Umelá inteligencia

Návody na vybrané cvičenia

.fuzzy regulátory
riadenie 2D autíčka
.neurónové siete
klasifikácia, detekcia spam-u
regresia, riadenie 2D autíčka
.analýza dát
nástroj RapidMiner Studio
príklady v jazyku Python
.jazyky UI
úvod do jazyka Python
.ďalšie
hranie hier
programovanie ohraničení
cheatsheet pre Python

s príkladmi pre keras, tensorflow

SKRIPTÁ

Michal Gregor
Marián Hruboš
Dušan Nemec

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. APVV-16-0006.

Umelá inteligencia, skriptá I
Návody na vybrané cvičenia
Vysokoškolské skriptá

Recenzenti: prof. Ing. Aleš Janota, PhD., EurIng.
Ing. Milan Botka, PhD.

Pre Žilinskú univerzitu v Žiline vydal:
CEIT, a.s., Univerzitná 8661/6A, 010 08 Žilina.

ISBN 978-80-89865-02-4

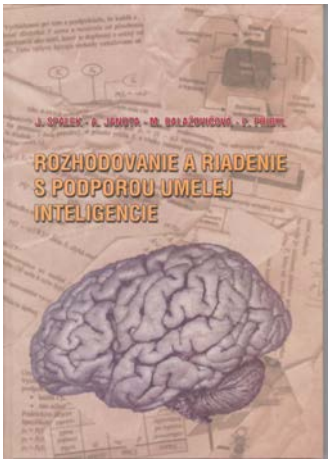
PREDSLOV

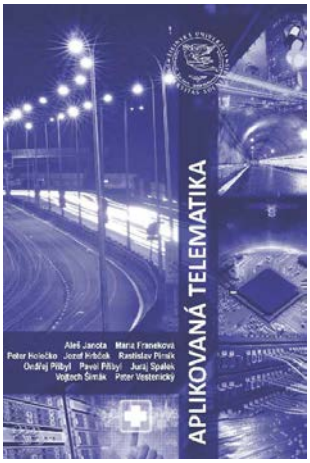
Katedra riadiacich a informačných systémov má mnohoročnú tradíciu vo výučbe metód umelej inteligencie a strojového učenia – či už ide o oblasť fuzzy systémov, umelých neurónových sietí, expertných systémov, základov analýzy dát alebo o iné príbuzné oblasti. Týmto témam sa historicky venovalo a i dnes sa venuje pomerne veľké množstvo predmetov. Môžeme medzi nimi spomenúť najmä predmety:

- Umelá inteligencia 1 (pôvodne Umelá inteligencia);
- Umelá inteligencia 2 (pôvodne Expertné systémy);
- Programovanie umelej inteligencie;
- Modelovanie telematických systémov;
- Inteligentné dopravné systémy;
- Fuzzy riadenie.

Katedra vydala pre túto oblasť v minulosti aj viacero knižných titulov, napr.:

- SPALEK, J. – JANOTA , A. – BALAŽOVIČOVÁ, M. – PŘIBYL, P. Rozhodovanie a riadenie s podporou umelej inteligencie. EDIS – vydavateľské centrum Žilinskej univerzity, 2005. ISBN 80-8070-354-X.
- GREGOR, M. Umelá inteligencia 1. Žilina: CEIT, a.s., 2014. ISBN 978-80-971684-1-4.
- JANOTA, A. – FRANEKOVÁ, M. – HOLEČKO, P. – HRBČEK, J. – PIRNÍK, R. – PŘIBYL, O. – PŘIBYL, P. – SPALEK, J. – ŠIMÁK, V. – VESTENICKÝ, P. Aplikovaná telematika. EDIS – vydavateľské centrum Žilinskej univerzity, 2015. ISBN 978-80-554-1037-1.





Vybrané knižné publikácie z oblasti umelej inteligencie.

Novým prírastkom medzi uvedené tituly sú tieto skriptá, obsahujúce návody na vybrané cvičenia z predmetov Umelá inteligencia 1 a Umelá inteligencia 2. Cieľom tohto nového textu je poskytnúť prehľad o niektorých vybraných témach, ku ktorým sa na oboch predmetoch realizujú cvičenia. Text môže poslúžiť ako praktický návod na realizáciu príslušných cvičení. Je užitočný pre študentov, ktorým poslúži ako forma kompaktných poznámok z praktických cvičení s podrobnejším komentárom, ktorý napomáha lepšie porozumenie diskutovaných tém. Skriptá môžu rovnako poslúžiť aj vyučujúcim oboch predmetov ako podklad na prípravu.

Na záver ešte jednu praktickú poznámku. V elektronickej verzii skrípt sú na viacerých miestach k textu priložené súbory. Sú označené nasledujúcim spôsobom: 🖇. Extrahovať ich možno typicky buď ľavým alebo pravým kliknutím (v závislosti od konkrétnej aplikácie použitej na čítanie PDF). Na miestach, kde sú v texte vložené rozsiahlejšie zdrojové kódy, tieto sú takisto priložené aj vo forme textových súborov a dajú sa otvoriť kliknutím na odkaz „[otvoriť zdrojový kód]“.

Aplikácia Adobe Reader má (vo verziách aktuálnych v čase písania tohto textu) predvolene zakázané akokoľvek manipulovať so súbormi v jazyku Python – či už ich ukladať na disk, alebo ich priamo otvoriť v inej aplikácii. Nastavenie je možné zmeniť jedine prostredníctvom systémových registrov[1]. Z tohto dôvodu odporúčame na prezeranie dokumentu použiť radšej inú, menej excentricky konfigurovanú aplikáciu.

POĎAKOVANIE

Na úvod tejto publikácie by sme radi osobitne poďakovali dvom zakladateľom tradície výučby a výskumu v oblasti umelej inteligencie a strojového učenia na Katedre riadiacich a informačných systémov: prof. Ing. **Jurajovi Spalekovi**, PhD. a prof. Ing. **Alešovi Janotovi**, PhD. Bez ich rozsiahlych aktivít v tejto oblasti a bez predmetov, ktoré spoločne vytvorili, by tieto skriptá nikdy nevznikli.

OBSAH

ZOZNAM SKRATIEK

CP programovanie ohraničení (angl. constraint programming), s. 113, 115, 116, 121, 126

CSP problém spĺňania ohraničení (angl. constraint satisfaction problem), s. 113–116, 122

ID3 iteratívny dichotomizér 3 (angl. Iterative Dichotomiser 3), s. 89, 98

PID proporcionálno integračno derivačný (regulátor), s. 44

ReLU jednotka (neurón) s rektifikovanou lineárnou funkciou (angl. rectified linear unit), s. 65

SAT problém spĺňania logických formúl (angl. boolean satisfiability problem), s. 115

KAPITOLA 1

ÚVOD DO JAZYKA PYTHON

1.1 | Úvod

Tento krátky dokument má za cieľ predstaviť minimálny úvod do jazyka Python pre potreby predmetov Umelá inteligencia 1 a Umelá inteligencia 2. Syntakticky sa zameriame na verziu 3.x. Existuje tu niekoľko rozdielov oproti verzii 2.x, ale nie sú príliš zásadné. Mnohé zdrojové kódy možno z verzie 2.x do verzie 3.x dokonca automaticky konvertovať pomocou skriptu 2to3, ktorý je súčasťou Python 3.x distribúcií.

Cieľom dokumentu nie je predstaviť všeobecný úvod do programovania v jazyku Python, ale len ukázať niekoľko základných rozdielov oproti práci v jazykoch C/C++/php a iných, s ktorými sa na našej katedre možno stretnúť. Okrem toho sa budeme sústreďovať predovšetkým na tie aspekty jazyka a na také balíčky, v ktorých prípade je pravdepodobné, že sa s nimi na predmete Umelá inteligencia stretneme.

Podrobnejší úvod do jazyka Python možno nájsť napríklad v oficiálnom návode [2]. Vrelo odporúčame si ho prelistovať. Je písaný jednoducho a je pomerne stručný. Základy je možné pohodlne preštudovať v priebehu dvoch dní. Užitočnou pomôckou na cvičeniach bude aj Python Cheatsheet. Obsahuje základné stavebné bloky, z ktorých budeme zostavovať skripty. Je ho možné nájsť v prílohách pod označením 7.5.2.

1.2 | Inštalácia Python-u a balíčkov

Úvod do Python-u začneme – nie príliš prekvapivo – jeho inštaláciou. Keďže v návode pracujeme s Python-om 3.x, ukážeme, ako sa inštaluje a používa konkrétne táto verzia. Postup inštalácie na Linux-ových a Windows-ových strojoch sa trochu líši, uvedieme preto obe možnosti.

1.2.1 Inštalácia Python-u na Linux-e

Väčšina Linux-ových distribúcií má Python k dispozícii ako balíček v repozitároch. Inštalácia je vďaka tomu vcelku jednoduchá – napr. na Ubuntu stačí z príkazového riadku spustiť:

Lst. 1:

[otvoriť zdrojový kód]

```
sudo apt-get install python3 python3-pip
```

Pripomeňme, že `sudo` slúži na spustenie zvyšku príkazu s administrátorskými privilégiami. Čo je obsahom samotného príkazu? Inštaluje z repozitárov dva balíčky: `python3` a `python3-pip`. Balíček `python3` obsahuje samotný Python interpreter. Nástroj pip [3] zase uľahčuje inštaláciu doplňujúcich python-ových balíčkov.

Okrem týchto dvoch základných balíčkov sa nám môžu zísť napríklad aj tie nasledujúce:

Lst. 2:

[otvoriť zdrojový kód]

```
python3-matplotlib python3-numpy python3-pyqt5 python3-scipy
sudo apt-get install spyder3
```

Okrem balíčkov pribalených v systéme je samozrejme možné nainštalovať si aj niektorú predpripravenú distribúciu Python-u pre Linux (populárna je napr. distribúcia Anaconda [4]). Táto možnosť je užitočná najmä vtedy, ak v repozitároch nie je k dispozícii požadovaná verzia Python-u.

1.2.2 Inštalácia Python-u na Windows-e

Pri inštalácii Python-u pre Windows (a podobne aj OS X), je dôrazne odporúčané zvoliť si niektorú predpripravenú python-ovú distribúciu – napríklad známu distribúciu Anaconda [4]. Výhodou je jednak, že väčšina distribúcií má jednoduchý grafický inštalátor, a tiež, že obsahujú nielen samotný Python, ale majú predinštalovaných aj mnoho prídavných balíčkov, ktoré pre nás môžu byť užitočné. Anaconda obsahuje navyše aj prostredie spyder, v ktorom budeme pracovať na niektorých cvičeniach.

1.2.3 Inštalácia balíčkov pomocou `pip`

Balíčky pre Python je možné inštalovať dvojakým spôsobom: buď priamo z python-ových repozitárov (pomocou nástroja pip [3]), alebo manuálne. Povedzme, že chceme nainštalovať balíček `pandas`, ktorý budeme na cvičeniach používať na prácu s niektorými typmi dátových súborov. Ak použijeme nástroj `pip`, stačí v príkazovom riadku napísať:

Lst. 3:

[otvoriť zdrojový kód]

```
sudo pip3 install pandas    # Ubuntu, Python 3
pip install pandas          # Windows / Anaconda na Linux-e
```

Na Ubuntu používame nie príkaz `pip`, ale príkaz `pip3`. Keďže v repozitároch je aj verzia Python-u 2.x, príkaz `pip` slúži na inštaláciu balíčkov pre túto verziu. Príkaz `pip3` je pre Python 3.x. Ak ale používame distribúciu Anaconda, aj na Linux-e použijeme jednoducho `pip`.

Obdobným spôsobom je možné inštalovať aj viacero balíčkov naraz, napr.:

Lst. 4:

[otvoriť zdrojový kód]

```
sudo pip3 install balicek1 balicek2     # Ubuntu, Python 3
pip install balicek1 balicek2           # Windows / Anaconda na Linux-e
```

Výhodou `pip` je najmä to, že umožňuje vyhľadávať a inštalovať balíčky priamo z pythonových repozitárov. Nie je potrebné ich komplikovane vyhľadávať na internete, manuálne sťahovať a inštalovať.

1.2.4 Manuálna inštalácia balíčkov

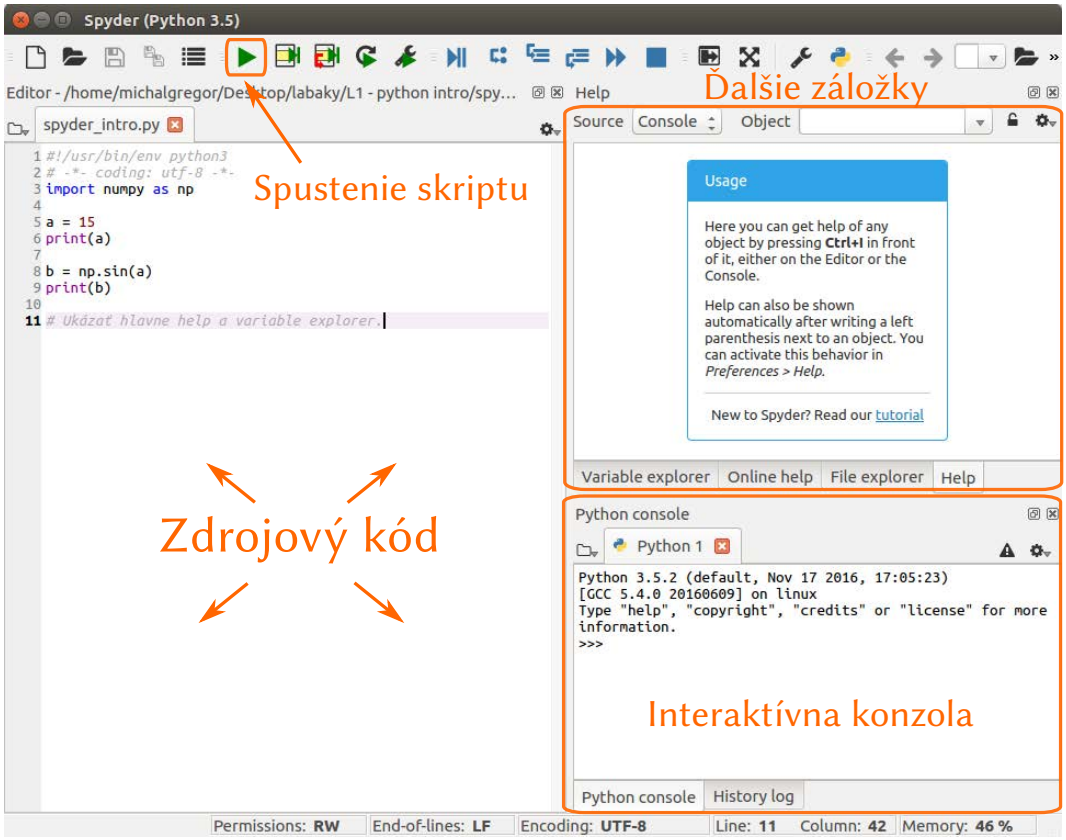
Ak chceme inštalovať balíček, ktorý nie je súčasťou repozitárov, alebo chceme vyskúšať jeho novšiu, vývojovú verziu, je potrebné si balíček stiahnuť manuálne. K balíčkom býva obvykle priložený súbor s názvom `setup.py`, ktorý obsahuje inštalačný skript. Spustíme ho príkazom `python setup.py install`, resp. `python3 setup.py install` – ak sme na Ubuntu.

1.3 | Prostredie Spyder

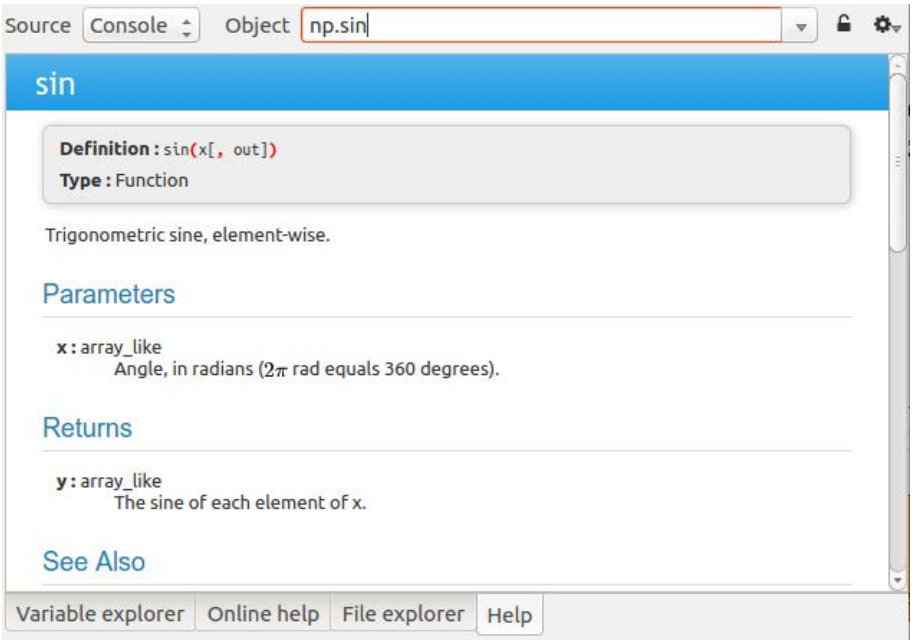
Na cvičeniach kde sa pracuje s Python-om, budeme používať dva typy prostredí. Prvým z nich je vývojové prostredie Spyder. Je k dispozícii v Ubuntu repozitároch a takisto aj v rámci distribúcie Anaconda. Základné rozhranie ukazuje Obr. 1.1. Hlavná časť okna zobrazuje zdrojový kód. Ten je možné spustiť priamo v Spyder-i: v interaktívnej konzole. Kód sa spúšťa buď použitím tlačidla označeného na obrázku, alebo stlačením klávesy `F5`. Zvyšok okna obsahuje ešte ďalšie záložky – ku niektorým z nich sa ešte vrátime nižšie.

1.3.1 Vstavaná nápoveda

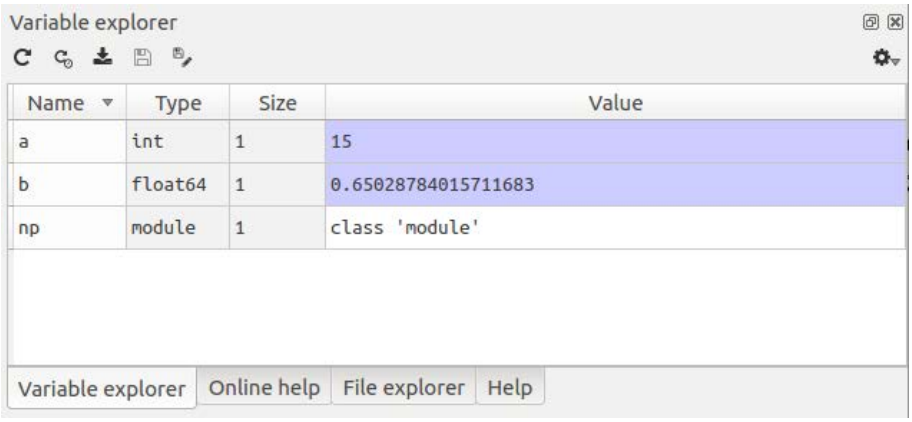
Prostredie Spyder umožňuje zobrazovať vstavanú nápovedu. Ide o zabudovanú dokumentáciu ku jednotlivým funkciám, triedam a pod. Zabudovaná dokumentácia sa v Python-e realizuje pomocou tzv. dokumentačných reťazcov, o ktorých ešte budeme hovoriť nižšie (časť 1.6.2). V prostredí Spyder sa dá dokumentácia zobraziť pomocou vstavaného okna `Help`, ktoré vidno na Obr. 1.2. Doňho stačí zadať názov funkcie, triedy, ... a Spyder zobrazí príslušný záznam z dokumentácie.



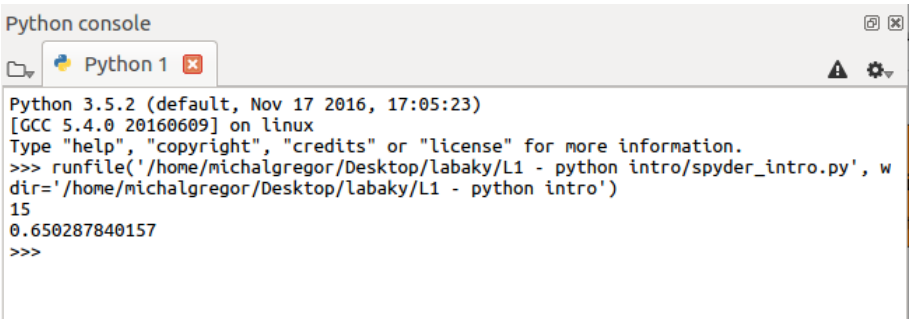
Obr. 1.1: Grafické užívateľské rozhranie prostredia Spyder.



Obr. 1.2: Okno `Help` v prostredí Spyder.



Obr. 1.3: Prehliadanie premenných v prostredí Spyder.



Obr. 1.4: Interaktívna konzola v prostredí Spyder.

1.3.2 Prehliadanie premenných

Spyder obsahuje aj prehliadač premenných, ktorý umožňuje zobraziť premenné, ktoré práve existujú v aktívnej python-ovej konzole. Zobrazí ich typy, veľkosť a hodnoty, ako to vidno na Obr. 1.3. Takýto náhľad môže byť užitočný pri ladení programu. V prehliadači sa dá okrem toho nastaviť, aby sa náhľad premenných s určitou periodicitou aktualizoval.

1.3.3 Interaktívne konzoly

Jeden zo segmentov grafického rozhrania v prostredí Spyder obsahuje interaktívne konzoly. Ilustrácia je na Obr. 1.4. V konzole, ktorá je v tom čase aktívna, sa spúšťajú skripty a vypisujú sa ich výsledky. Konzol je možné otvoriť aj viacero súčasne alebo ich, povedzme, nasilu ukončiť (to je užitočné napríklad v prípade, keď vykonávaný skript zamrzne, či vstúpi do nekonečného cyklu).

1.4 | Prostredie iPython Notebook

Ako druhé prostredie budeme na niektorých cvičeniach používať prostredie iPython Notebook. Notebook-y predstavujú interaktívne aplikácie kombinujúce formátovaný text, obrázky (prípadne

širší multimediálny obsah) a zdrojový kód. Forma ich prezentácie a interakcie s nimi je webová. Na ich lokálne spustenie je potrebné mať nainštalovaný Python spoločne s balíčkami Jupyter a iPython. Notebooky sú však integrované aj do niektorých cloudových služieb. Napr. spoločnosť Microsoft ich ponúka v rámci platformy Azure. Ich pasívne prezeranie (bez možnosti spustenia kódu) umožňuje napríklad aj portál github.

Notebook-y majú niekoľko podstatných výhod oproti klasickým python-ovým zdrojovým súborom, ak cieľom je ponúknuť interaktívny návod ku nejakému nástroju, balíčku, metóde a pod. Umožňujú totiž kombinovať štruktúrovaný text a obrazový materiál s postupne prezentovanými segmentmi kódu. Umožňujú teda postupne, vo viacerých krokoch, vysvetliť jednotlivé koncepty spojené s danou témou. Navyše, keďže kód nie je prezentovaný v jednom monolitickom bloku, ale po menších častiach, je možné priamo pozorovať efekty jednotlivých jeho segmentov. V prípade, že dôjde ku chybe, je navyše užívateľovi jasné, v ktorom kroku prestali veci fungovať tak, ako mali. To mu umožňuje hneď sa vrátiť ku príslušným inštrukciám a pokúsiť sa ich vykonať znovu: pochopiť, v čom bol problém.

Podrobnejšie návody na použitie notebook rozhrania je možné nájsť napr. v [5]. My sa na tomto mieste zameriame len na tie aspekty, ktoré bude najužitočnejšie poznať na cvičeniach z predmetu Umelá inteligencia 1.

1.4.1 Lokálne spustenie notebook-u

Ak máme k dispozícii nejakú distribúciu Python-u a balíčky Jupyter a iPython (v distribúcii Anaconda je všetko potrebné pribalené), lokálne spustenie iPython notebook-ov je jednoduché. Stačí otvoriť príkazový riadok v priečinku, kde sa iPython notebook nachádza, a zadať príkaz:

Lst. 5:

[otvoriť zdrojový kód]

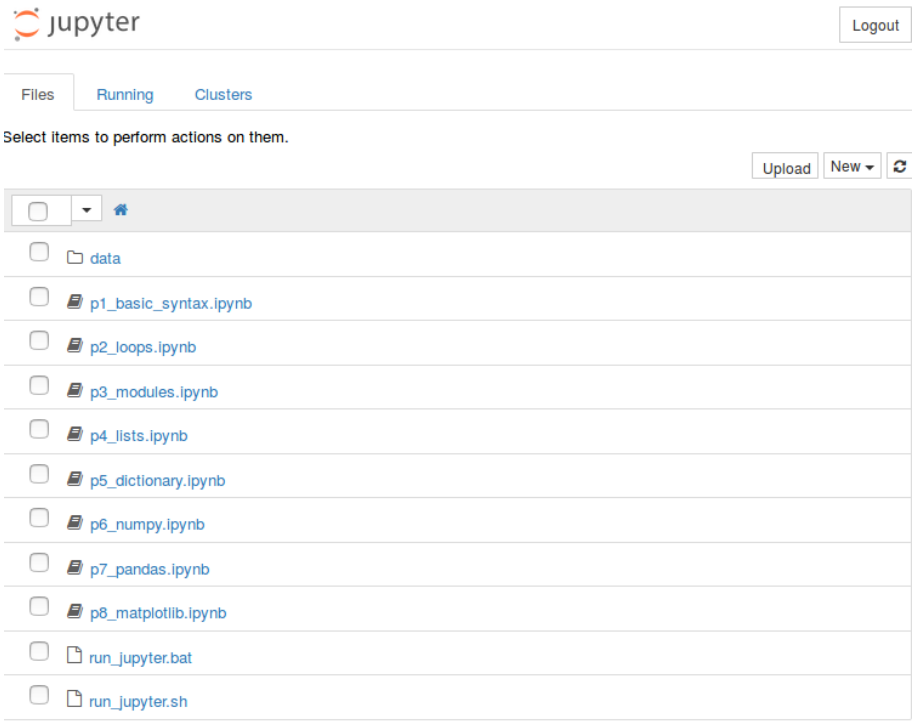
```
jupyter notebook .
```

Tým spustíme notebook-ový jupyter server z aktuálneho priečinku. Zároveň sa automaticky otvorí aj webový prehliadač s príslušným rozhraním. Ukážka je na Obr. 1.5.

Ako vidno, webové rozhranie ukazuje zoznam relevantných súborov v aktuálnom priečinku. Stačí si z nich vybrať požadovaný iPython notebook a otvoriť ho. Rozhranie samotných notebook-ov ukazuje Obr. 1.6.

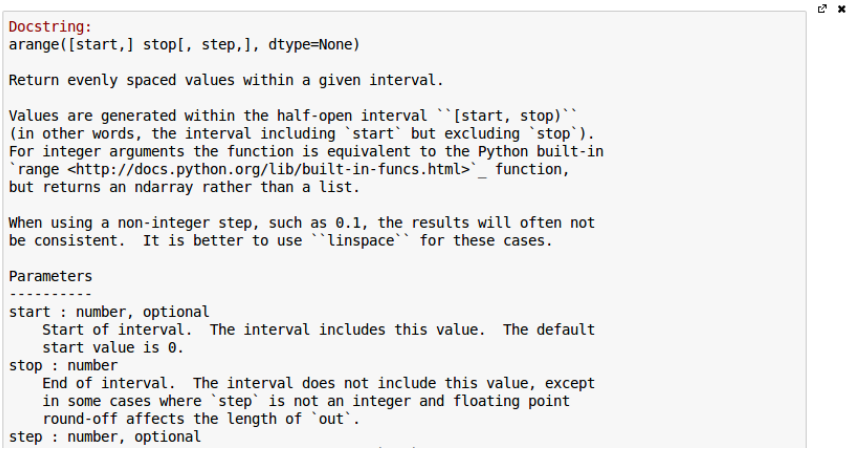
1.4.2 Mardown a bunky s kódom

Každý iPython notebook sa skladá z buniek. Bunky sú viacerých typov. My sa zaujímame jednak o `markdown` bunky a jednak o `code` bunky. `Markdown` bunky obsahujú formátovaný text (písaný v markdown jazyku podobnému tomu, čo používa Wikipedia). Ďalej môžu obsahovať obrázky, hypertextové odkazy, multimediálny obsah alebo v podstate ľubovoľný interaktívny webový obsah. Obsah `markdown` bunky sa dá na editovanie otvoriť dvojklikom myšou.



Obr. 1.5: Webové rozhranie Jupyter.

Obr. 1.6: iPython notebook.



Obr. 1.7: iPython: dokumentácia v plávajúcom okne.

Odlišné sú bunky typu `code`, ktoré obsahujú jednoducho python-ový kód. Je ich možné priamo z notebook-u interaktívne spustiť. Spustiť sa dajú aj bunky typu `markdown`, ale v ich prípade to len znamená, že sa ich obsah (zapísaný v `markdown` jazyku) zobrazí v podobe formátovaného textu.

Bunku je možné spustiť nasledujúcim spôsobom:

1. Najprv ju kliknutím označíme.
2. Potom stlačíme `SHIFT+ENTER` alebo klikneme na tlačidlo play v hornom paneli.

Je aj možnosť spustiť zaradom všetky bunky v notebook-u. Je dostupná cez menu Cell->Run All.

Nové bunky je možné pridať pomocou tlačidla plus alebo stlačením `SHIFT+ENTER` v poslednej bunke notebook-u. Ak chceme existujúce bunky zmazať, je to možné urobiť buď cez menu Edit->Delete Cells, alebo tak, že ich kliknutím vyberieme a použijeme klávesovú skratku `D+D` (stlačíme dvakrát po sebe klávesu D).

1.4.3 Vstavaná nápoveda

Prístup ku iPython dokumentácii je v menu Help. Dá sa tiež dotazovať na vstavanú dokumentáciu pre konkrétne funkcie, triedy, objekty a pod. Realizuje sa to tak, že v bunke typu `code` napíšeme otáznik nasledovaný identifikátorom príslušného objektu, napr.:

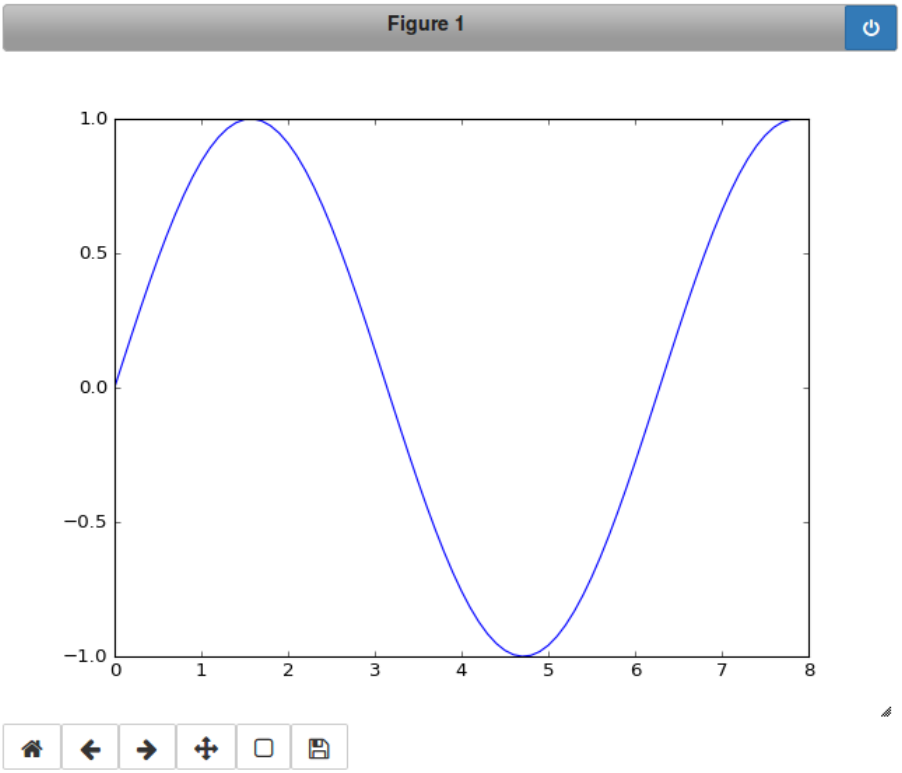
Lst. 6:

[otvoriť zdrojový kód]

```
1 import numpy as np
2 ?np.arange
```

Druhou možnosťou je zobraziť vstavanú dokumentáciu v plávajúcom okne. Tento typ dokumentácie sa zobrazuje stlačením kláves `SHIFT+TAB`. Príklad je na Obr. 1.7.

Zabudovaná dokumentácia sa v Python-e realizuje pomocou tzv. dokumentačných reťazcov, o ktorých ešte budeme hovoriť nižšie (časť 1.6.2).



Obr. 1.8: iPython: interaktívny graf.

1.4.4 Magické príkazy

Okrem štandardného python-ového kódu sa v iPython termináloch dajú vykonávať aj tzv. magické príkazy (angl. magic commands). Slúžia väčšinou na parameterizáciu samotného iPython prostredia. My sa nimi nebudeme zaoberať podrobne. Použijeme len jeden magický príkaz, ktorý nám umožní v notebook-och zobrazovať interaktívne grafy pomocou balíčka `matplotlib`. Tento magický príkaz má nasledujúci tvar:

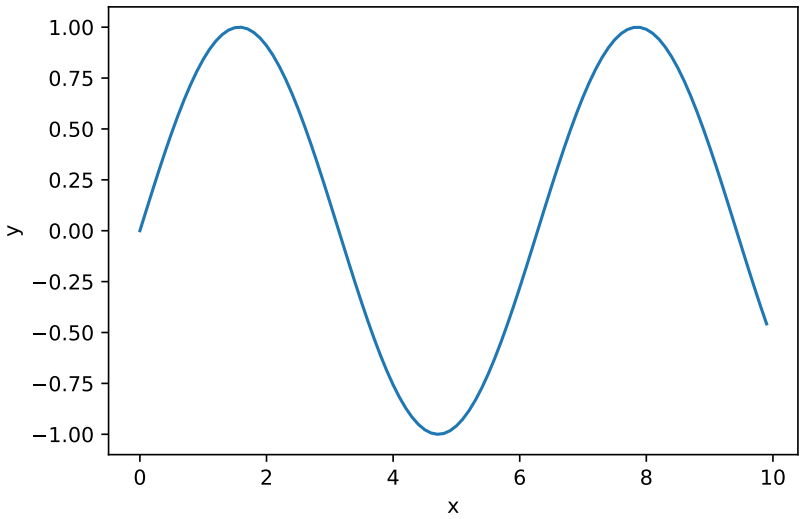
Lst. 7:

[otvoriť zdrojový kód]

```
%matplotlib notebook
```

Na pozadí ním vlastne nastavujeme, aký backend má použiť balíček `matplotlib` pri vykresľovaní grafov. Backend notebook je interaktívny backend pre iPython notebook-y. Ak by sme chceli zobrazovať v rámci notebook-u grafy len staticky – nie interaktívne – mohli by sme namiesto toho použiť backend `inline`. Výhodou interaktívnych grafov je, že umožňujú časti zobrazeného grafu priblížiť, otočiť, výsledné zobrazenie exportovať ako obrázok a pod. Ukážka interaktívneho grafu spoločne s jeho ovládacími prvkami je na Obr. 1.8.

Magický príkaz `%matplotlib notebook` stačí spustiť raz – najlepšie hneď ako jednu z prvých inštrukcií po otvorení notebook-u. *Nie je* potrebné spúšťať ho znovu pred každým blokom kódu, ktorý vykresľuje grafy.



Obr. 1.9: Výstup zdrojového kódu.

1.5 | Ukážka python-ového kódu

Nie je dobré dlho diskutovať o vlastnostiach programovacieho jazyka a neukázať príklad zdrojového kódu. Preto už na tomto mieste prikladáme malý kúsok python-ového kódu, ktorý vygeneruje a vykreslí sínusovú funkciu. Samotnému kódu netreba ešte detailne rozumieť – má slúžiť len na ilustráciu toho, ako vyzerá syntax jazyka. Výstupom kódu je graf na Obr. 1.9.

Lst. 8: [otvoriť zdrojový kód]

```
import numpy as np
import matplotlib.pyplot as plt

# Vygenerujeme postupnosť čísel z rozsahu <0, 10) s krokom 0.1.
x = np.arange(0, 10, 0.1)

# Vypočítame pre ne sínus.
y = np.sin(x)

# Vykreslíme.
plt.plot(x, y)
plt.xlabel("x")
plt.ylabel("y")
```

1.6 | Základy python-ovej syntaxe

Ďalej predstavíme hlavné rysy python-ovej syntaxe – predovšetkým tie, ktorými sa Python najviac líši od iných známych jazykov ako sú C, C++, Java a pod.

V Python-e, tak ako v mnohých skriptovacích jazykoch, sa typy premenných explicitne nedeklarujú. Píšeme priamo:

Lst. 9:

[otvoriť zdrojový kód]

```
a = 7
x = 1.5
s = "textový reťazec"
```

Tým sme vytvorili celočíselnú premennú `a`, reálnu premennú `x` a textový reťazec `s`.

Ako vidno, jednotlivé príkazy sa oddeľujú znakom nového riadku. Ak chceme mať v tom istom riadku viacero príkazov, môžeme ich však oddeliť bodkočiarkou, napr.:

Lst. 10:

[otvoriť zdrojový kód]

```
a = 7; x = 1.5
```

Veľkou zvláštnosťou Python-u je, že v ňom majú význam aj iné biele znaky. Bloky kódu sa napríklad vyjadrujú odsadením, t.j.:

Lst. 11:

[otvoriť zdrojový kód]

```
if x == 1 and a < 10:
    y = 5
elif a > 17 or x == 0:
    y = 11
    z = 12
else:
    y = 0
```

Povedzme v C/C++ by sme to isté realizovali pomocou zložených zátvoriek, napr.:

Lst. 12:

[otvoriť zdrojový kód]

```
if(x == 1 && a < 10) {
    y = 5;
} else if(a > 17 && x == 0) {
    y = 11;
    z = 12;
} else {
    y = 0;
}
```

Syntax `if-then` podmienok ešte podrobnejšie ukážeme na inom mieste. Tu chceme ilustrovať len toľko, že bloky kódu sa tvoria odsadením. Spočiatku môže byť problém si na režim odsadzovania zvyknúť. Je však pravda, že bloky kódu sa typicky kvôli prehľadnosti odsadzujú aj v jazykoch, ktoré to priamo nevyžadujú. Rozdiel je teda len v tom, že Python si túto elementárnu kultúru formátovania kódu vynucuje.

Odsadenie blokov musí byť konzistentné (napr. vždy odsadzovať štyrmi znakmi). Nekonzistentné odsadenie sa považuje za syntaktickú chybu, napr.: