)

- Príklad: majme zoznam x (v R aj Python), chceme vložiť nový prvok na koniec zoznamu

R:

```
> x = list(1,2,3,4)
> append(x, 8)
```

Python:

```
x = [1, 2, 3, 4]
x.append(8)
```

Reťazce

- Reťazcové vstupy sú rovnako ako v R buď v jednoduchých alebo dvojitých úvodzovkách
  - t.j. 'hello world' je to isté ako "hello world"
- Je možné špecifikovať viacriadkový reťazec použitím troch úvodzoviek na začiatku a na konci (platí pre obidva typy) .... potom aj výpis je viacriadkový

```
a = '''Toto je retazec formatovaný ako
viacriadkovy s pouzitim trojice
jednoduchych uvodzoviek.'''
print(a)
```

Výpis:

```
Toto je retazec formatovaný ako
viacriadkovy s pouzitim trojice
jednoduchych uvodzoviek.
```

Reťazce (2)

- Reťazce sú tiež polia (zoznamy) znakov

  x = "Hello, World!" ... print(b[2:5]) ... Výstup:

  llo

- Dĺžka reťazca – len(x)
- Formátovanie výstupov premenných spolu s reťazcami
  - format() funkcia

  pocet = 53

  suma = 1253.5

  txt = "Dnes sme predali {} výrobkov v celkovej sume {} EUR"

  print(txt.format(pocet,suma))

  Výstup:

  Dnes sme predali 53 výrobkov v celkovej sume 1253.5 EUR

Metódy spracovania reťazcov

- Reťazce je možné jednoducho upravovať pomocou rôznych funkcií, viď. niektoré vybrané nižšie
  - Volanie: ak x je reťazec x.nazov_funkcie(parametre_funkcie)
- Konverzie
  - lower / upper – konvertuje reťazec na malé / veľké písmená
  - capitalize / casefold – prvé písmeno konvertuje na veľké / malé
  - title – prvé písmená každého slova bude veľké
  - strip – vráti orezaný reťazec (žiadne medzery na začiatku ani na konci reťazca)
- Vyhľadávanie a nahrádzanie
  - count – vráti počet výskytov podreťazca alebo znaku (vzoru) v celom reťazci
  - find, index – vrátia pozíciu výskytu vzoru v celom reťazci
  - startswith, endswith – vráti True ak sa reťazec začína / končí daným vzorom
  - replace – nájde vzor v reťazci a nahradí ho iným definovaným používateľom
- Rozdeľovanie a spájanie
  - split – rozdelí reťazec použitím definovaného separátora a vráti zoznam podreťazcov
  - splitlines – rozdelí reťazec podľa ukončení riadkov (\n) a vráti zoznam jednotlivých riadkov
  - join – spájanie reťazcov
- Overovacie funkcie - is* funkcie
  - isnumeric – vráti True ak ide o numerickú hodnotu
  - islower – vráti True ak sú všetky znaky malé písmená
  - ….

```
x = "Hello, World!"
print(x.lower())
```

Výpis:

```
hello, world!
```

Metódy pre prácu s kolekciami

- Niektoré funkcie pre prácu s kolekciami (zoznamy, tuples, ...) už boli uvedené
  - Indexovanie a výber podčastí, použitie in napríklad pre cyklus for, append na pridanie prvku na koniec zoznamu , len(x) pre veľkosť kolekcie
- Avšak existujú aj ďalšie funkcie pre kolekcie (niektoré špecifické pre zoznam/tuple)
  - Ak chceme zmeniť hodnotu, môžeme vybrať prvok a priradiť mu novú hodnotu
  - insert ... vloženie prvku na konkrétne miesto
  - remove ... odstránenie prvku jeho špecifikáciou
  - pop .... odstránenie prvku na pozícii (posledný ak nie je špecifikované)
  - del .... ide o kľúčové slovo realizujúce to isté čo pop ak dáme index (del mylist[2]) alebo zmaže kolekciu ak nie (del mylist)
  - clear … vyčistí (vyprázdni) kolekciu
  - copy ... kopírovanie zoznamov, môžeme aj cez list() funkciu
  - Spojenie zoznamov je možné cez +, použitím append v cykle pre druhý zoznam, alebo použitím metódy extent() .... list1.extend(list2)
  - reverse – otočí zoznam Reverses the order of the list
  - sort – usporiada zoznam

Metódy pre prácu s kolekciami (2)

- Množiny – nesmú sa opakovať prvky, nezáleží na poradí
  - Pridávanie / odoberanie
    - add – pridanie jedného prvku do množiny
    - update – pridanie viacerých prvkov do množiny
    - remove / discard – obidve sa snažia odstrániť prvok, remove však vyhlási error ak taký prvok neexistuje
  - Množinové operácie s ďalšou množinou
    - union (zjednotenie), intersection (prienik), difference (rozdiel), ...
  - Overovacie voči ďalšej množine
    - isdisjoint, issubset, issuperset … napr. x.issubset(y) je True ak x je podmnožinou y

Metódy pre prácu s kolekciami (3)

```
x = {
    "meno": "Jan",
    "priezvisko": "Hrasko",
    "vek": 32
}
```

- Mapa/Slovník (dict)
  - Prístup k prvku aj cez kľúč, priamo alebo cez metódu get .... x["meno"] … x.get("meno")
  - update – slúži na vloženie nového páru kľúč,hodnota
  - keys – vráti všetky kľúče
  - values – vráti všetky hodnoty ako zoznam
  - items – vráti páry kľúč,hodnota ako jednotlivé n-tice (tuple)
  - fromkeys – vráti podmnožinu slovníka pre definované kľúče
  - pop / popitem – pre odstránenie špecifického páru / posledného páru
- Môžeme vytvárať aj vnorené slovníky – ľahko dosiahneme štruktúru podobnú JSON-u
- Mapa / slovník môže reprezentovať jednoduchý dataframe – kľúč je názov atribútu, hodnota je tuple/zoznam reprezentujúci stĺpec hodnôt daného atribútu (aj keď v praxi sa používajú rozšírenia, ako dataframe knižnice pandas)

Dátumy / časy

- datetime modul (balík)

```
import datetime
x = datetime.datetime.now()
print(x.year) # vypise rok 2020
print(x.strftime("%A")) # vypise den tyzdna celym slovom,ako Tuesday
```

- Vytvorenie dátumu / dátumu a konkrétneho času

```
y = datetime.datetime(2020, 3, 17)
2020-03-16 00:00:00
z = datetime.datetime(2020, 3, 20, 7, 10, 5)
2020-03-20 07:10:05
```

- strftime – formátuje výstup
  - Formátovanie má veľa možností, používaných podobným spôsobom ako v R či iných jazykoch
  - https://strftime.org/

Code	Meaning	Example
%a	Weekday as locale's abbreviated name.	Mon
%A	Weekday as locale's full name.	Monday
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	1
%d	Day of the month as a zero-padded decimal number.	30
%-d	Day of the month as a decimal number. (Platform specific)	30
%b	Month as locale's abbreviated name.	Sep
%B	Month as locale's full name.	September
%m	Month as a zero-padded decimal number.	09
%-m	Month as a decimal number. (Platform specific)	9
%y	Year without century as a zero-padded decimal number.	13
%Y	Year with century as a decimal number.	2013
%H	Hour (24-hour clock) as a zero-padded decimal number.	07
%-H	Hour (24-hour clock) as a decimal number. (Platform specific)	7
%I	Hour (12-hour clock) as a zero-padded decimal number.	07
%-I	Hour (12-hour clock) as a decimal number. (Platform specific)	7
%p	Locale's equivalent of either AM or PM.	AM
%M	Minute as a zero-padded decimal number.	06
%-M	Minute as a decimal number. (Platform specific)	6
%S	Second as a zero-padded decimal number.	05
%-S	Second as a decimal number. (Platform specific)	5
%f	Microsecond as a decimal number, zero-padded on the left.	000000
%z	UTC offset in the form +HHMM or -HHMM (empty string if the	

Vektory / Matice - numpy

- Základ vektorov a matíc je riešený rozšírením základných kolekcií cez knižnicu numpy
  - Používa sa často ako základ v ďalších rozšíreniach pre spracovania dát (pandas, scikit, ...)
- Numpy array (import numpy as np)
  - Základ je numpy array, ktorý môže byť 1-D (vektor) alebo 2-D (matica)
  - Môže sa vytvoriť podobne ako zoznam

    x = np.array([1, 2, 3])

    y = np.array([[1,2,3],[4,5,6]])
- Základné vlastnosti
  - Shape ... Tvar / Veľkosť matice (vektora)
    - print(x.shape()) .... Výstup: (3,)
    - print(y.shape()) .... Výstupdáva (2,3)
  - Prístup k prvkom – súradnice v rámci poľa, podobne ako pri zoznamoch, akurát s možnosťou ďalšieho rozmeru
    - Rovnako funguje väčšina súvisiacich operácii, ako napríklad výber podčastí, indexovanie, ...
    - Môžeme použiť aj výber prvkov pomocou nejakej podmienky – print(y > 2) vypíše True/False podľa toho či element spĺňa podmienku – má hodnotu > 2

Vektory / Matice - numpy (2)

- Podobne ako v R je možné robiť množstvo operácií, ktoré sa vykonávajú vektorovo
  - Matematické vektorové / maticové operácie
    - Operácie sčítania (np.add), odčítania (np.subtract), násobenia po elementoch (np.multiply), delenia po elementoch (np.divide), ...
    - Vektorové/Maticové násobenie (np.dot)
    - Transponovanie matice x .... x.T
  - Zmeny tvaru

    reshape metóda .... arr = np.array([1, 2, 3, 4, 5, 6]) .... newarr = arr.reshape(3, 2)
  - Spájanie / rozdeľovanie numpy polí
    - np.concatenate pre join dvoch polí, pre obdobu rbind/cbind z R existujú np.hstack a np.vstack funkcie
    - Rozdelenie polí – funkcia np.array_split
  - Vyhľadávanie, usporiadavanie, filtrovanie polí
    - Metódy ako np.where, np.sort, ...

Funkcie

- Funkcie v Pythone sa
  - tvoria sa pomocou kľúčového slova **def**
  - môžu byť rekurzívne
  - ako argument môžeme použiť list
  - môžu mať default hodnotu argumentov (tak ako v R)

```
def nasobenie(a, b=1):
        return a*b
print(nasobenie(1,3))   #.... Výstup: 3
print(nasobenie(8,2))   # .... Výstup: 16
print(nasobenie(2))  # .... Výstup: 2 ... Použitá default hodnota
```

- Takisto ako v R vieme použiť pomenované argumenty, čiže zadávať argumenty ako dvojicu názov_argumentu = hodnota

```
def exp(x, y):
        return x**y

print(exp(y = 3, x = 2))  # ..... Výstup: 8
```

Funkcie (2)

- Existuje obdoba „...“ argumentu z R – ak použijeme * pred názvom argumentu, t.j. že môžem funkciu definovať pre ľubovoľný počet argumentov

```
def mojafunkcia(*argumenty):
    str = ""
    for arg in argumenty:
        str += arg
    print(str)

mojafunkcia('Jeden', 'dlhy', 'spojeny', 'retazec','bez','separatoru’)
```

Výstup z volania funkcie:

```
Jedendlhyspojenyretazecbezseparatoru
```