Python OOP

I. Začiatočník

IT ACADEMY

Lektor: PhDr. Ing. Mgr. Miroslav Reiter, DiS., MBA, DBA
Kontakt: riaditel@itacademy.sk

Ako Začneme?

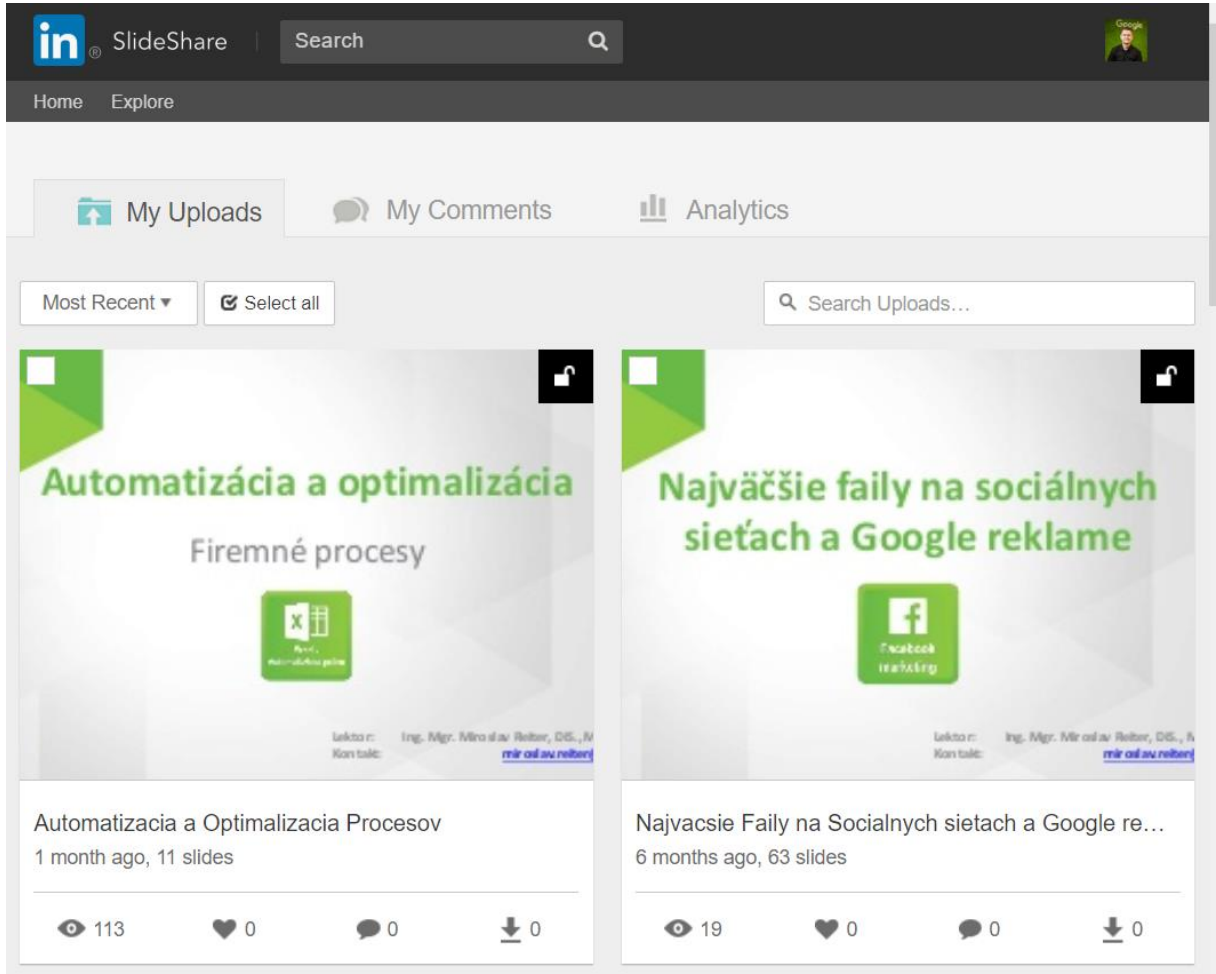
1. **Stiahnite si Cvičný Súbor**

➢ https://github.com/miroslav-reiter/Kurzy_SAV_Analytika_Python_R

2. **Pridajte** si ma na **LinkedIn**

➢ www.linkedin.com/in/miroslav-reiter/

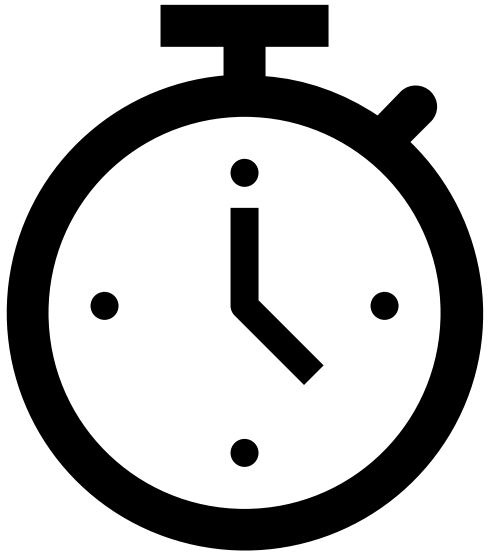
3. **Prezentácia** a **materiály** po prednáške



Úvodné Informácie

- Časový rozvrh (9:00-13:30)
- Prestávky
- Mobilné telefóny a zariadenia

- Priprav si otázky a rovno sa pýtaj
- Interaktívna forma



O mne - Miroslav Reiter

40000+ klientov a 1000+ firiem

IT Architekt
Programátor
Manažér

Microsoft
Google
ISTQB tréner

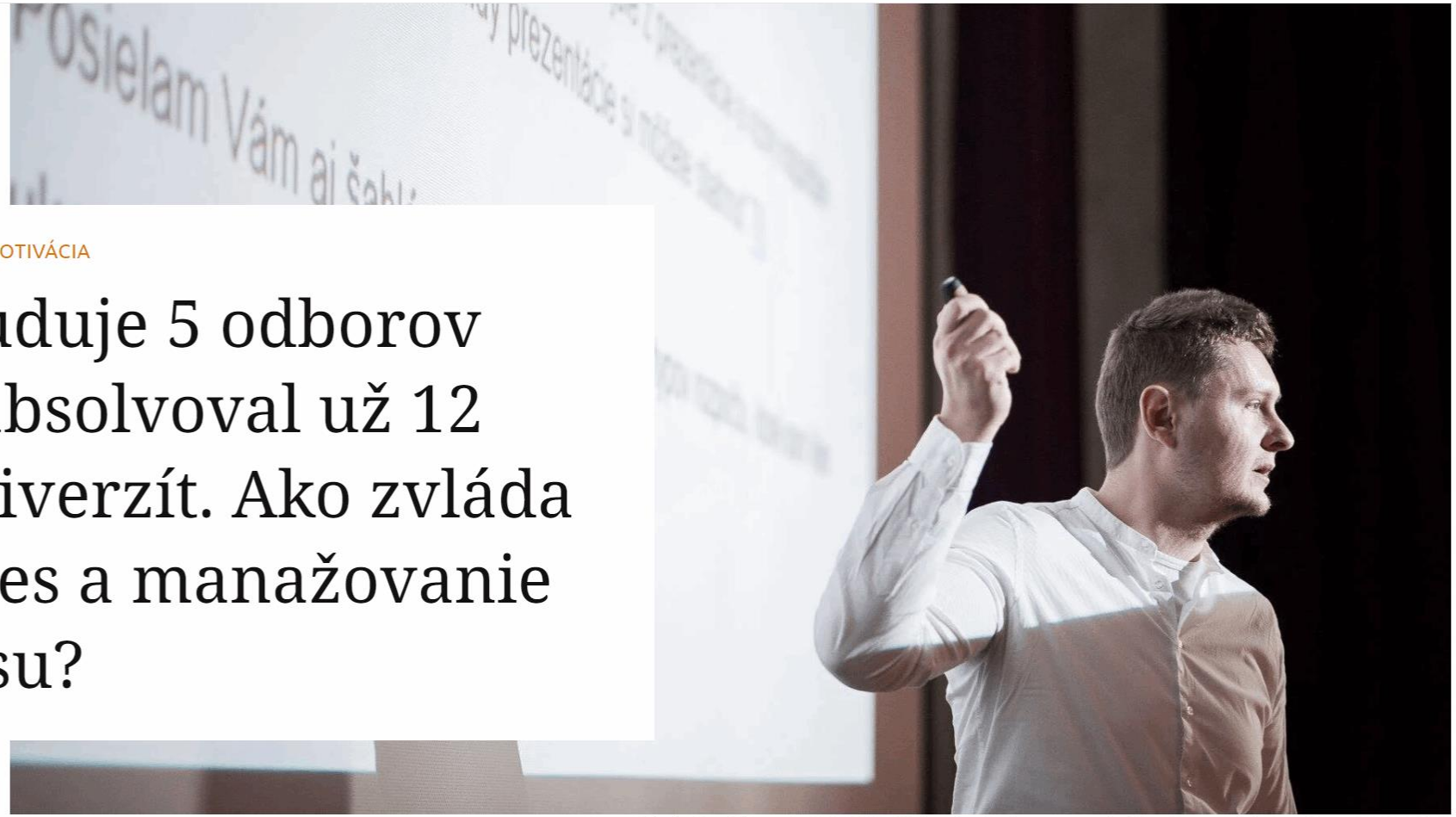
134 certifikácii

151 príručiek a publikácii

13 škôl

62 projektov

Vlastná firma



MOTIVÁCIA

Študuje 5 odborov a absolvoval už 12 univerzít. Ako zvláda stres a manažovanie času?

Foto: Jakub Kovalík pre FMK UCM | Miroslav Reiter na prednáške Grow with Google na FMK UCM.

Nikola Kotláriková

19. júl 2022 · 8 min. čítania

Miroslav Reiter

1. PhDr. VŠM (Podnikovný manažment)
2. **Ing. STU FEI (Aplikovaná informatika)**
3. **Mgr. UK FM (Strategický manažment a marketing)**
4. **Mgr. VŠM (Manažérstvo kvality)**
5. Mgr. VŠEMVŠ (Verejná správa)
6. **Mgr. DTI (Učiteľstvo ekonomických predmetov)**
7. DiS. AMOS (Cestovný ruch)
8. **MBA LIGS (Executive management)**
9. **DBA Humanum (IT manažment)**
10. MPA IES (Verejná správa a samospráva krajov)
11. **MSc. Humanum (Bezpečnosť inf. systémov)**
12. **Ing. Paed. IGIP STU**
13. **Mgr. PEVŠ (Bezpečnosť informačných systémov)**

DIGITÁLNA KOALÍCIA.SK

SLOVENSKÁ AKADÉMIA VIED ACADEMIA SCIENTIARUM SLOVACA

IT ACADEMY

DIGITÁLNA UNIVERZITA

STU FIIT

FAKULTA MANAGEMENTU
Univerzita Komenského
v Bratislave

VITA

Filter

People | Events

Search by filter

- Program
- Country/Region
- State/Province
- Language
- Award Category
- Technology Area

Search Profiles

Search Profiles

Search

Most Valuable Professionals ✕ | Country/Region: Slovakia ✕

Most Valuable Professionals

Miroslav Reiter

🌟 IT Architect and Programmer 💼 Hard worker 🎓 Lecturer and Certified Trainer

Slovakia

Most Valuable Professionals

Peter Belko

Content Developer, Technical Writer, Consultant, Trainer for M365 + Office apps

Slovakia

← 1 →

Show 18 results

Miroslav Reiter

Slovakia

IT Academy s.r.o.

Most Valuable Professionals

High Impact Activities

This community leader has not added a high impact activity yet.

Profile Events

Headline

🌞 IT Architect and Programmer 💼 Hard worker 🎓 Lecturer and Certified Trainer

Biography

🤦 I'm a hard worker, intellectual and joker. I love learning and teaching as well. My main objective is to teach people and improve their IT knowledge. To create truly practical knowledge necessary for life and present it in an interesting way. I don't like snake charmers and people who cannot...

Read more

Award Category

M365

Technology Area

Visio, Excel

Languages

English, Slovak

Social

Miroslav Reiter

získava status

Google Certified Trainer

Automation

Google

Znalec

PhDr. Ing. Mgr. Miroslav Reiter

Evidenčné číslo: 915864

Miesto výkonu činnosti

Tomášikova 50
83104 Bratislava
Slovenská republika

Zobraziť na mape

Kontaktné údaje

Mobil: +421 908 163 084
E-mail: znalec@it-academy.sk

Odbory a odvetvia

Odbor / Odvetvie	História	Stav
100000 - Elektrotechnika		
100400 - Riadiaca technika, výpočtová technika (hardware)	14.02.2023 - **Zápis**	AKTÍVNY
100800 - Nosiče zvukových a zvukovoobrazových záznamov	14.02.2023 - **Zápis**	AKTÍVNY
100900 - Počítačové programy (software)	14.02.2023 - **Zápis**	AKTÍVNY
101000 - Bezpečnosť a ochrana informačných systémov	14.02.2023 - **Zápis**	AKTÍVNY

VITA
Virtual IT Academy

excel

Registrácia | Prihlásenie

Online Kurzy | Ročné Predplatné Kurzy | IT IQ Sieň Slávy | REPAS a KOMPAS | Kurzy pre Firmy | Časté Otázky | Kontakty | SK

Vyberte si online kurz

Naučte sa programovať, tvoriť webstránky a grafiku, manažovať alebo sa zamerajte na osobný rozvoj. Všetko jednoducho vďaka našim online kurzom z pohodlia domova.

Ročné predplatné na všetky online kurzy

~~2299.99€~~

399.99€

Prístup pre Vás do všetkých **aktuálnych aj pripravovaných** online kurzov

12 mesačná platnosť

Kúpiť teraz

Zistiť viac

- 432 kurzov v ponuke
- Zábavné online lekcie
- Akreditované kurzy
- 12 rokov skúseností
- Certifikovaní profesionálni lektori

Odporúčame Kurzy špeciálne pre vás

Online kurz SAP I. Začiatočník
224.00€ ~~292.50€~~

Online kurz SQL I. Začiatočník
89.00€ ~~112.00€~~

Online kurz Copywriting I. Začiatočník
50.00€ ~~66.20€~~

Online kurz Projektový Manažér I. Začiatočník
196.00€ ~~268.01€~~

Online kurz Data Science I. Začiatočník
119.00€ ~~139.00€~~

Online kurz Java I. Začiatočník
67.00€ ~~88.40€~~

Luigi, Mário a Yoshi

Čo Robíte?

1. Študent/učiteľ
2. Zamestnanec
3. Podnikateľ
4. Nezamestnaný/materská
5. Dievča pre všetko

National competence centre for high performance computing SLOVAKIA

EURO

EuroHPC
Joint Undertaking

SLOVENSKÁ AKADÉMIA VIED
SLOVAK ACADEMIA SCIENTIARUM

Národné kompetenčné centrum pre vysokovýkonné počítanie

Slovenská akadémia vied

Projekt EuroCC

Vzdelávanie

Kurzy:
itkurzy.sav.sk

Propagácia

Prednášky:
https://eurocc.nscc.sk/news/prednasky/

HPC služby

Prístup k výpočtovým prostriedkom

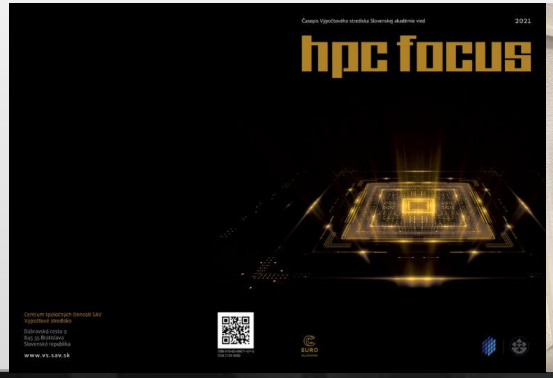
Mapovanie HPC prostredia

Prieskum:
https://eurocc.nscc.sk/mapping-survey/

Spolupráca

Pilotné projekty

Dlhodobá spolupráca



Qubit Conference

robíme it

Slovenská obchodná a priemyselná komora

S kým spolupracujeme:

- Akademické inštitúcie, univerzity, ústavy SAV,...
- Verejná správa
- Súkromné firmy, tretí sektor

Naučte sa pracovať v prostredí HPC systémov:

Najbližší kurz:

HPC infraštruktúra / 18. máj 2022

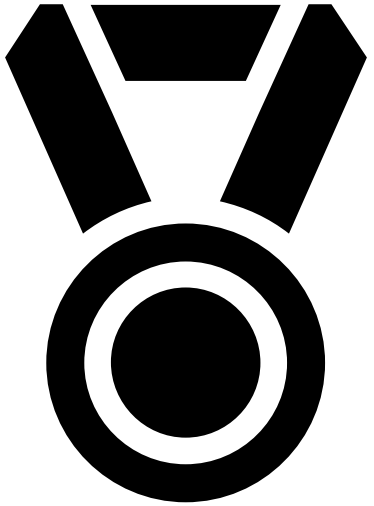
Hľadáme nových kolegov do tímu!
https://eurocc.nscc.sk/career/

Sledujte nás na sociálnych sieťach:

Vaše Ciele a Očakávania

1. Doplniť si znalosti z jazyka Python
2. Základy objektového myslenia
3. Základy OO programovania
4. Doplniť si znalosti z programovania
5. Doplniť si znalosti z vývojových prostredí (IDE)
6. Využitie AI pri programovaní

Zábava je v zaručená v každom bode :-)



Čo je OOP a na čo je dobré?

Zoznam Premenných (Objektov)

python

Zoznam Premenných (Objektov)

- dir() will give you the list of in scope variables:
- globals() will give you a dictionary of global variables
- locals() will give you a dictionary of local variables

- who - zoznam premenných, len mená/názvy
- whos - zoznam premenných, mená/názvy aj typ a hodnoty
- who_ls - zoznam premenných, len mená/názvy, doslova zoznam
- _ ukáž podrtrznik
- vars()
- vars().keys()
- vars().values()

nbextension variable inspector

- **Install**

➢`pip install jupyter_contrib_nbextensions`

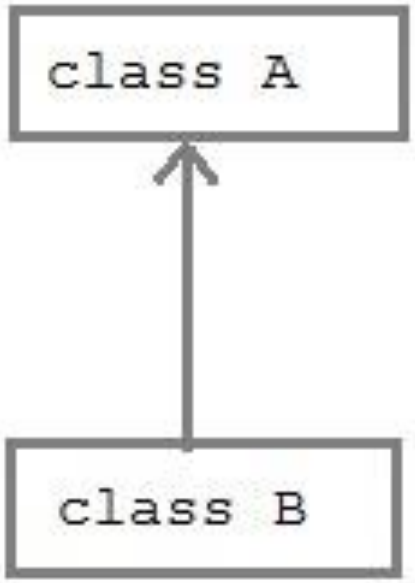
➢`jupyter contrib nbextension install --user`

- **Enable**

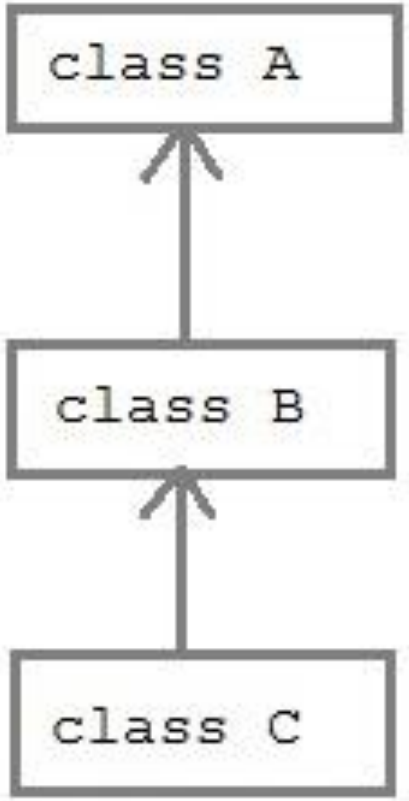
➢`jupyter nbextension enable varInspector/main`

Len priamo z CMD

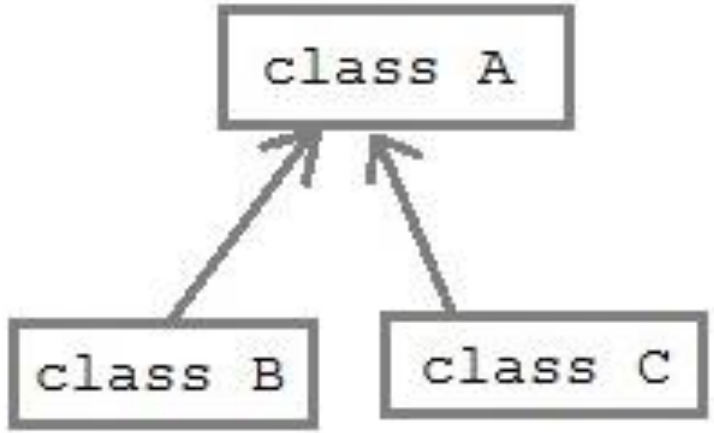
Typy Dedičností



Simple Inheritance



Multilevel inheritance



Heirarchical inheritance

Objekty (Objects)

python

Triedy (Classes)

jupyter

Quit Logout

Files Running Clusters Nbextensions

Configurable nbextensions

☐ disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

filter: by description, section, or tags

- ☐ (some) LaTeX environments for Jupyter
- ☐ AutoSaveTime
- ☐ Code prettify
- ☐ Collapsible Headings
- ☐ Equation Auto Numbering
- ☐ Exercise2
- ☐ Help panel
- ☐ Highlight selected word
- ☐ isort formatter
- ☐ Limit Output
- ☐ Navigation-Hotkeys
- ☐ Notify
- ☐ Ruler
- ☐ ScrollDown
- ☐ Snippets
- ☐ Table of Contents (2)
- ☑ Variable Inspector
- ☐ 2to3 Converter
- ☐ Autoscroll
- ☐ Codefolding
- ☐ Comment/Uncomment Hotkey
- ☑ ExecuteTime
- ☐ Export Embedded HTML
- ☐ Hide Header
- ☐ highlighter
- ☑ jupyter-js-widgets/extension
- ☐ Live Markdown Preview
- ☑ Nbextensions dashboard tab
- ☐ Printview
- ☐ Ruler in Editor
- ☑ Select CodeMirror Keymap
- ☑ Snippets Menu
- ☐ table_beautifier
- ☑ zenmode
- ☐ AddBefore
- ☐ Cell Filter
- ☐ Codefolding in Editor
- ☑ contrib_nbextensions_help_item
- ☐ Execution Dependencies
- ☐ Freeze
- ☐ Hide input
- ☐ Hinterland
- ☐ Keyboard shortcut editor
- ☐ Load TeX macros
- ☑ Nbextensions edit menu item
- ☐ Python Markdown
- ☐ Runtools
- ☐ SKILL Syntax
- ☑ spellchecker
- ☑ Toggle all line numbers
- ☑ Autopep8
- ☐ Code Font Size
- ☐ CodeMirror mode extensions
- ☑ datestamper
- ☐ Exercise
- ☑ Gist-it
- ☐ Hide input all
- ☐ Initialization cells
- ☐ Launch QTConsole
- ☐ Move selected cells
- ☐ nbTranslate
- ☐ Rubberband
- ☐ Scratchpad
- ☐ Skip-Traceback
- ☐ Split Cells Notebook
- ☐ Tree Filter

ExecuteTime

Display when each cell has been executed and how long it took

section: notebook
require path: `execute_time/ExecuteTime`
compatibility: 4.x, 5.x

Enable **Disable**

Last executed 2016-02-17 13:39:49 in 5ms

```
In [2]: randword(0.5)
```

Last executed 2016-02-17 13:39:49 in 519ms

```
Out[2]: 'zsygaoxnhe'
```

jupyter Premenne Last Checkpoint: pred 27 minútami (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [1]:

```
1 a = 5
2 b = True
3 c = "Karol"
4 d = []
5 e = {}
6 f = ()
7 g = 6.5
8
```

In [ ]:

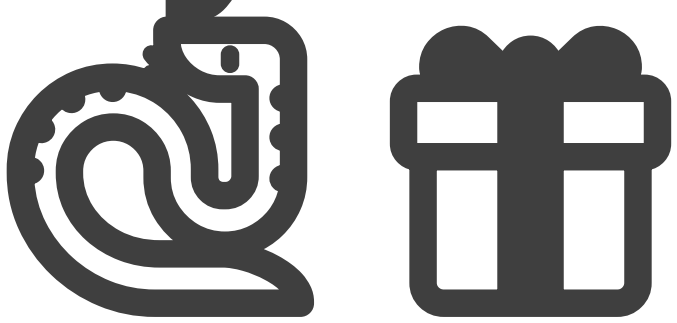
```
1
```

Variable Inspector

↻[-][x]

X	Name	Type	Size	Value
x	a	int	28	5
x	g	float	24	6.5
x	f	tuple	40	()
x	d	list	56	[]
x	e	dict	64	{}
x	b	bool	28	True
x	c	str	54	Karol

Objektové Programovanie Python

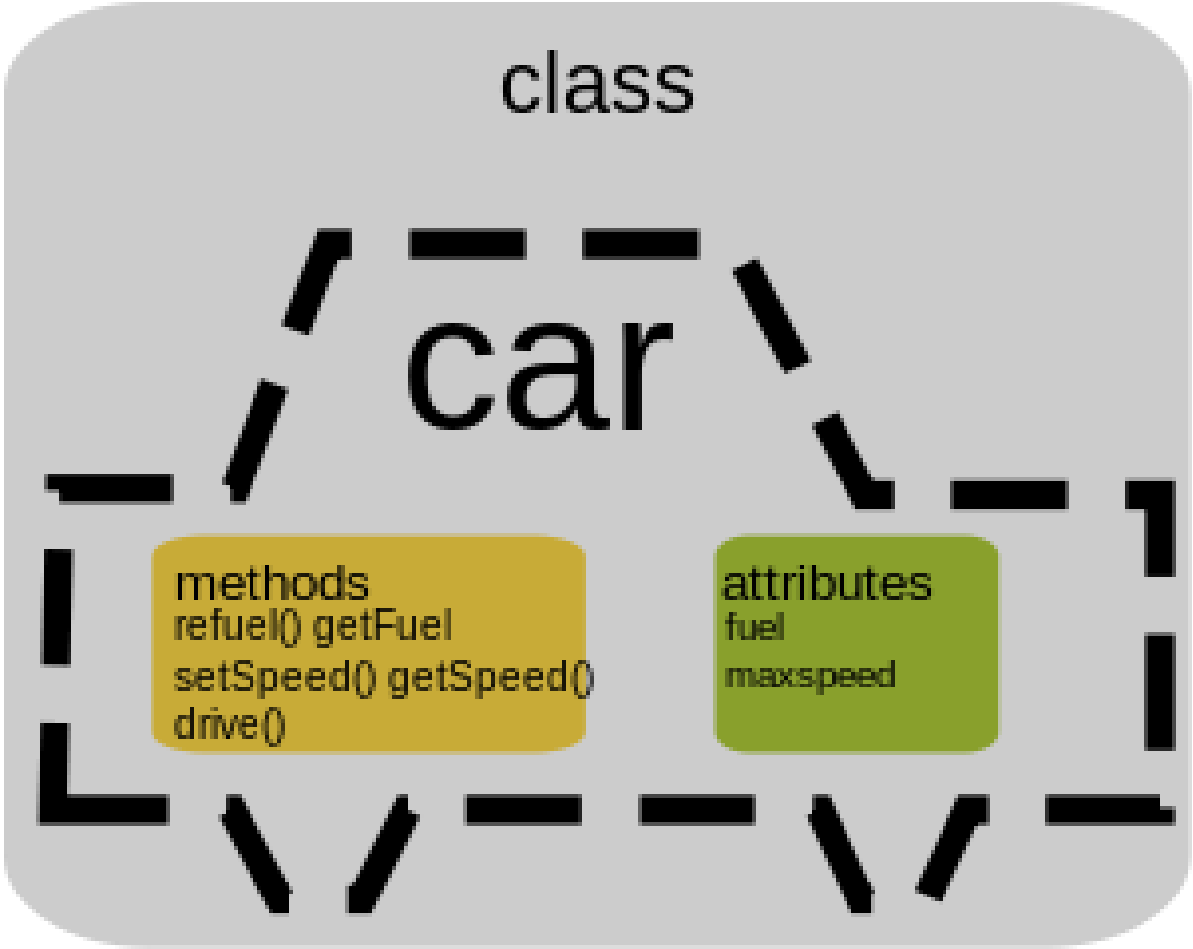


OOP

1. Triedy a Objekty
2. Atribúty a Metódy
3. Konštruktor __init__(self)
4. Dedičnosť
5. Polymorfizmus
6. Zapuzdrenie

Prečo Používame OOP?

- Nová paradigma
- Časti programu sú zoskupované do väčších celkov – objektov
- Objekty sa vytvárajú na základe preddefinovaných kritérií – tried
- Prehľadnejší kód
- Podobnosť k reálnemu svetu



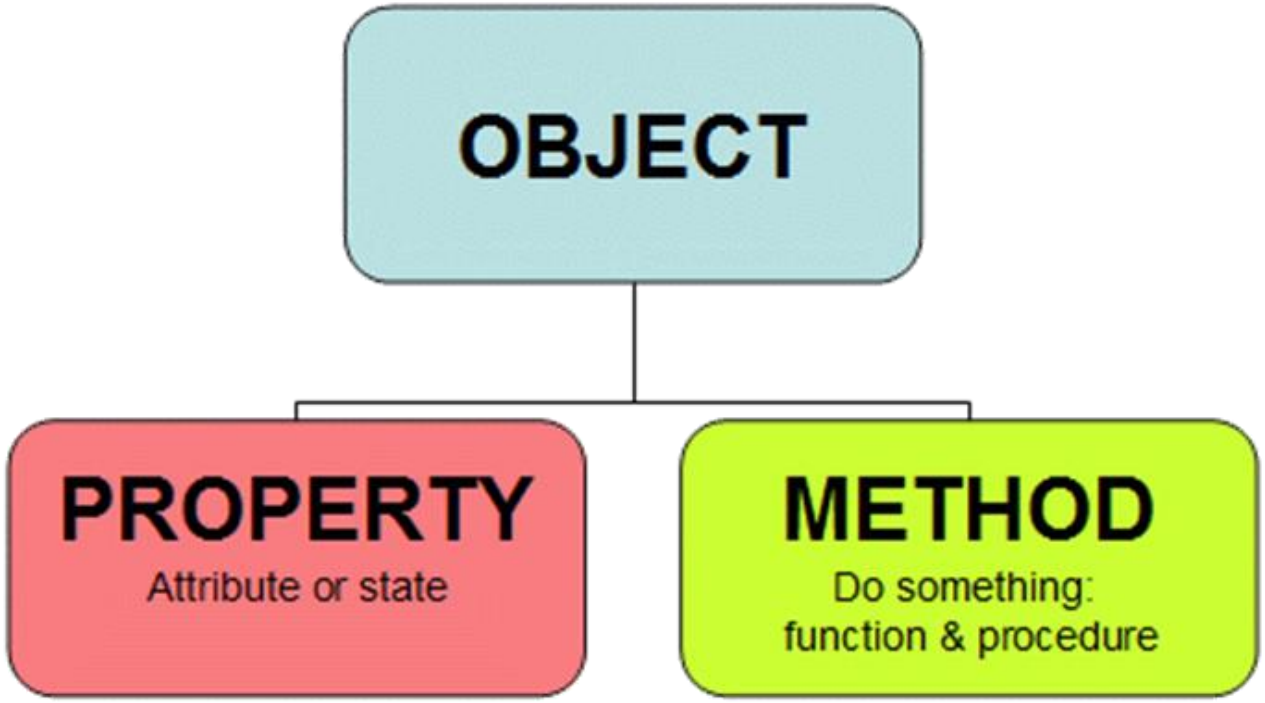

Abstraction
Encapsulation
Modularity
Messaging
object
oriented
programming
Identity
Hierarchy
Typing
Persistence
Concurrency
Polymorphism

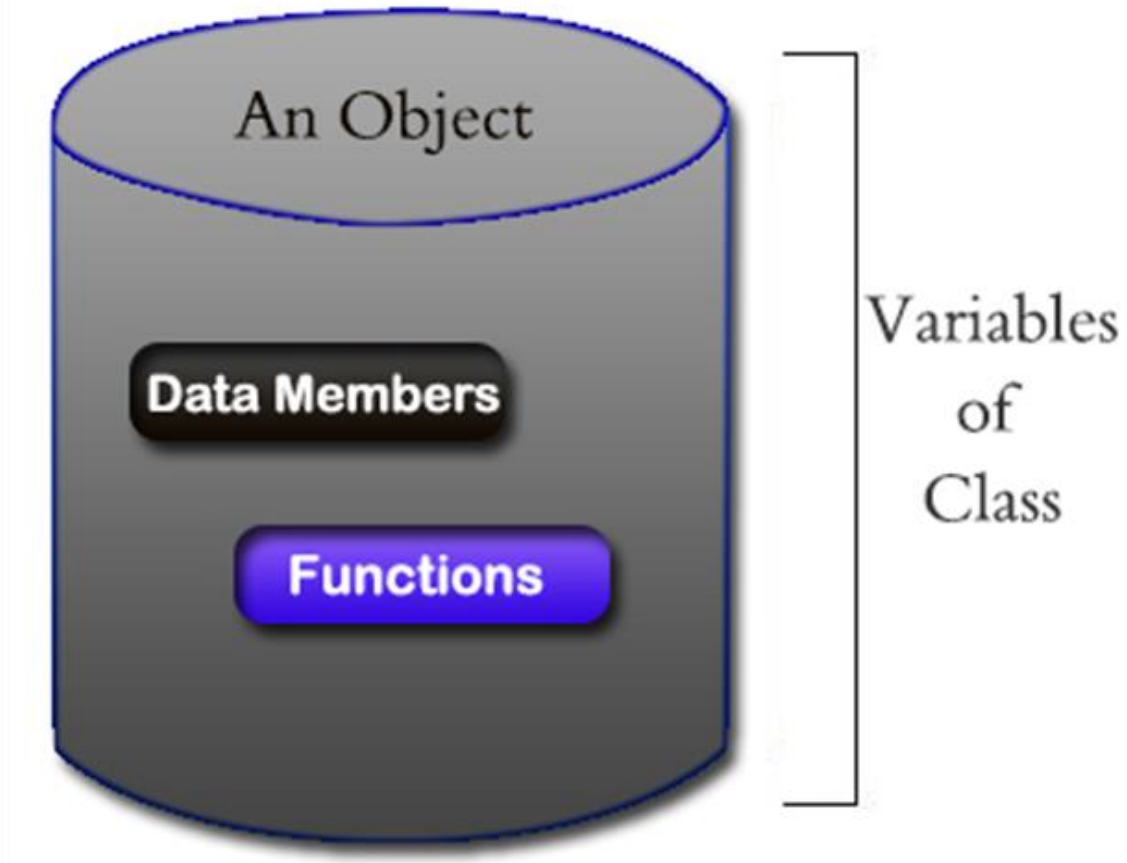
IT ACADEMY

Premenné a Objekty

python

Základy OOP


OBJECT
PROPERTY
Attribute or state
METHOD
Do something:
function & procedure


An Object
Data Members
Functions
Variables
of
Class

OOP a objekt

1. Čo je to za objekt?
2. Aké má vlastnosti?
3. Aké bude mať asi správanie?

4. Ako podľa tohto objektu vytvoríme "ideálnu" triedu a naopak?
5. Má/bude mať potomkov?
6. Koľko takýchto objektov potrebujeme/budeme potrebovať?

Triedy a Objekty

Trieda

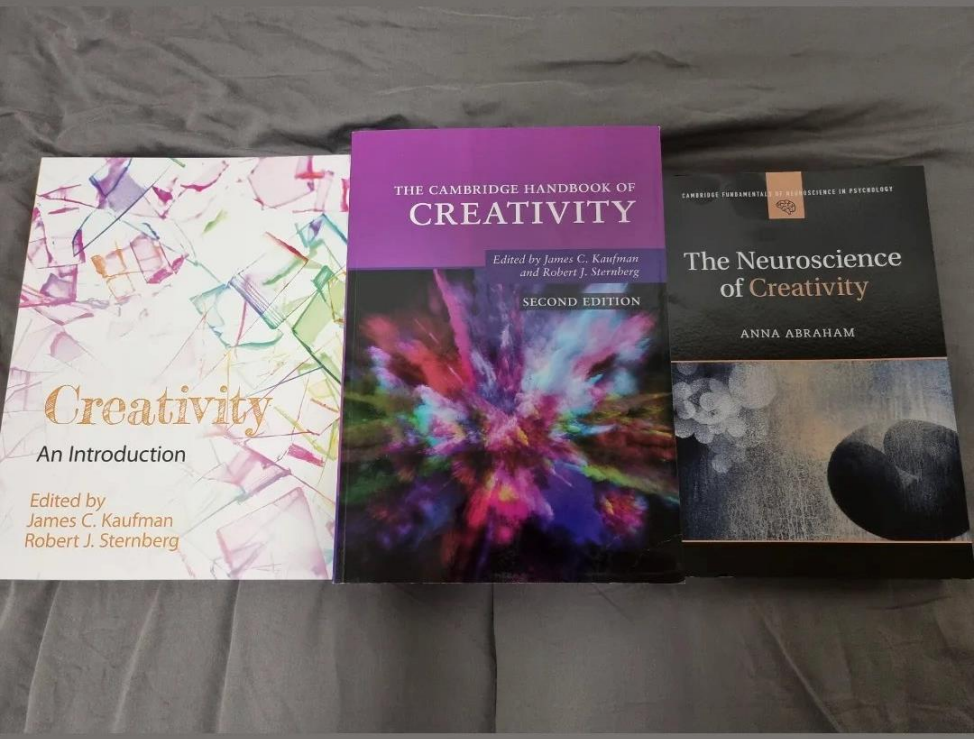

```
class Pes:
    meno = ""

    def stekaj(self):
        print("Haf")
```

Objekty (inštancie triedy Pes)

```
pes1 = Pes()
pes1.meno = "Luigi"
```

```
pes2 = Pes()
pes2.meno = "Mario"
```

OBJECT	CLASS
Object is an instance ofa class.	Class is a blue print from which objects are created
Object is a real world entity such as chair, pen. table, laptop etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.
Object is created through new keyword. ob = Employee()	Class is declared using class keyword. class Employee



OOP a Objekty

1. Čo majú tieto objekty spoločné?
2. Čo majú tieto objekty odlišné?
3. Je možné niektoré vlastnosti generalizovať(zovšeobecniť) a používať opakovane?
4. Koľko takýchto objektov potrebujeme/budeme potrebovať?

Objekt

- Vyjadriteľný podstatným meno
- Všetko v Pythone je objekt nejakého typu
- Má svoje idčko
- Zabudované metódy __method__

```
>>> type("Adam Sangala")
<type 'str'>
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__
eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__g
etnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le_
_', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '_
_reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__se
tattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_fie
ld_name_split', '_formatter_parser', 'capitalize', 'center', 'count',
'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index
', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', '
isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
>>> dir("Adam Sangala")
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__
eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__g
etnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le_
_', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '_
_reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__se
tattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_fie
ld_name_split', '_formatter_parser', 'capitalize', 'center', 'count',
'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index
', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', '
isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

Vyber čo je trieda a čo je objekt?

1. Adam Šangala
2. Softvér na tvorbu hmatových orientačných máp
3. Formulár
4. Príručka žiadateľa o grant
5. Trieda
6. Projekt
7. Seat Ibiza
8. Objekt
9. Operačný program Efektívna verejná správa
10. Žena

r/Showerthoughts
u/Stormfly • 1h

The sentence "Don't objectify women" has "women" as the object of the sentence.

Funny Mindblowing

Vote 33 Share

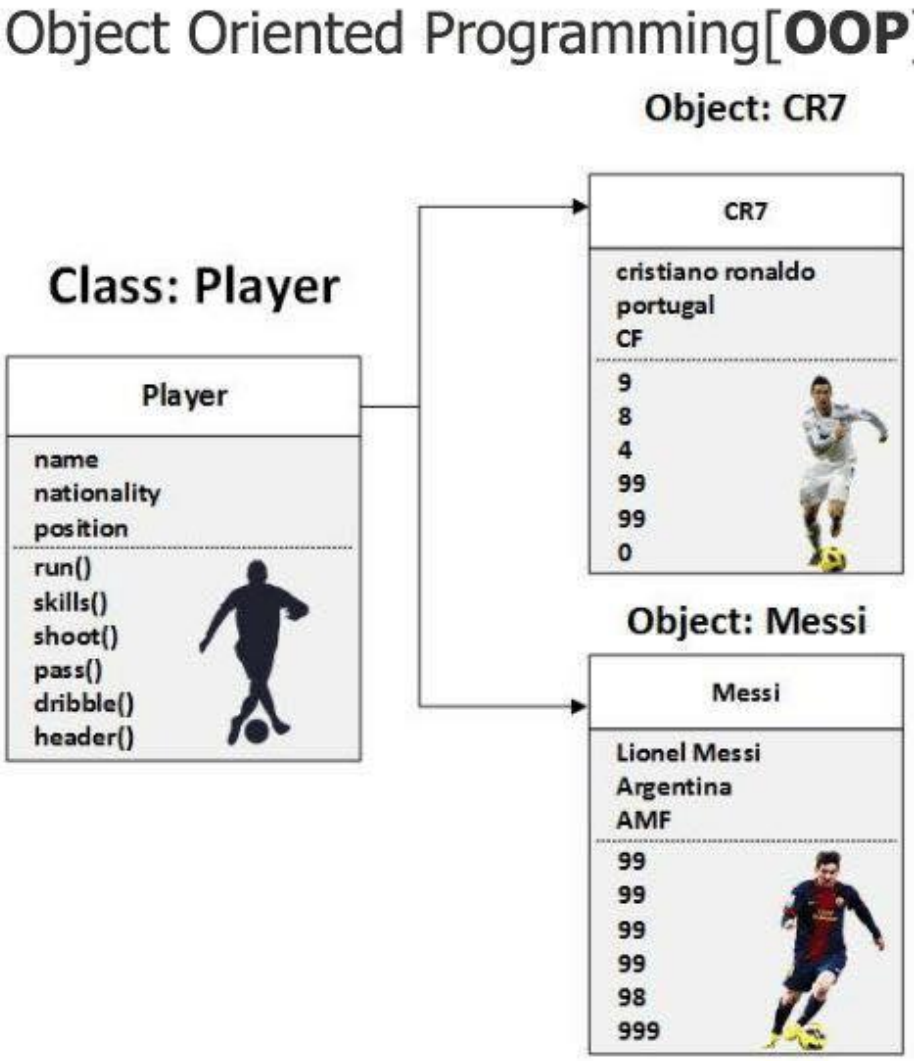
BEST COMMENTS

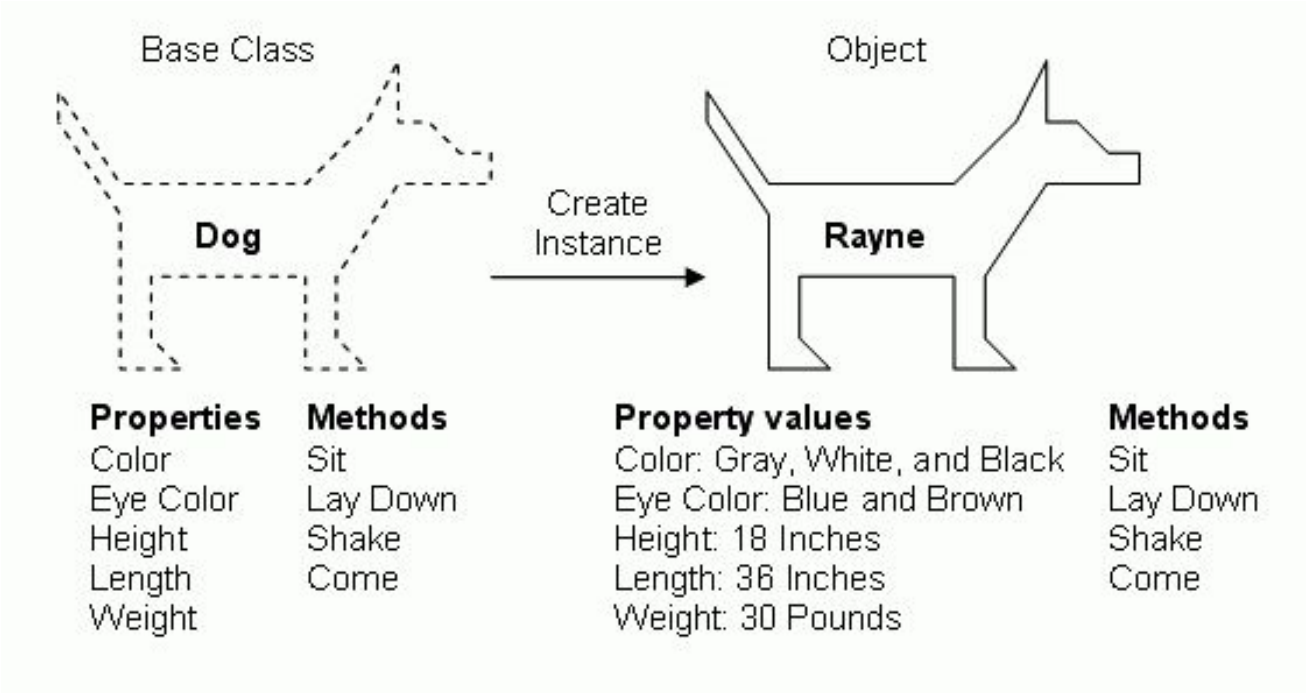
Azzarrel • 1h

Woman w = new Woman(); where is your god now?

Reply 58

Triedy a Objekty





Objektovo Orientované Programovanie

1. **Zapúzdrenie (encapsulation) - Privilegovaný selektívny prístup k dátam** daného objektu triedy, čím umožňuje **ochranu dát objektu**
2. **Dedičnosť (inheritance) -** Umožňuje vytvárať **nové triedy z už existujúcich tried** s prípadnou modifikáciou ich vlastností
3. **Mnohotvárnosť (polymorphisms) -** Umožňuje **rôzne správanie sa metód** s rovnakým názvom triedy alebo pri dedení triedy

Metódy triedy

- Verejné, chranené (_), súkromné (__)
- Funkcie definované v triede
- Definícia:

```
def metoda(self, parametre ):
    self.atribut1 = ...
    self.metoda(parametre)
...
```

- Každá metóda musí mať pri definovaní ako prvý parameter self
- Self je premenná, ktorý reprezentuje samotnú inštanciu (objekt) danej tried

```
1. # Vytvorenie instancie triedy
2. Objekt = Nazov_Triedy(...)
3. # Vytvorenie noveho
   atributu/zmena hodnoty
4. Objekt.atribut = hodnota
5. # Zavolanie metoda
6. Objekt.metoda(parametre)
7. # Zrušenie/vymazanie danej
   triedy
8. Del objekt
```

Vyber čo je trieda a čo je atribút?

Výška	Tvar	Celková zazmluvnená suma
Poskytovateľ pomoci	Popis projektu	Miesto realizácie
IČO	Merateľné ukazovatele	Dátum platnosti zmluvy
Operačný program	DPH	Formulár

Objekty
Inštancie
Triedy

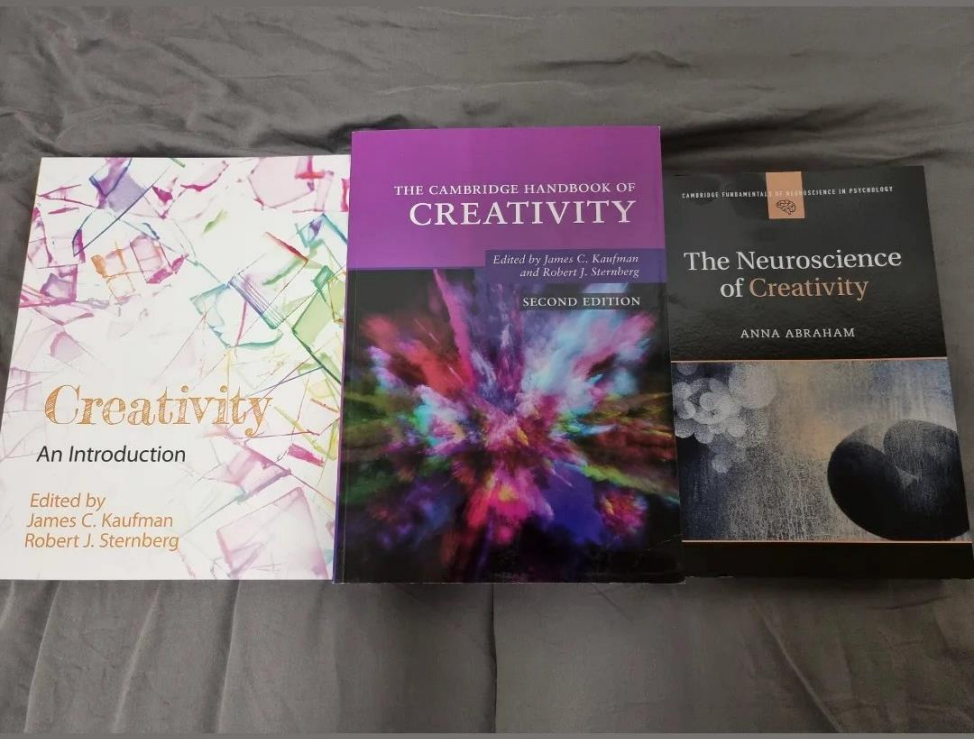
python

Objekt

- Vyjadriteľný podstatným meno
- Všetko v Pythone je objekt nejakého typu
- Má svoje idčko
- Zabudované metódy __method__

```
>>> type("Adam Sangala")
<type 'str'>
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__
eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__g
etnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le_
_', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '_
_reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__se
tattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_fie
ld_name_split', '_formatter_parser', 'capitalize', 'center', 'count',
'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index
', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', '
isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
>>> dir("Adam Sangala")
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__
eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__g
etnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le_
_', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '_
_reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__se
tattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_fie
ld_name_split', '_formatter_parser', 'capitalize', 'center', 'count',
'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index
', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', '
isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

OOP a Objekty



1. Čo majú tieto objekty spoločné?
2. Čo majú tieto objekty odlišné?
3. Je možné niektoré vlastnosti generalizovať(zovšeobecniť) a používať opakovane?
4. Koľko takýchto objektov potrebujeme/budeme potrebovať?

Princíp identity a funkcia id()

Úplna identita:

- Objekt.atribut/metoda
- Kto/Čo . Čo chcem dostať/vykonať
- Karol.zaplat()
- Karol.vek

Anonymná identita:

- Neuvediem meno objektu . Čo chcem dostať/vykonať
- zaplat()

- Vracia **celé číslo** (alebo dlhé celé číslo), pre ktoré je zaručené, že je **jedinečné** a **konštantné** pre tento objekt počas jeho životnosti
- Podobné ako v jazyku C adresa objektu v pamäti
- Používanie v **operátore is a is not**

Používanie id()


```
1 a = 1
2 b = a
3 c = b
4 d = 1
5
6 print id(a)
7 print id(b)
8 print id(c)
9 print id(d)
```

```
40928888
40928888
40928888
40928888
```

Operátor is a in

Testovanie identity

```
>>> a = 'pub'
>>> b = ''.join(['p', 'u', 'b'])
>>> a == b
True
>>> a is b
False
```

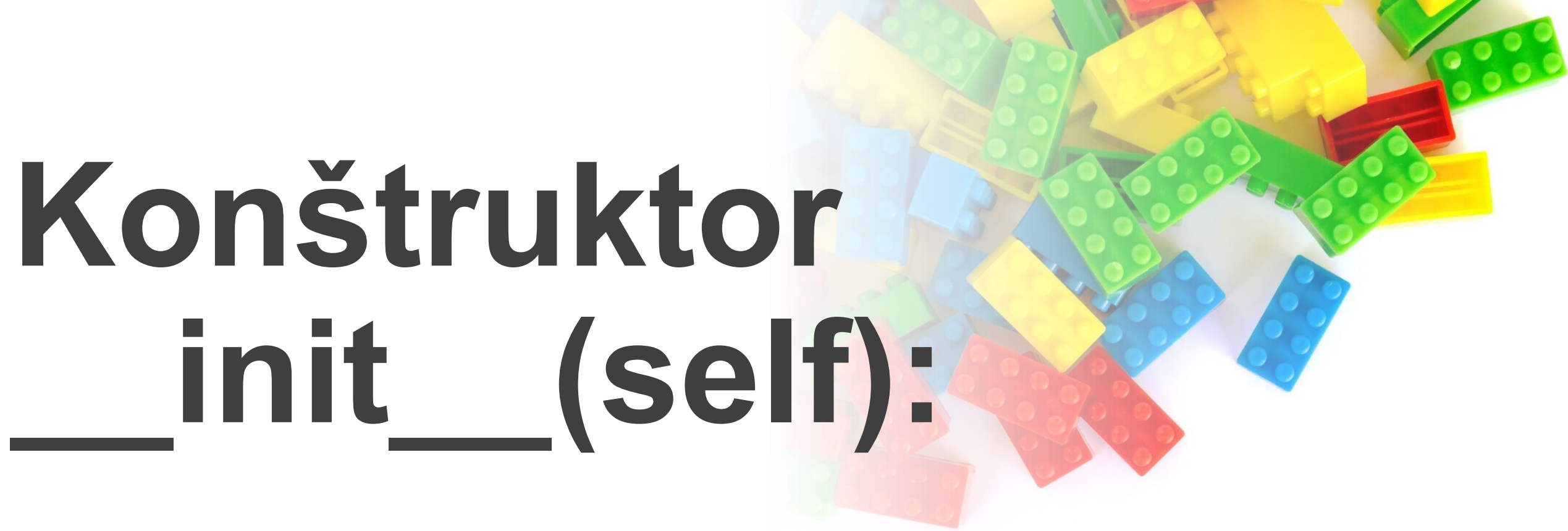
Testovanie začlenenia

```
meno = 'laco'
'a' in meno
'x' in meno

rodina = ['mama','otec','brat']
'sestra' in rodina
'mama' in rodina
```

```
obchod=['chleba','mlieko','maso','maslo','ryza']

for jedlo in obchod:
    print "chcem " + jedlo
```

is je id(a) == id(b)

Konštruktor __init__(self):



python

Konštruktor

- Používajú sa na **vytváranie inštancií objektu**
- Úlohou konštruktorov je **inicializovať (priradiť hodnoty) dátovým členom triedy**, keď je vytvorený objekt triedy
- V Pythone sa metóda __init__() nazýva konštruktor a volá sa vždy, keď je vytvorený objekt

```
def __init__(self):
    # telo konštruktora
```

Typy Konštruktorov

Predvolený (Defaultný)

- Jednoduchý konštruktor, ktorý neprijíma žiadne argumenty
- Jeho definícia má iba 1 argument, ktorý je odkazom na konštruovanú inštanciu self

Parametrizovaný

- Konštruktor s parametrami je známy ako parametrizovaný konštruktor.
- Parametrizovaný konštruktor berie svoj 1 argument ako odkaz na konštruovanú inštanciu známu ako self a ostatné argumenty poskytuje programátor.

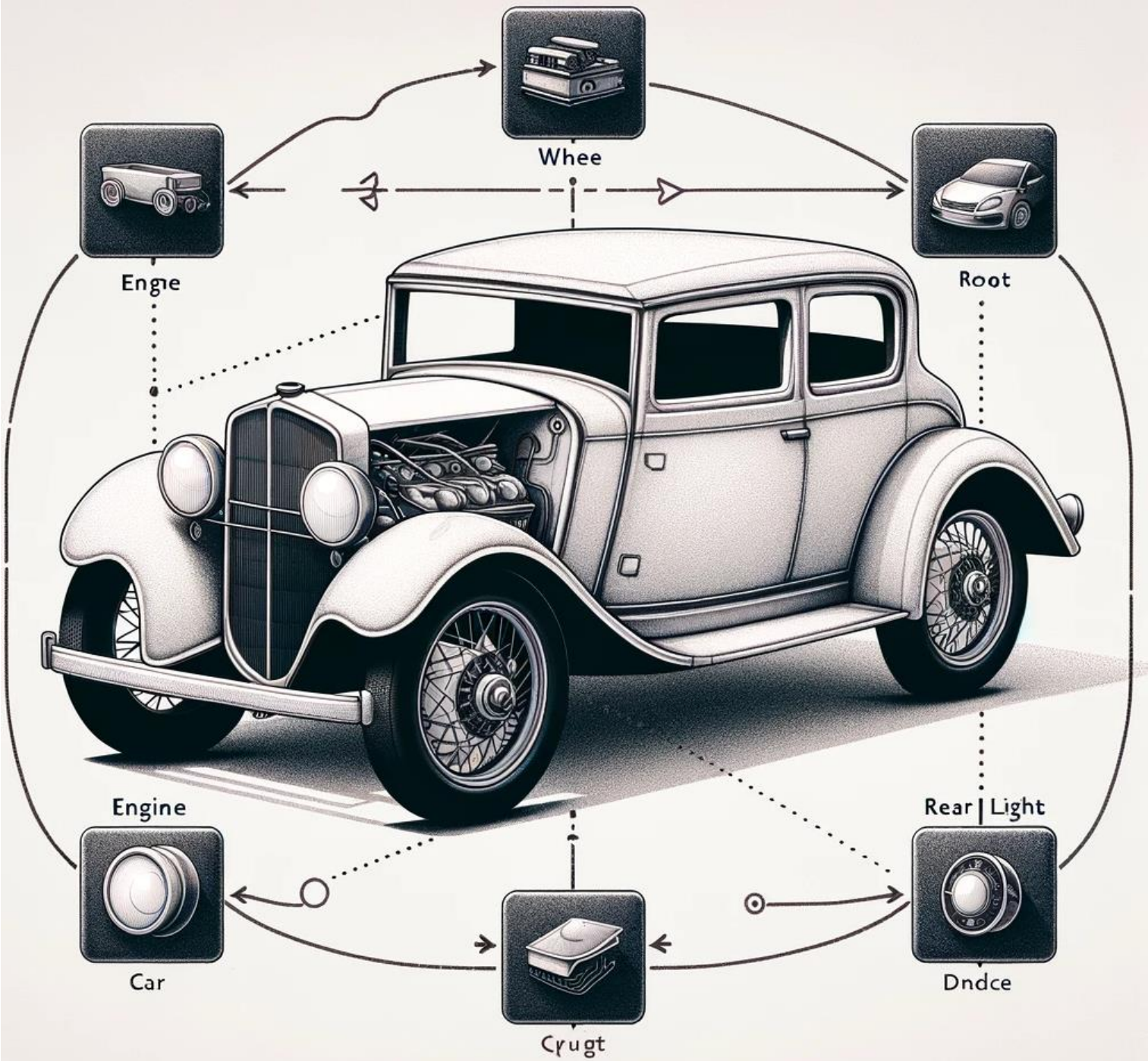
Kompozícia

python

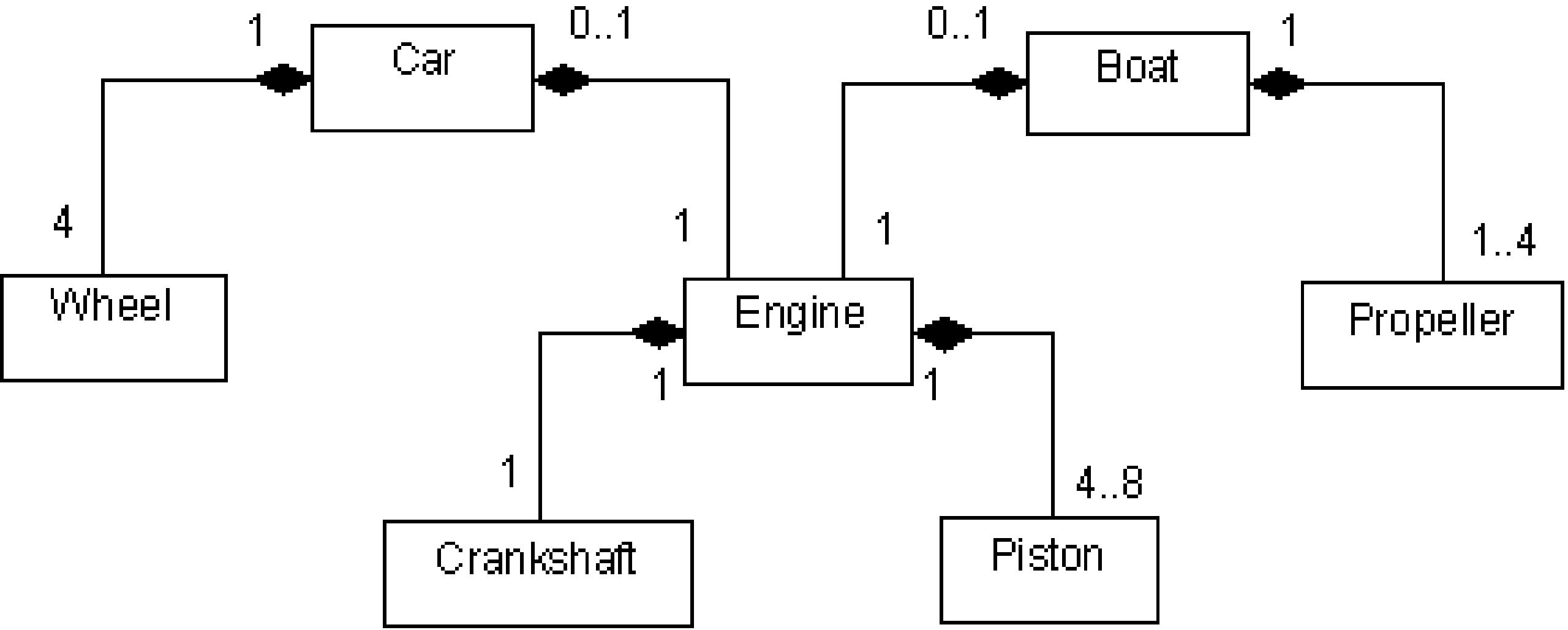
Kompozícia

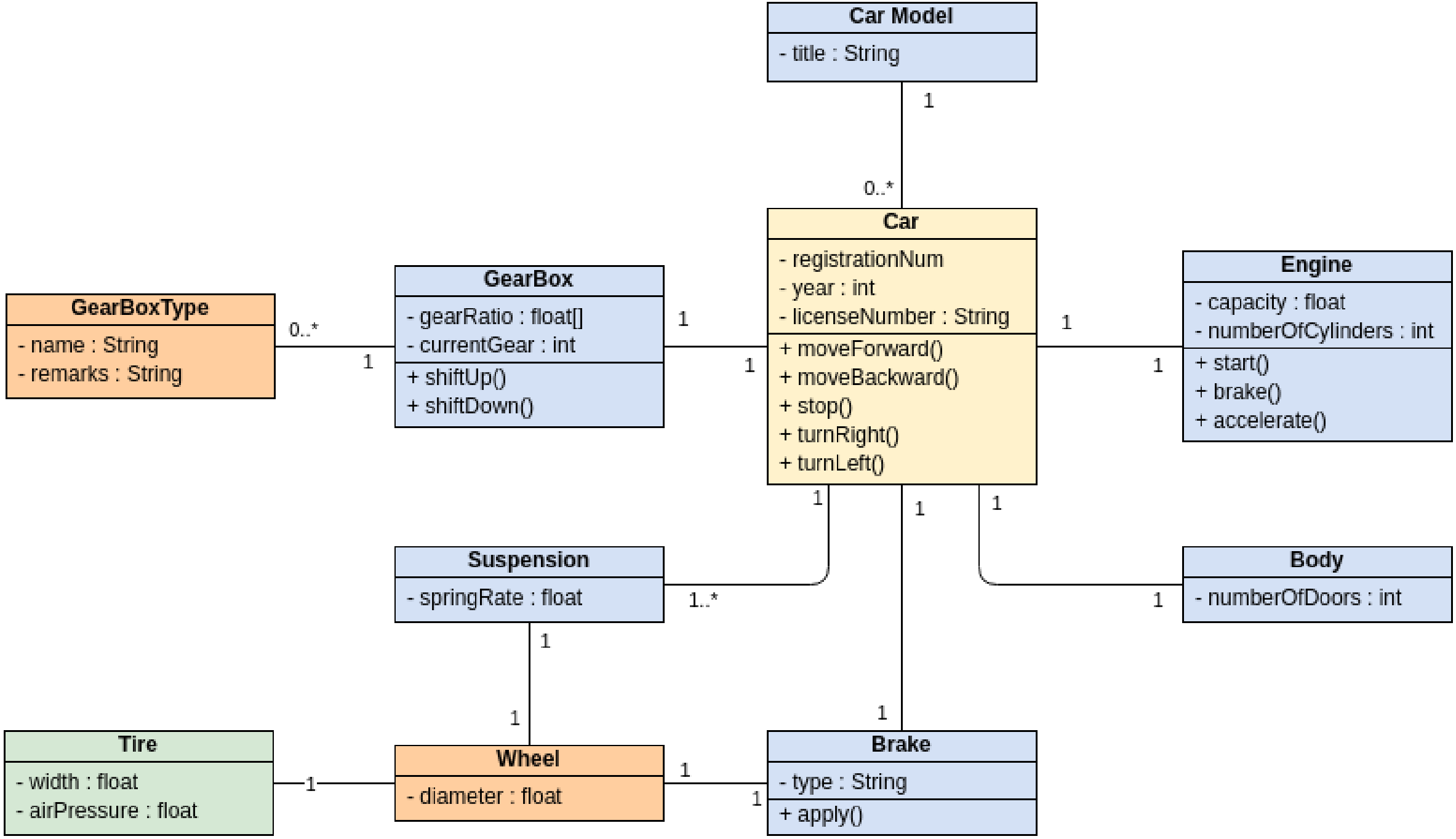
- **Dôležitým princípom OOP**
- Umožňuje **vytváranie zložitejších štruktúr** z **jednoduchších objektov**
- **Umožňuje triede získavať metódy iných tried prostredníctvom ich inštancií**
- Pri kompozícii **obsahuje trieda objekty iných tried ako svoje atribúty**, čím **nadobúda ich metódy**
- Uumožňuje väčšiu **flexibilitu** a **opätovnú použiteľnosť kódu**, pretože **zmeny v jednej triede nemusia nutne ovplyvniť triedy, ktoré ju používajú**

Kompozícia



Kompozícia UML

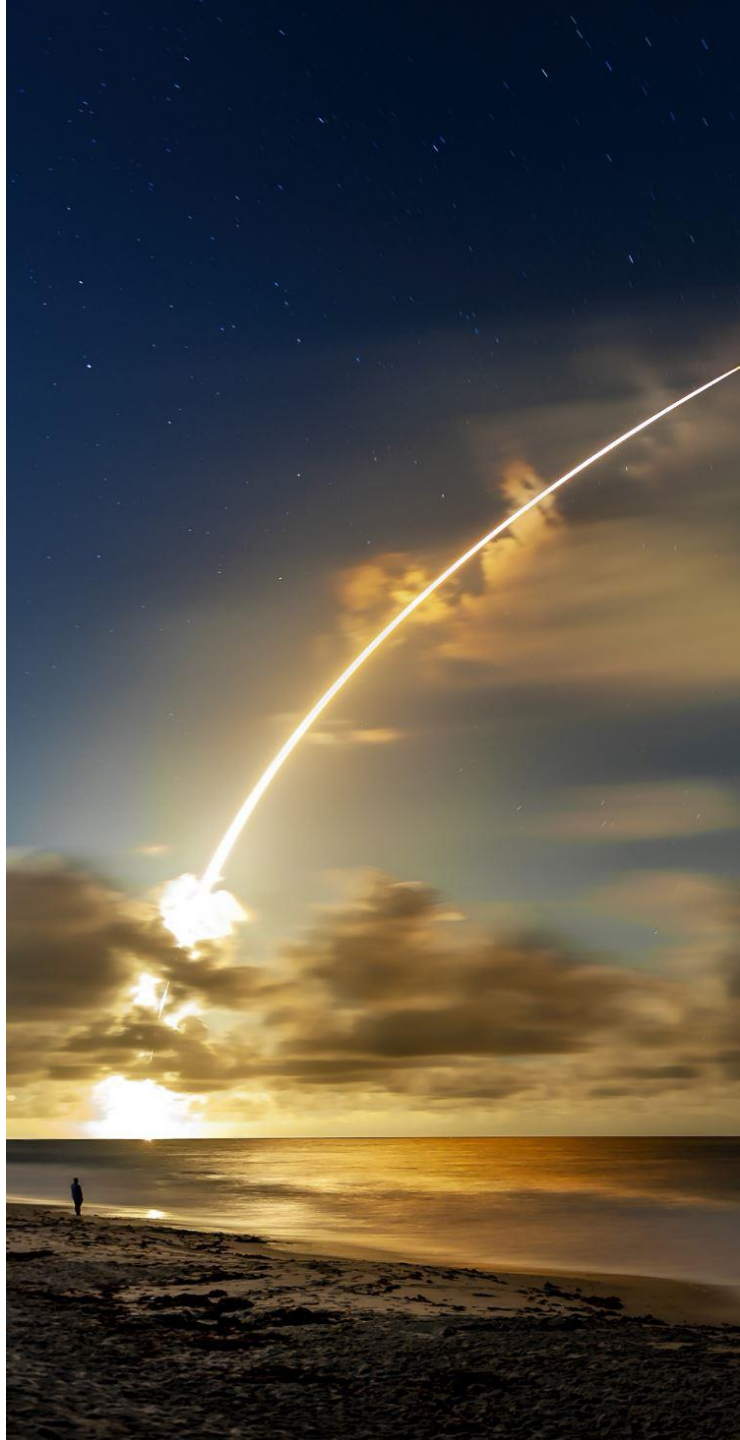



Car Model
- title : String
1
0..*
Car
- registrationNum
- year : int
- licenseNumber : String
+ moveForward()
+ moveBackward()
+ stop()
+ turnRight()
+ turnLeft()
GearBoxType
- name : String
- remarks : String
0..*
1
GearBox
- gearRatio : float[]
- currentGear : int
+ shiftUp()
+ shiftDown()
1
1
1
1
Engine
- capacity : float
- numberOfCylinders : int
+ start()
+ brake()
+ accelerate()
1
1
1
1
1
Suspension
- springRate : float
1..*
1
Body
- numberOfDoors : int
1
1
1
Tire
- width : float
- airPressure : float
1
Wheel
- diameter : float
1
1
Brake
- type : String
+ apply()

```python
class Motor:
    def __init__(self, vykon):
        self.vykon = vykon
    def start(self):
        print(f"Motor s výkonom {self.vykon} konských síl štartuje.")
class Auto:
    def __init__(self, model, motor):
        self.model = model
        self.motor = motor  # Kompozícia: Auto obsahuje motor
    def start(self):
        print(f"Auto modelu {self.model} štartuje.")
        self.motor.start()  # Volanie metódy motora
# Vytvorenie inštancie motora
motor = Motor(150)
# Vytvorenie inštancie auta s daným motorom
auto = Auto("Škoda Octavia", motor)
# Štartovanie auta
auto.start()
```

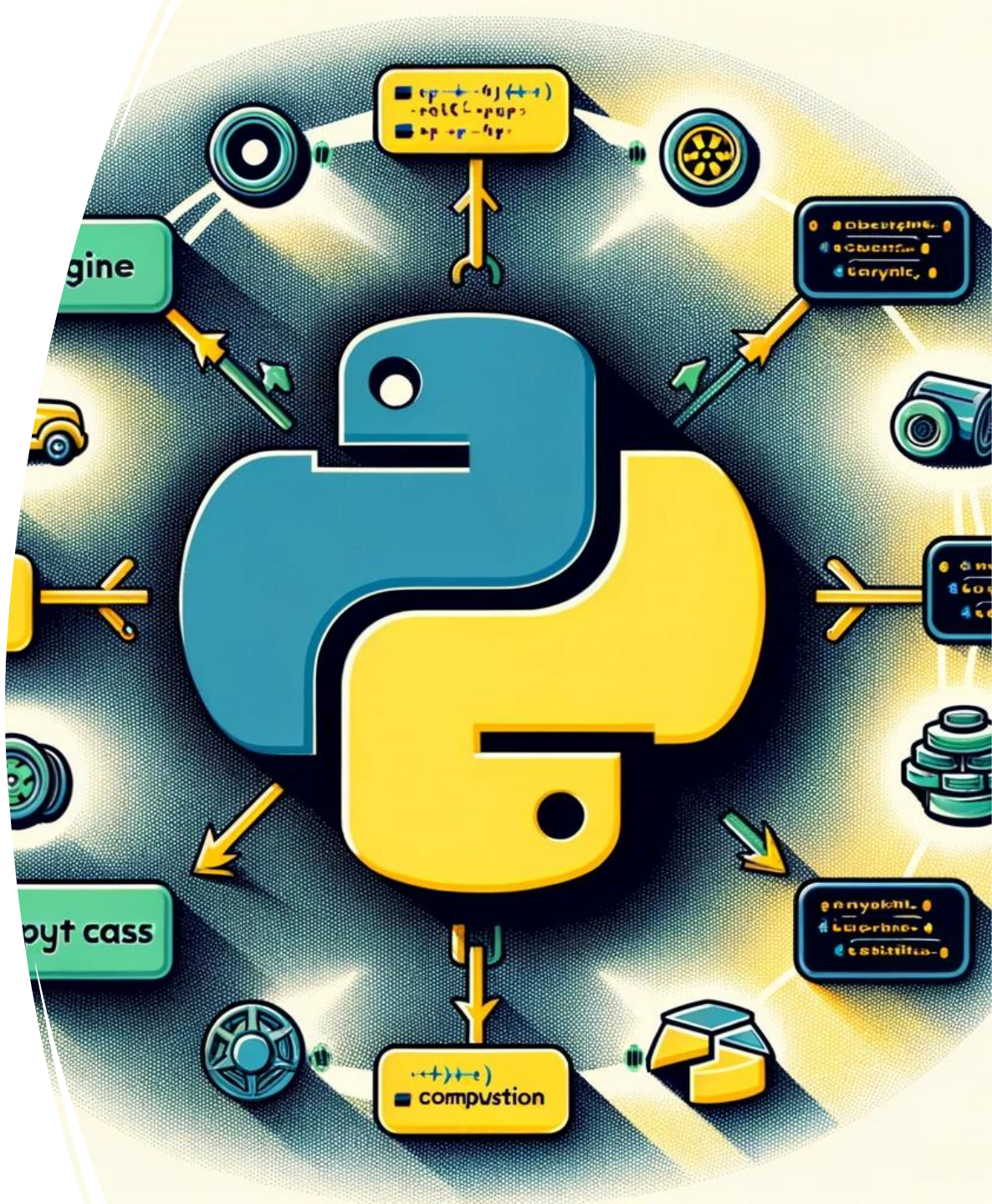
Výhody Kompozície

1. **Flexibilita**: Kompozícia umožňuje vytvárať flexibilnejšie dizajny, pretože **môžete meniť komponované objekty za behu programu**.
2. **Uzamknutie**: Keď použijete kompozíciu, môžete **uzamknúť určité aspekty vašej triedy**, aby neboli **prístupné zvonka**.
3. **Zníženie závislosti**: Vďaka kompozícii sú **triedy menej závislé jedna od druhej**.
4. **Znovupoužiteľnosť kódu**: **Komponenty**, ktoré sú **navrhnuté** ako **samostatné**, **opätovne použiteľné moduly**, **môžu byť ľahko použité v rôznych častiach** aplikácie alebo **dokonca v iných projektoch**.



Úlohy
Kompozícia

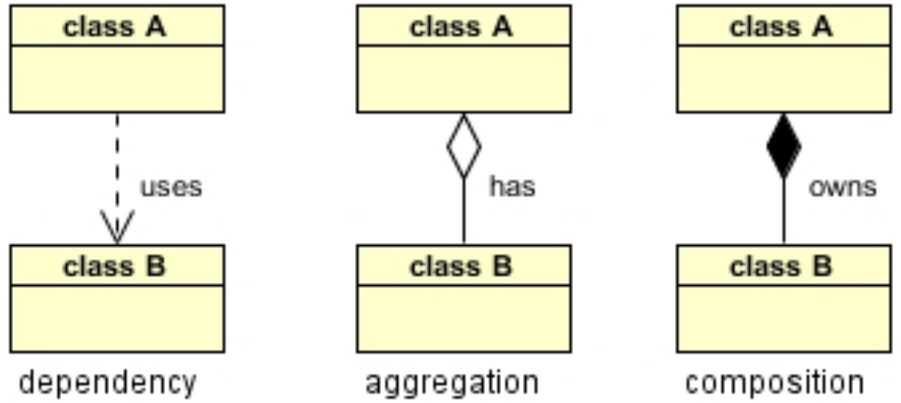
1. Pochopenie kompozície
2. Používanie kompozície
3. Osvojenie si výhod kompozície

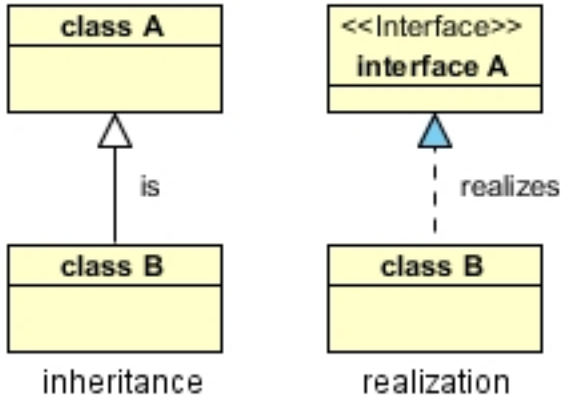


UML
Vzťahy

Diagram Tried (Class Diagrams)

- Popisuje rôzne typy objektov, ktoré existujú v systéme a vzťahy medzi nimi
- Jeden z **najviac užitočných** a bežne používaných **diagramov**
- Umožňuje plánovať akú funkcionalitu majú mať objekty/triedy a ako sa ovplyvňujú
- **Triedy** majú **atribúty** a **metódy**





Diagram, ktorému rozumie programátor

Diagram Tried (Class Diagrams)

Označenie tried:

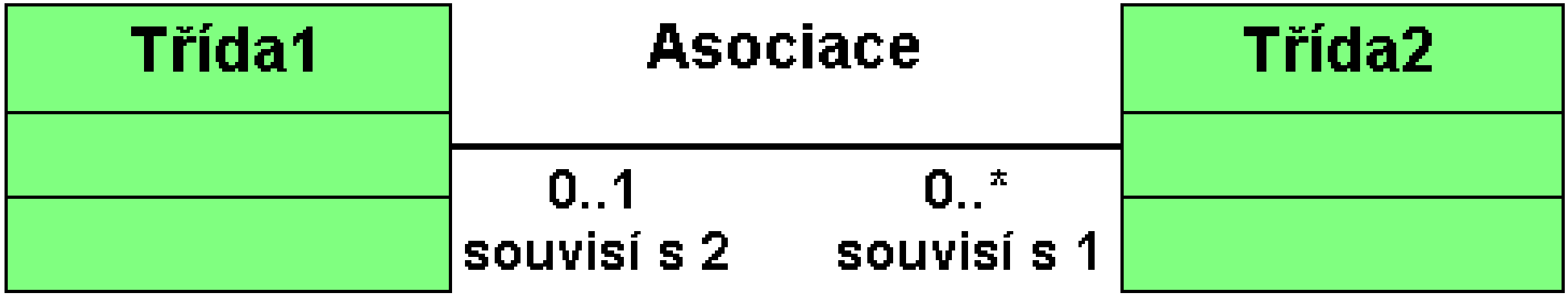
- **+ public** class
- **- private** class
- **# protected** class
- **a abstract** class

Dátové zložky

Metódy

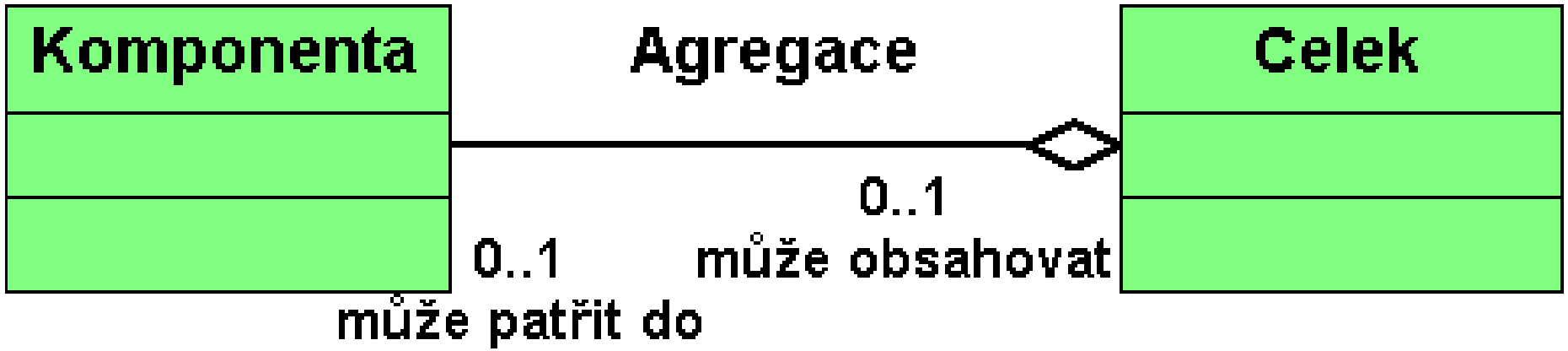
Asociácia

- **Asociácia** - **vzťah medzi dvoma triedami**
- **Reprezentácia** pomocou **čiary**



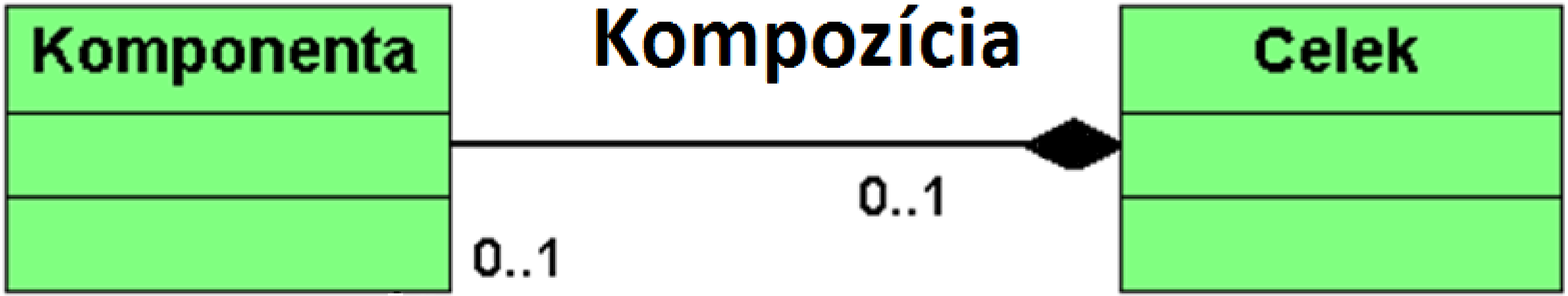
Agregácia

- **Vyjadruje vzťah celok - časť** forma asociácie, ktorá
- **Element môže „prežiť“** svoj **kontajner**, príp. **stať sa súčasťou iného kontejneru**



Kompozícia

- **Silnejšia väzba ako agregácia - zrušením kontejneru automaticky zrušíme aj element, ktorý ho obsahuje**
- Daný element **môže byť súčasťou práve jedného kontajneru**

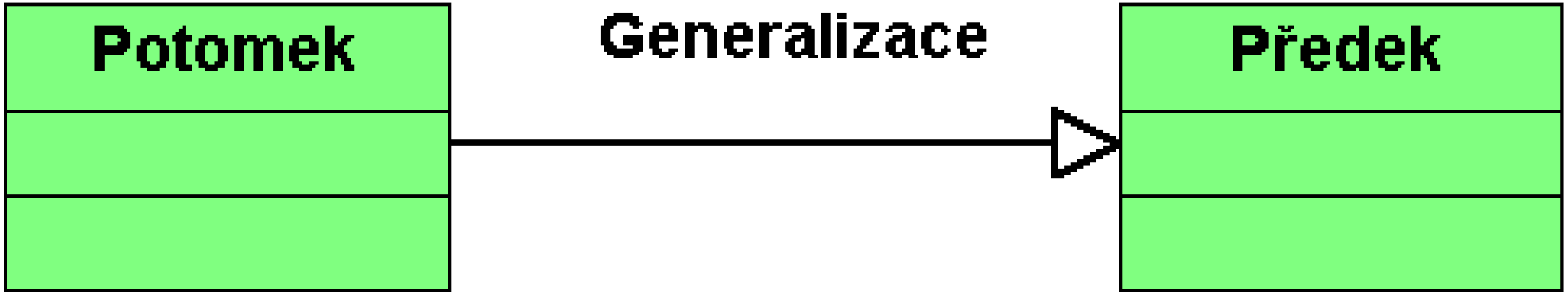


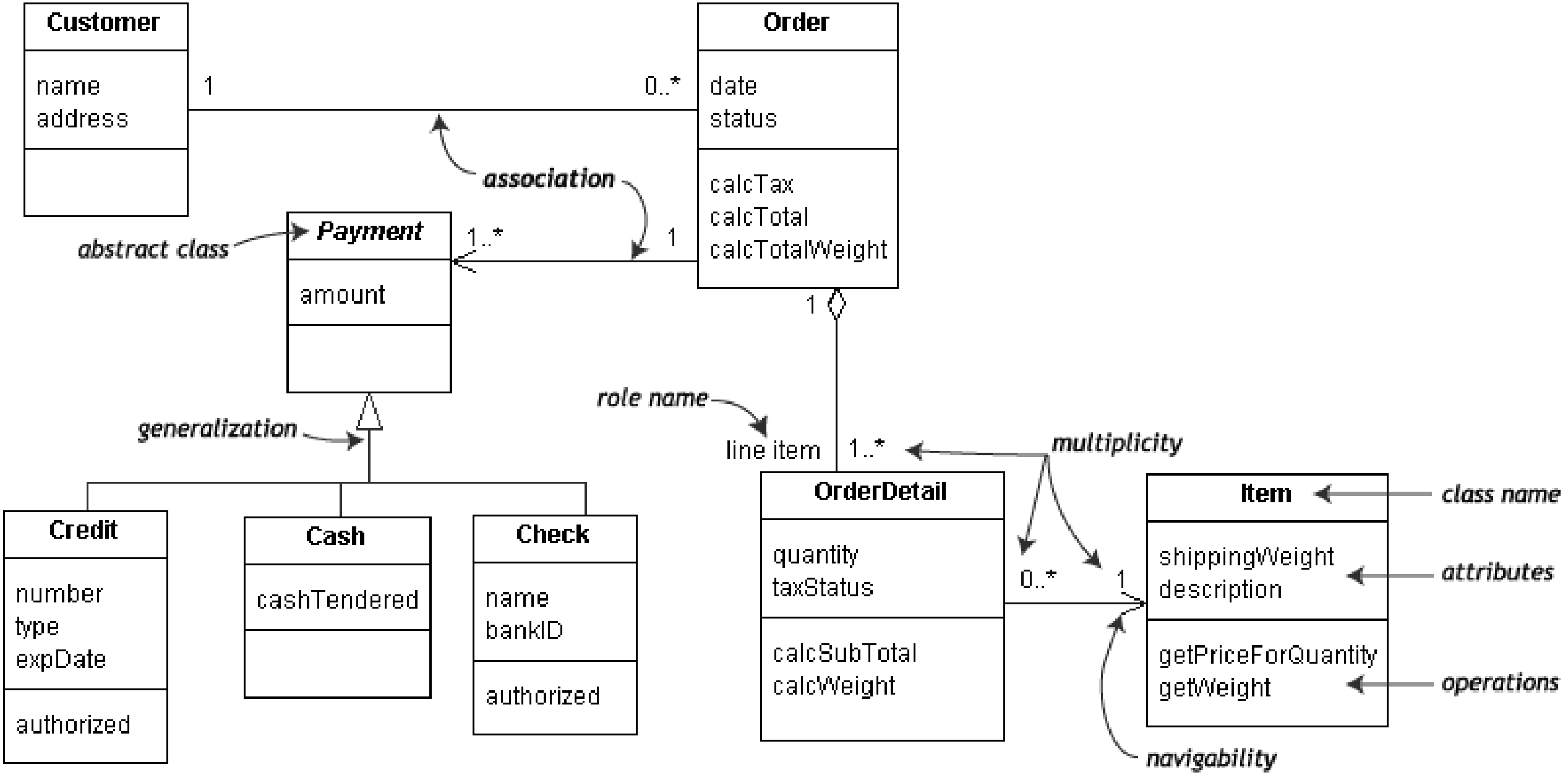
Závislosť

- Znázorňuje závislosť medzi triedami
- **Zmenou parametra v prvej triede nastane zmena aj v druhej**

Dedičnosť

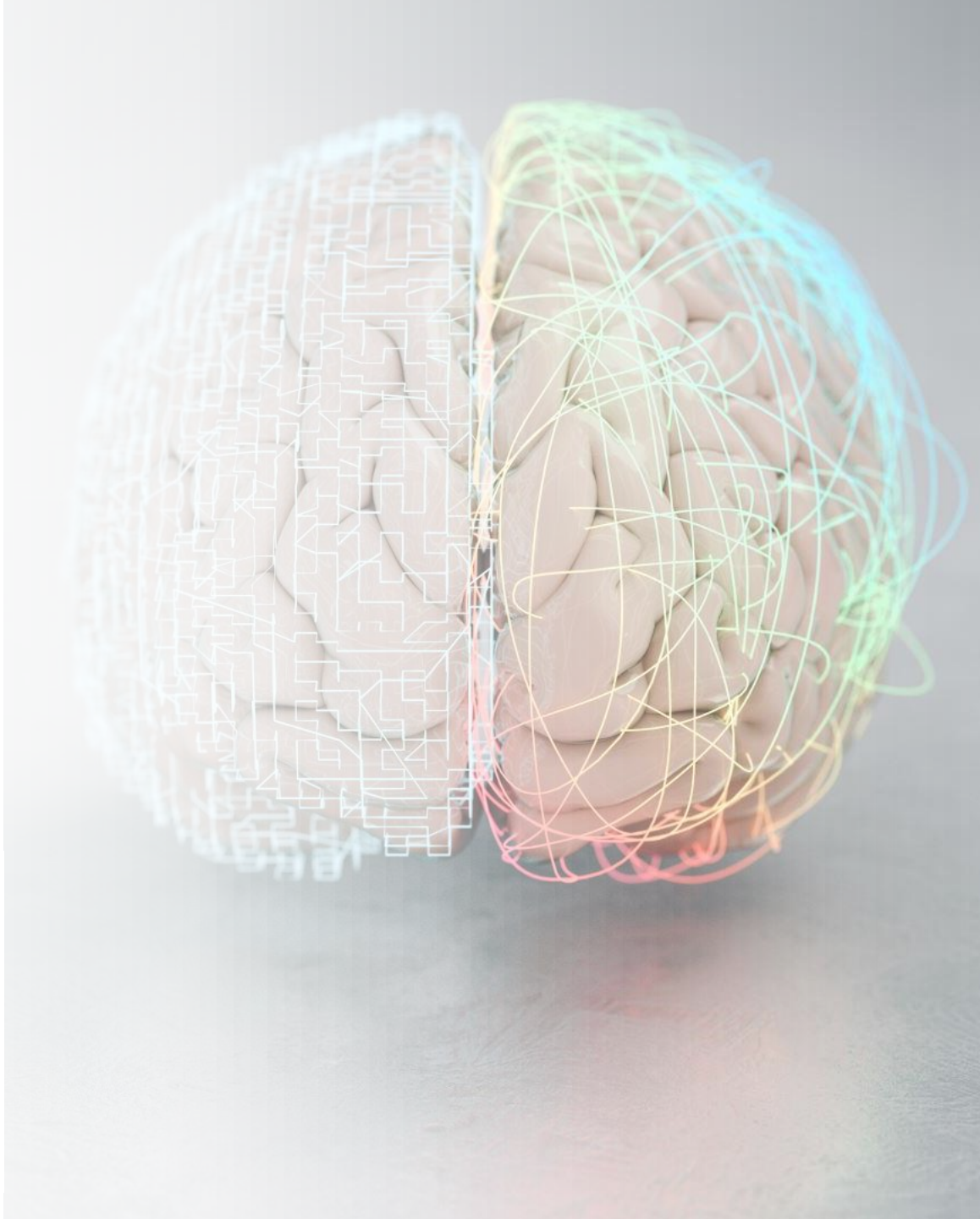
- Dedičnosť - **šípka** vedie **k rodičovskej triede**




Customer
name
address
1
0..*
Order
date
status
calcTax
calcTotal
calcTotalWeight
association
1
1..*
abstract class
Payment
amount
1
role name
line item
1..*
multiplicity
generalization
Credit
number
type
expDate
authorized
Cash
cashTendered
Check
name
bankID
authorized
OrderDetail
quantity
taxStatus
calcSubTotal
calcWeight
0..*
1
Item
shippingWeight
description
getPriceForQuantity
getWeight
class name
attributes
operations
navigability

Umelá Inteligencia AI a OOP

python



AI Generovanie Zdrojového Kódu

AI Modify: doplň dáta a metódy

```
# doplň dáta a metódy
# trieda lev
class lev:
    def __init__(self):
        self.data = []
    def __init__(self, name=""):
        self.name = name     # property 'name' for the lion

    def roar(self):          # method 'roar' prints a message
        print(f'The lion {self.name} roars!')

    # Add your class methods and properties here
```

Accept | Regenerate | Reject

Generate: trieda lev

```python
# trieda lev
class lev:
    def __init__(self, health=100, hunger=0):
        self.health = health
        self.hunger = hunger

    def feed(self, food):
        if food >= self.hunger:
            self.hunger = 0
            print("Lev is full.")
        else:
            self.hunger -= food
            print("Lev ate something but is still hungry.")

    def attack(self, damage):
        if self.health <= 0:
            print("Lev is already unconscious.")
        else:
            self.health -= damage
            if self.health <= 0:
                print("Lev is unconscious after the attack.")
            else:
                print(f"Lev is attacked, current health: {self.health}")

    def check_status(self):
        print(f"Lev's current health: {self.health}")
        print(f"Lev's current hunger: {self.hunger}")
```

Accept | Regenerate | Reject

Generate: vygeneruj triedu pre bankomat aj s dátami a metódami

```python
# vygeneruje triedu pre bankomat aj s dátami a metódami
class ATM:
    def __init__(self, balance=0):
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        return self.balance

    def withdraw(self, amount):
        if amount > self.balance:
            return "Insufficient balance"
        self.balance -= amount
        return self.balance

    def check_balance(self):
        return self.balance
```

Accept | Regenerate | Reject

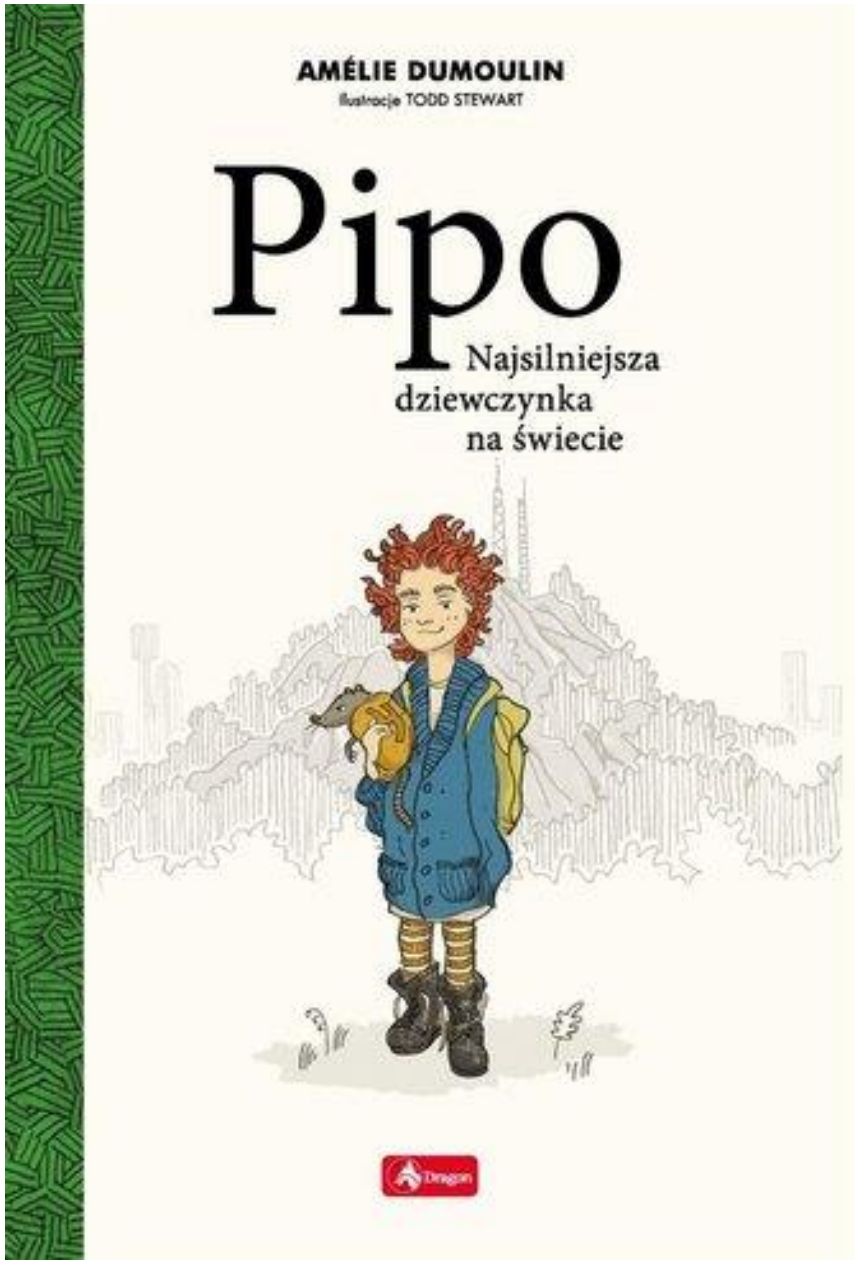
Modify: uprav jazyk na slovenský

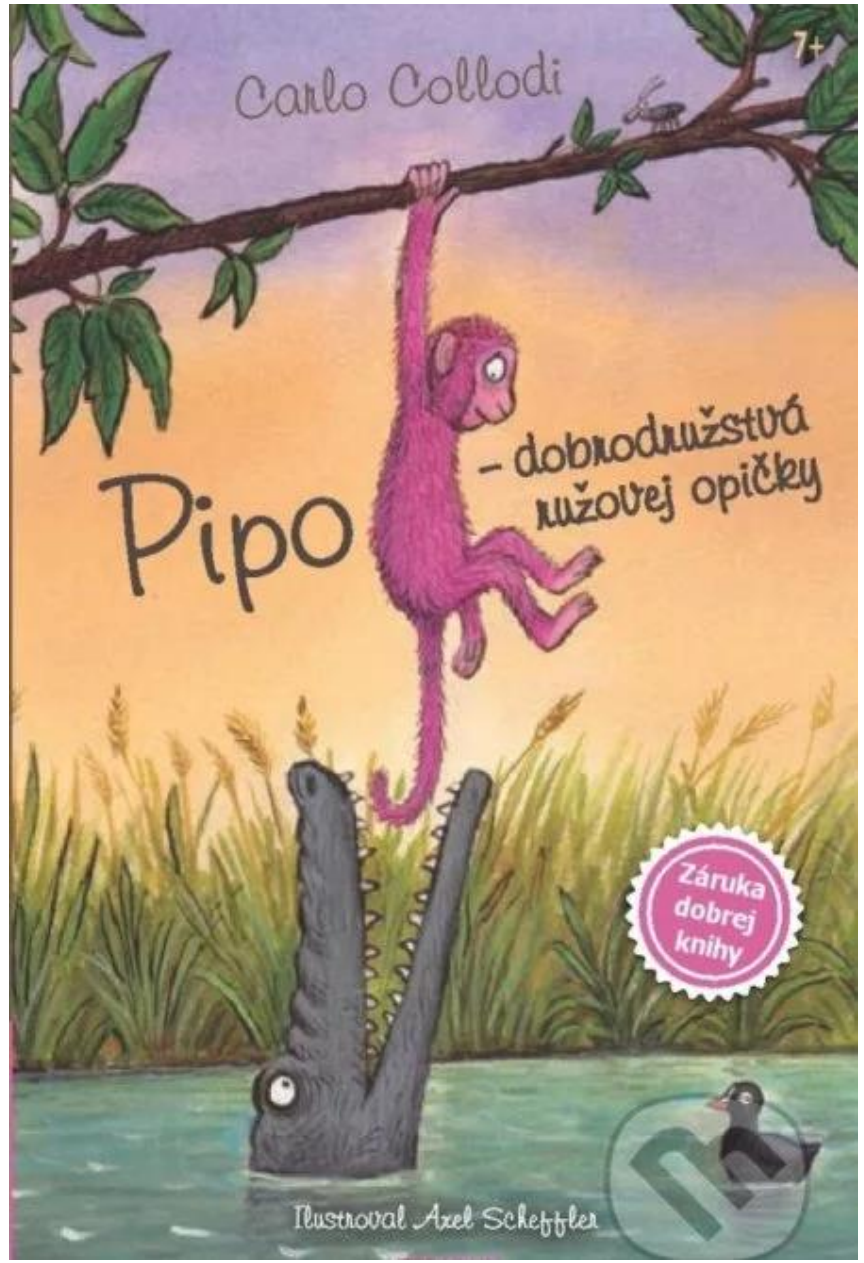
```python
# vygeneruje triedu pre bankomat aj s dátami a metódami
class ATM:
    def __init__(self, balance=0):
        self.balance = balance
# uprav jazyk na slovenský
# Trieda pre bankomat s dátami a metódami
class Bankomat:
    def __init__(self, zostatok=0):
        self.zostatok = zostatok

    def deposit(self, amount):
        self.balance += amount
        return self.balance
    def vklad(self, suma):
        self.zostatok += suma
        return self.zostatok

    def withdraw(self, amount):
        if amount > self.balance:
            return "Insufficient balance"
        self.balance -= amount
        return self.balance
    def vyber(self, suma):
        if suma > self.zostatok:
            return "Nedostatočný zostatok"
        self.zostatok -= suma
        return self.zostatok

    def check_balance(self):
        return self.balance
    def skontroluj_zostatok(self):
        return self.zostatok
```

```python
# uprav jazyk na slovenský
# Trieda pre bankomat s dátami a metódami
class Bankomat:
    def __init__(self, zostatok=0):
        self.zostatok = zostatok

    def vklad(self, suma):
        self.zostatok += suma
        return self.zostatok

    def vyber(self, suma):
        if suma > self.zostatok:
            return "Nedostatočný zostatok"
        self.zostatok -= suma
        return self.zostatok

    def skontroluj_zostatok(self):
        return self.zostatok

```

Čo je PIP?

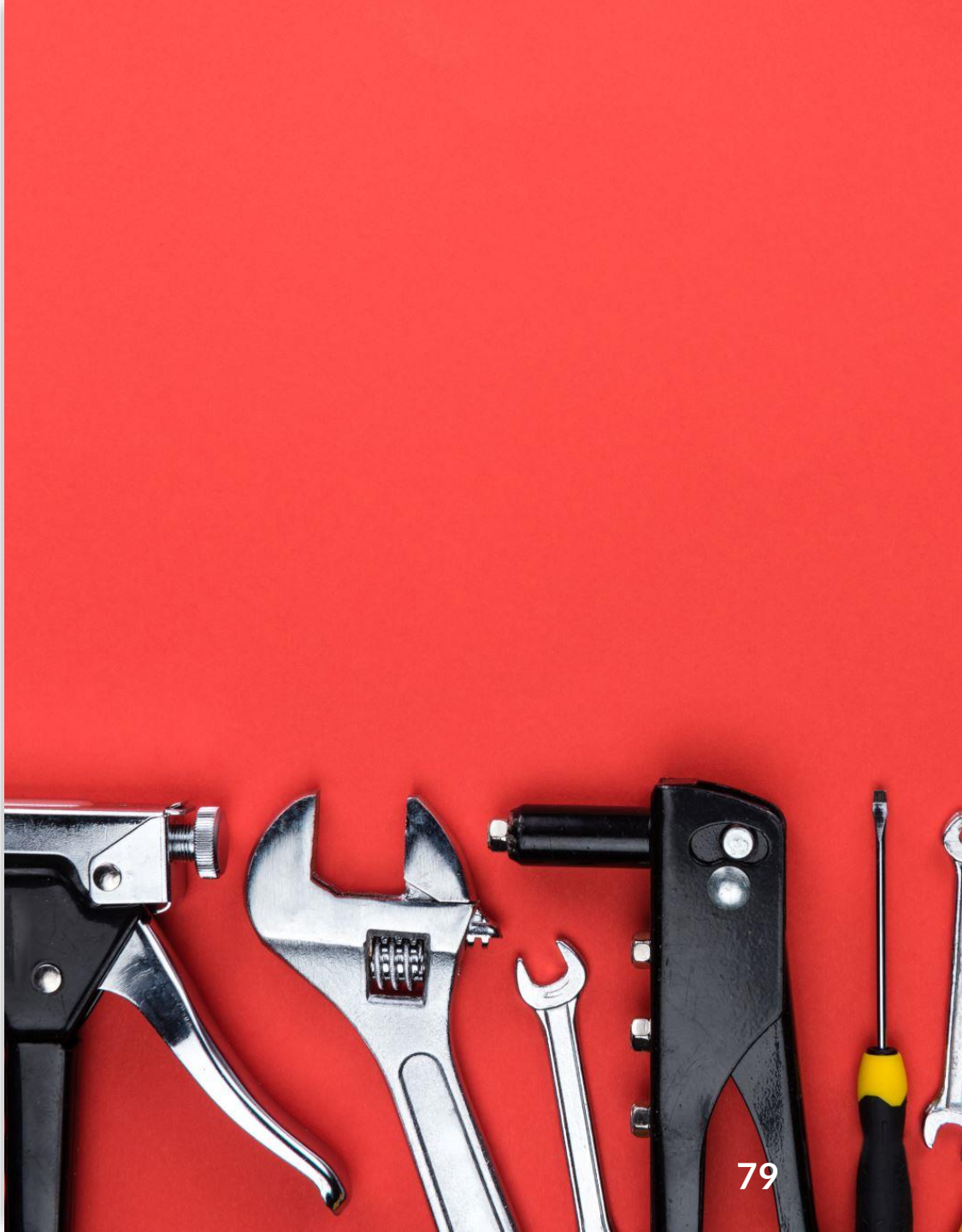

Pipo
a Pogo
MLADÉ
LETÁ


AMÉLIE DUMOULIN
Ilustracje TODD STEWART
Pipo
Najsilniejsza
dziewczynka
na świecie
Dragon


Carlo Collodi
7+
Pipo
– dobrodružstvá
ružovej opičky
Záruka
dobrej
knihy
Ilustroval Axel Scheffler

PIP Package Installer for Python

- Správca balíčkov (Package Manager) pre moduly programovacieho jazyka Python
- Jeho hlavným repozitárom je PyPI
- Názov je anglickou rekurzívnou skratkou Pip installs packages (doslova Pip inštaluje balíčky)
- Klientský program je napísaný v Pythone a uvoľnený pod licenciou MIT
- Ponúka rozhranie príkazového riadka, pričom základným použitím je volanie typu



pip

pip is the package installer for Python. You can use it to install packages from the Python Package Index and other indexes.

If you want to learn about how to use pip, check out the following resources:

- Getting Started
- Python Packaging User Guide

If you find bugs, need help, or want to talk to the developers, use our mailing lists or chat rooms:

- GitHub Issues
- Discourse channel
- User IRC
- Development IRC

If you find any security issues, please report to security@python.org

Next
Getting Started

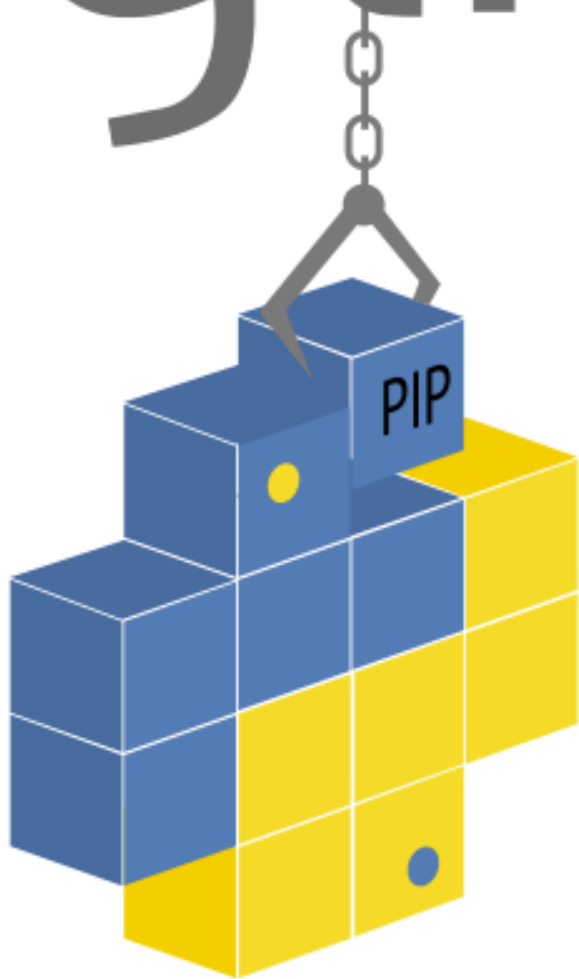
C:\Windows\System32\cmd.exe - pip install matlib

```
Microsoft Windows [Version 10.0.16299.785]
(c) 2017 Microsoft Corporation. Všetky práva vyhradené.

C:\Windows\System32>cd C:\Program Files (x86)\Python37-32\Scripts

C:\Program Files (x86)\Python37-32\Scripts>pip instal xlwt
ERROR: unknown command "instal" - maybe you meant "install"

C:\Program Files (x86)\Python37-32\Scripts>pip install xlwt
Collecting xlwt
  Downloading https://files.pythonhosted.org/packages/44/48/def306413b25c3d01753603b1a222a011b8621aed27cd7f89cbc27e6b0f4/xlwt-1.3.0-py2.py3-none-any.
whl (99kB)
    100% |████████████████████████████████| 102kB 826kB/s
Installing collected packages: xlwt
Could not install packages due to an EnvironmentError: [WinError 5] Access is denied: 'c:\\program files (x86)\\python37-32\\Lib\\site-packages\\xlwt
'
Consider using the `--user` option or check the permissions.

You are using pip version 10.0.1, however version 18.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Program Files (x86)\Python37-32\Scripts>pip install matlib
Collecting matlib
```

python™ Package Index

PIP a easy install

python™

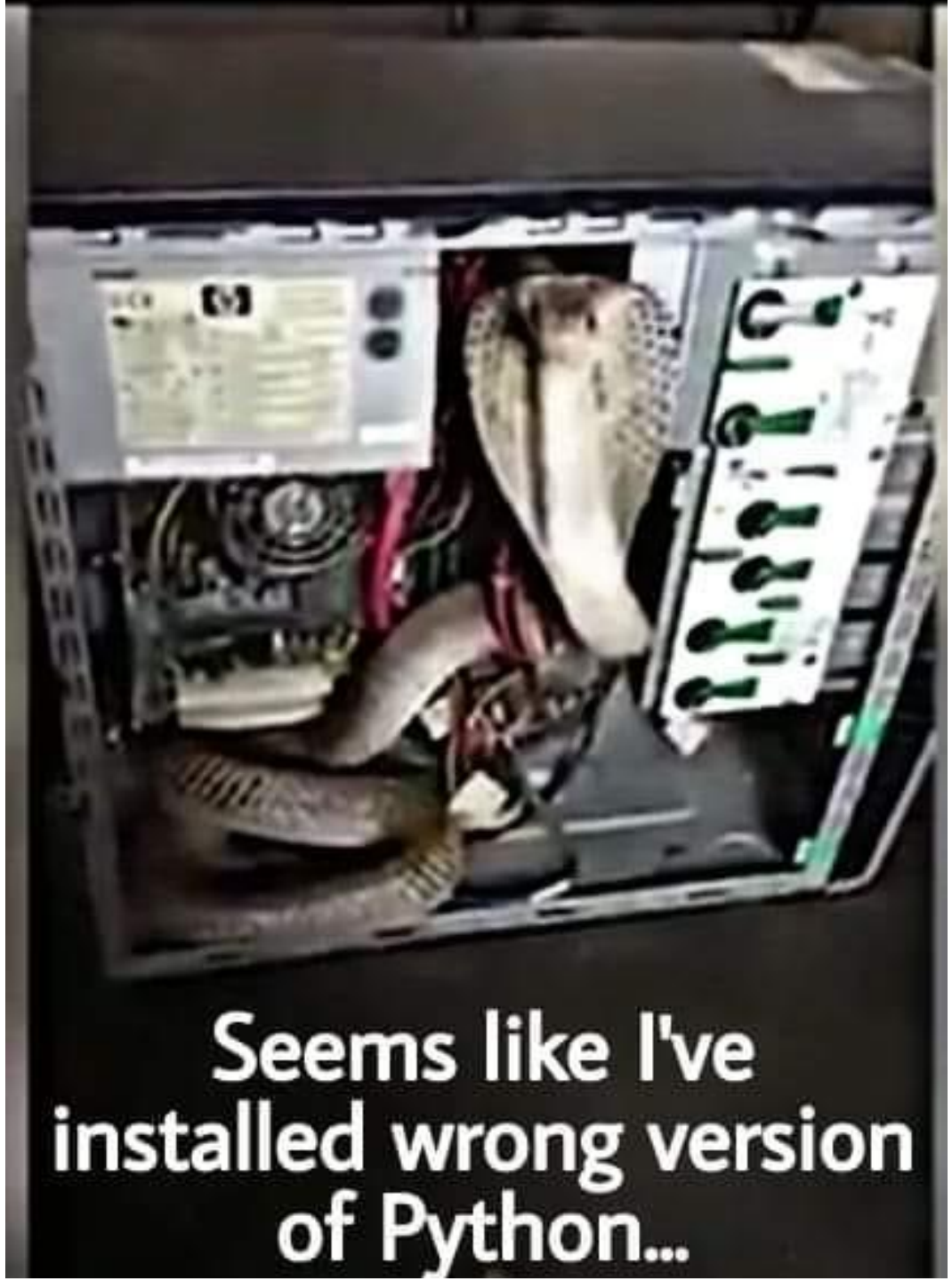


pip Installation

PIP Inštalácia Odinštalácia Balíkov

```
! pip install meno_balika
! pip install prettytable
! pip uninstall prettytable
! pip list
```

```
py -m pip install SomePackage              # posledná verzia
py -m pip install SomePackage==1.0.4       # špecifická verzia
py -m pip install 'SomePackage>=1.0.4'     # minimálna verzia
py -m pip install -r requirements.txt
py -m pip freeze > requirements.txt
py -m pip install -r requirements.txt
py -m pip uninstall SomePackage
```



sys Variables

argv	Command line args
builtin_module_names	Linked C modules
byteorder	Native byte order
check_interval	Signal check frequency
exec_prefix	Root directory
executable	Name of executable
exitfunc	Exit function name
modules	Loaded modules
path	Search path
platform	Current platform
stdin, stdout, stderr	File objects for I/O
version_info	Python version info
winver	Version number

sys.argv for *$ python foo.py bar -c qux --h*

sys.argv[0]	foo.py
sys.argv[1]	bar
sys.argv[2]	-c
sys.argv[3]	qux
sys.argv[4]	--h

os Variables

altsep	Alternative sep
curdir	Current dir string
defpath	Default search path
devnull	Path of null device
extsep	Extension separator
linesep	Line separator
name	Name of OS
pardir	Parent dir string
pathsep	Patch separator
sep	Path separator

Note Registered OS names: "posix", "nt", "mac", "os2", "ce", "java", "riscos"

Class Special Methods

__new__(cls)	__lt__(self, other)
__init__(self, args)	__le__(self, other)
__del__(self)	__gt__(self, other)
__repr__(self)	__ge__(self, other)
__str__(self)	__eq__(self, other)
__cmp__(self, other)	__ne__(self, other)
__index__(self)	__nonzero__(self)
__hash__(self)	
__getattr__(self, name)	
__getattribute__(self, name)	
__setattr__(self, name, attr)	
__delattr__(self, name)	
__call__(self, args, kwargs)	

String Methods

capitalize() *	lstrip()
center(width)	partition(sep)
count(sub, start, end)	replace(old, new)
decode()	rfind(sub, start ,end)
encode()	rindex(sub, start, end)
endswith(sub)	rjust(width)
expandtabs()	rpartition(sep)
find(sub, start, end)	rsplit(sep)
index(sub, start, end)	rstrip()
isalnum() *	split(sep)
isalpha() *	splitlines()
isdigit() *	startswith(sub)
islower() *	strip()
isspace() *	swapcase() *
istitle() *	title() *
isupper() *	translate(table)
join()	upper() *
ljust(width)	zfill(width)
lower() *	

Note Methods marked * are locale dependant for 8-bit strings.

List Methods

append(item)	pop(position)
count(item)	remove(item)
extend(list)	reverse()
index(item)	sort()
insert(position, item)	

File Methods

close()	readlines(size)
flush()	seek(offset)
fileno()	tell()
isatty()	truncate(size)
next()	write(string)
read(size)	writelines(list)
readline(size)	

Indexes and Slices (of *a=[0,1,2,3,4,5]*)

len(a)	6
a[0]	0
a[5]	5
a[-1]	5
a[-2]	4
a[1:]	[1,2,3,4,5]
a[:5]	[0,1,2,3,4]
a[:-2]	[0,1,2,3]
a[1:3]	[1,2]
a[1:-1]	[1,2,3,4]
b=a[:]	Shallow copy of a

Datetime Methods

today()	fromordinal(ordinal)
now(timezoneinfo)	combine(date, time)
utcnow()	strptime(date, format)
fromtimestamp(timestamp)	
utcfromtimestamp(timestamp)	

Time Methods

replace()	utcoffset()
isoformat()	dst()
__str__()	tzname()
strftime(format)	

Date Formatting (strftime and strptime)

%a	Abbreviated weekday (Sun)
%A	Weekday (Sunday)
%b	Abbreviated month name (Jan)
%B	Month name (January)
%c	Date and time
%d	Day (leading zeros) (01 to 31)
%H	24 hour (leading zeros) (00 to 23)
%I	12 hour (leading zeros) (01 to 12)
%j	Day of year (001 to 366)
%m	Month (01 to 12)
%M	Minute (00 to 59)
%p	AM or PM
%S	Second (00 to 61 [4])
%U	Week number [1] (00 to 53)
%w	Weekday [2] (0 to 6)
%W	Week number [3] (00 to 53)
%x	Date
%X	Time
%y	Year without century (00 to 99)
%Y	Year (2008)
%Z	Time zone (GMT)
%%	A literal "%" character (%)

1. Sunday as start of week. All days in a new year preceding the first Sunday are considered to be in week 0.
2. 0 is Sunday, 6 is Saturday.
3. Monday as start of week. All days in a new year preceding the first Monday are considered to be in week 0.
4. This is not a mistake. Range takes account of leap and double-leap seconds.

itacademy

Find, install and publish Python packages with the Python Package Index

Search projects

Or browse projects

486,618 projects 4,952,800 releases 9,297,799 files 747,641 users

python™ Package Index

The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages.

Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

cowsay 6.1

```
pip install cowsay
```

Latest version

Released: Sep 25, 2023

The famous cowsay for GNU/Linux is now available for python

Project description

Introduction

A python API / Command-line tool for the famous linux `cowsay`.
Take a look at CHANGELOG.md for the changes.

Brief History

`cowsay` for GNU/Linux was initially written in perl by Tony Monroe. More info here.

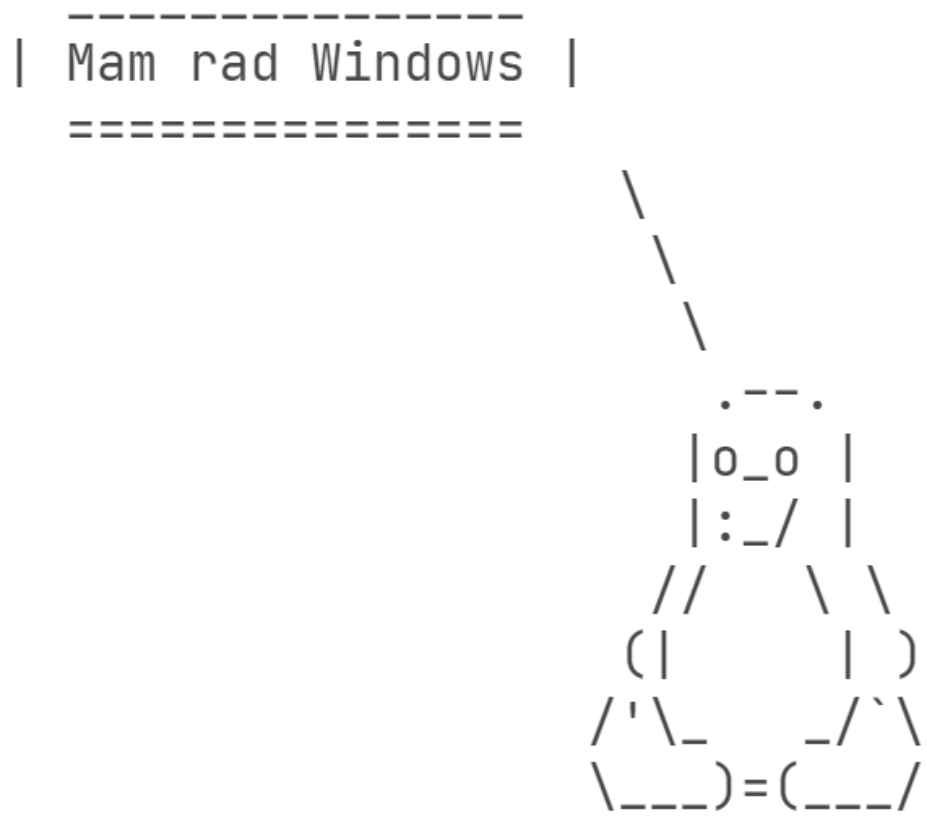
[1] 4.7s

```
1 # cowsay
2 ! pip install cowsay
```

```
Collecting cowsay
  Downloading cowsay-6.1-py3-none-any.whl (25 kB)
Installing collected packages: cowsay
Successfully installed cowsay-6.1

[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
```

Add code cell


Toto je Python Milka
^__^
(oo)_______
(__)\ )\/\
||----w |
|| ||


Mam rad Windows
.--.
|o_o |
|:_/ |

Modul Pretty Table

Modul PrettyTable

#!pip install prettytable

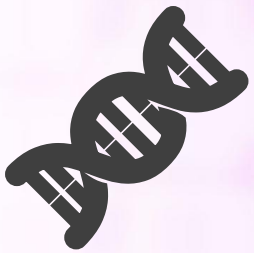
Pretty Table

```
0.2s
1 #!pip install prettytable
2
3 from prettytable import PrettyTable
4
5 tabulka_zakaznici = PrettyTable(["Meno", "Priezvisko", "Email",
  "Kontakt"])
6 print(tabulka_zakaznici)
7
8 tabulka_zakaznici.add_row(["Mario", "Sangala", "abc@nieco.sk", 123])
9 tabulka_zakaznici.add_row(["Ivana", "Sangala", "abc@nieco.sk", 456])
10 tabulka_zakaznici.add_row(["Monika", "Selicka", "abc@nieco.sk", 85])
11
12
13 print(tabulka_zakaznici)
```

```
+------+-----------+-------+---------+
| Meno | Priezvisko | Email | Kontakt |
+------+-----------+-------+---------+
+------+-----------+-------+---------+
+--------+-----------+--------------+---------+
|  Meno  | Priezvisko |    Email     | Kontakt |
+--------+-----------+--------------+---------+
| Mario  |  Sangala   | abc@nieco.sk |   123   |
| Ivana  |  Sangala   | abc@nieco.sk |   456   |
| Monika |  Selicka   | abc@nieco.sk |   852   |
+--------+-----------+--------------+---------+
```

Dedičnosť (Inheritance)

python

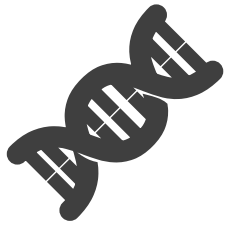


Čo je Dedičnosť (Biologická)

Dedičnosť

Dedičnosť alebo heredita je základný biologický proces, pri ktorom sa na rodičoch a ich potomstve vzniknutom pohlavnou cestou prejavuje podobnosť až zhodnosť v jednotlivých znakoch a vlastnostiach.

Dedičnosť v Pythone

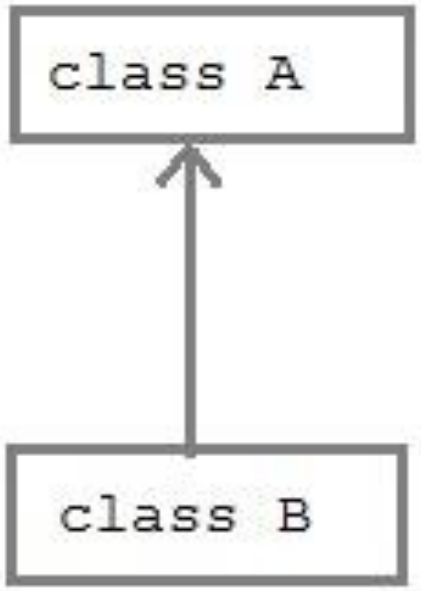


- Umožňuje **definovať triedu**, ktorá **dedí všetky metódy** a **vlastnosti** z **inej triedy**
- Tento koncept je **základným stavebným kameňom objektovo orientovaného programovania (OOP)**
- Poskytuje **spôsob**, **ako vytvoriť novú triedu pre použitie**, **modifikáciu** alebo **rozšírenie funkcionality existujúcich tried bez ich opätovného písania**

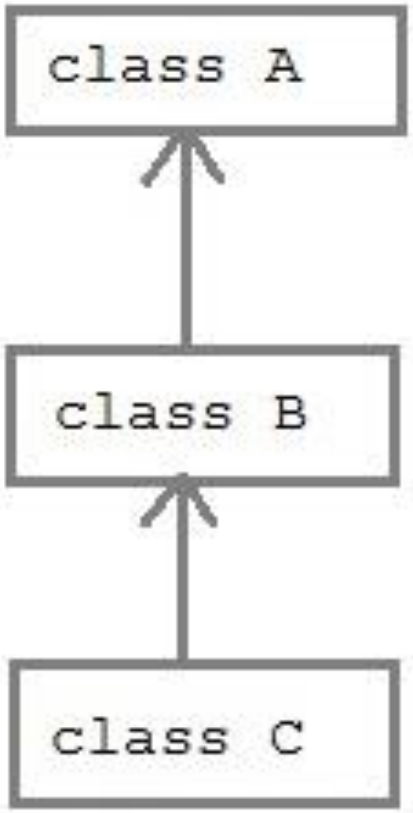
Dedičnosť

1. Trieda má/bude mať potomkov?
2. Potrebujeme odvodeniny?
3. Chcem rovnaké/podobné/odlišné správanie?
4. Akí sú predkovia? Aké majú vlastnosti a správanie?

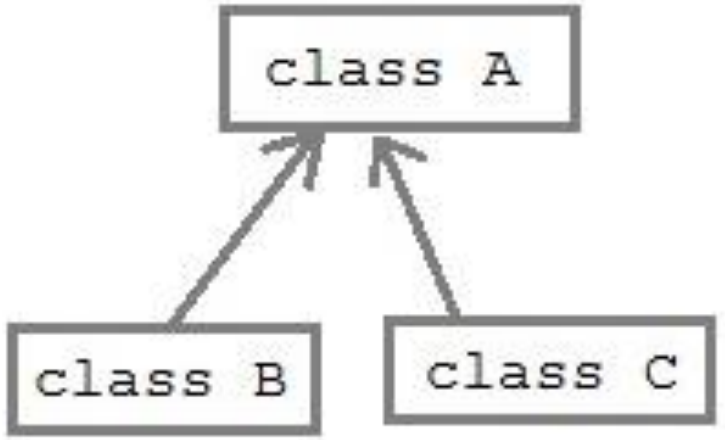
Typy Dedičností



Simple Inheritance

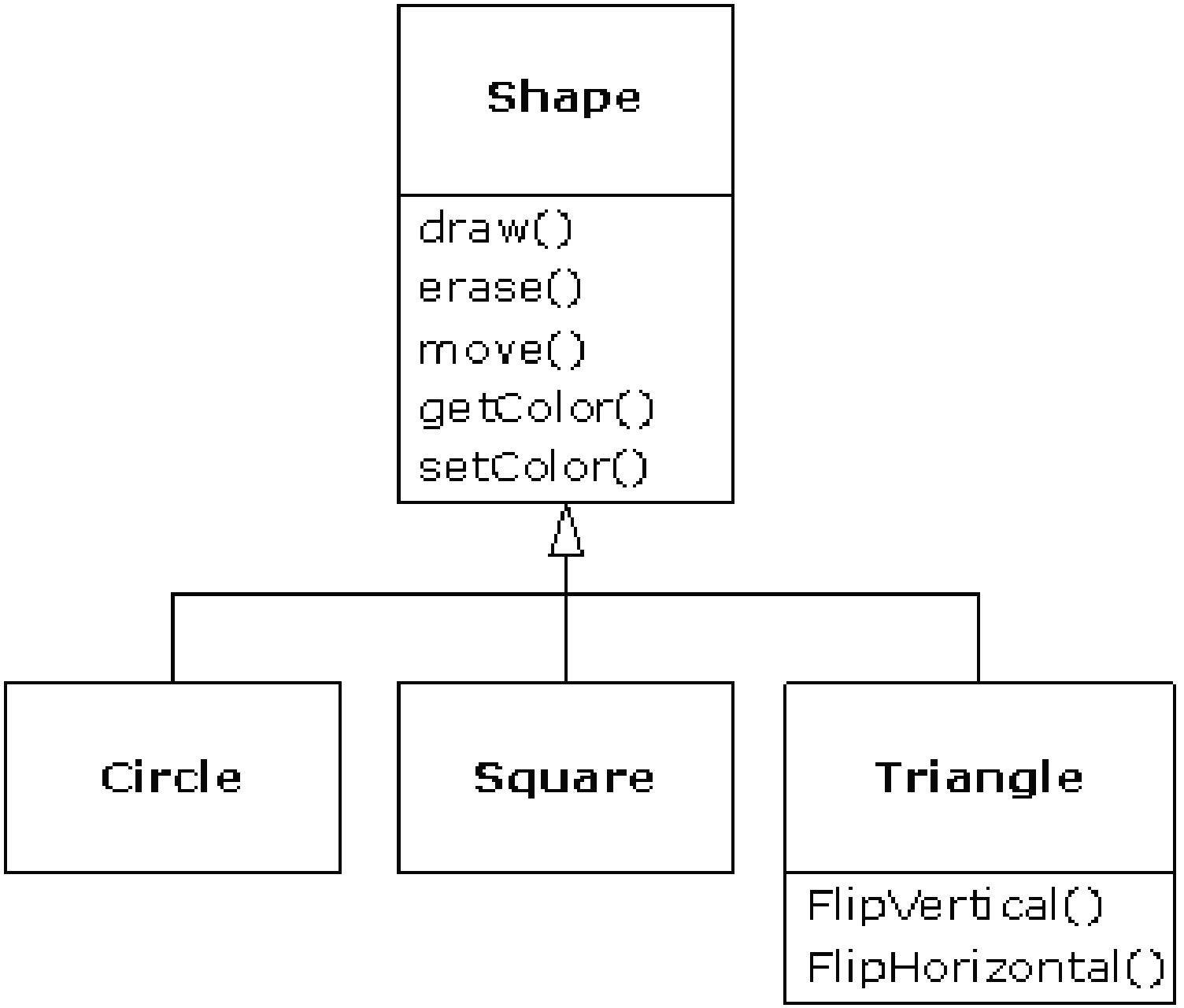


Multilevel inheritance



Heirarchical inheritance

Dedičnosť (Inheritance)



```python
class Vozidlo:
    def __init__(self, nazov, rychlost):
        self.nazov = nazov
        self.rychlost = rychlost
    def pohyb(self):
        print(f"{self.nazov} sa pohybuje rýchlosťou {self.rychlost} km/h.")

# Odvodená trieda Auto, ktorá dedí z Vozidlo
class Auto(Vozidlo):
    def __init__(self, nazov, rychlost, pocet_dveri):
        super().__init__(nazov, rychlost)
        self.pocet_dveri = pocet_dveri
    def pohyb(self):
        print(f"{self.nazov}, auto s {self.pocet_dveri} dverami, sa pohybuje rýchlosťou {self.rychlost} km/h.")

# Odvodená trieda Motocykel, ktorá dedí z Vozidlo
class Motocykel(Vozidlo):
    def pohyb(self):
        print(f"{self.nazov} sa rýchlo pohybuje rýchlosťou {self.rychlost} km/h na dvoch kolesách.")

auto = Auto("Skoda Octavia", 120, 4)
motocykel = Motocykel("Harley Davidson", 150)
auto.pohyb()
motocykel.pohyb()
```

isinstance a issubclass

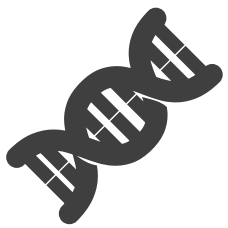
Boolean notácia:

- **is**
- **has**
- **can**

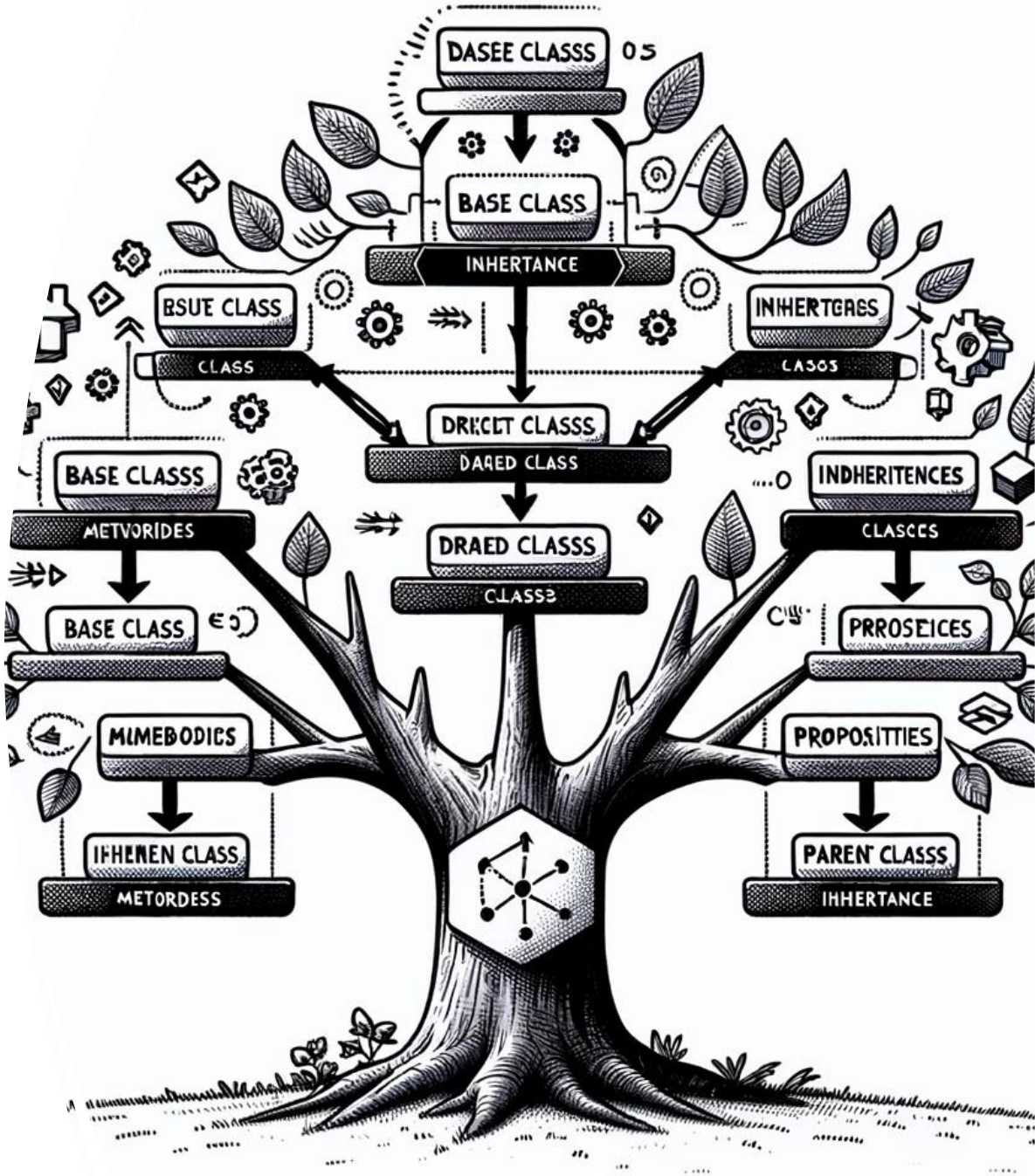
- Je Laco Objektom triedy str?
- Je trieda sama sebe podtriedou?

```
>>> isinstance("Laco",str)
True
>>> isinstance("Laco",int)
False
>>> isinstance(5,int)
True
>>> isinstance(5L,int)
False
>>> issubclass(int,object)
True
>>> issubclass(Exception,object)
True
>>> issubclass(ArithmeticError,Exception)
True
>>> issubclass(SyntaxError,Exception)
True
```

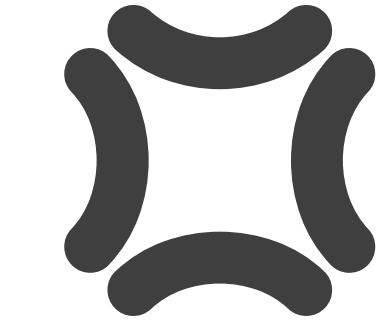
Úlohy Dedičnosť



1. Pochopenie dedičnosti
2. Používanie dedičnosti
3. Osvojenie si výhod dedičnosti



Abstrakcia



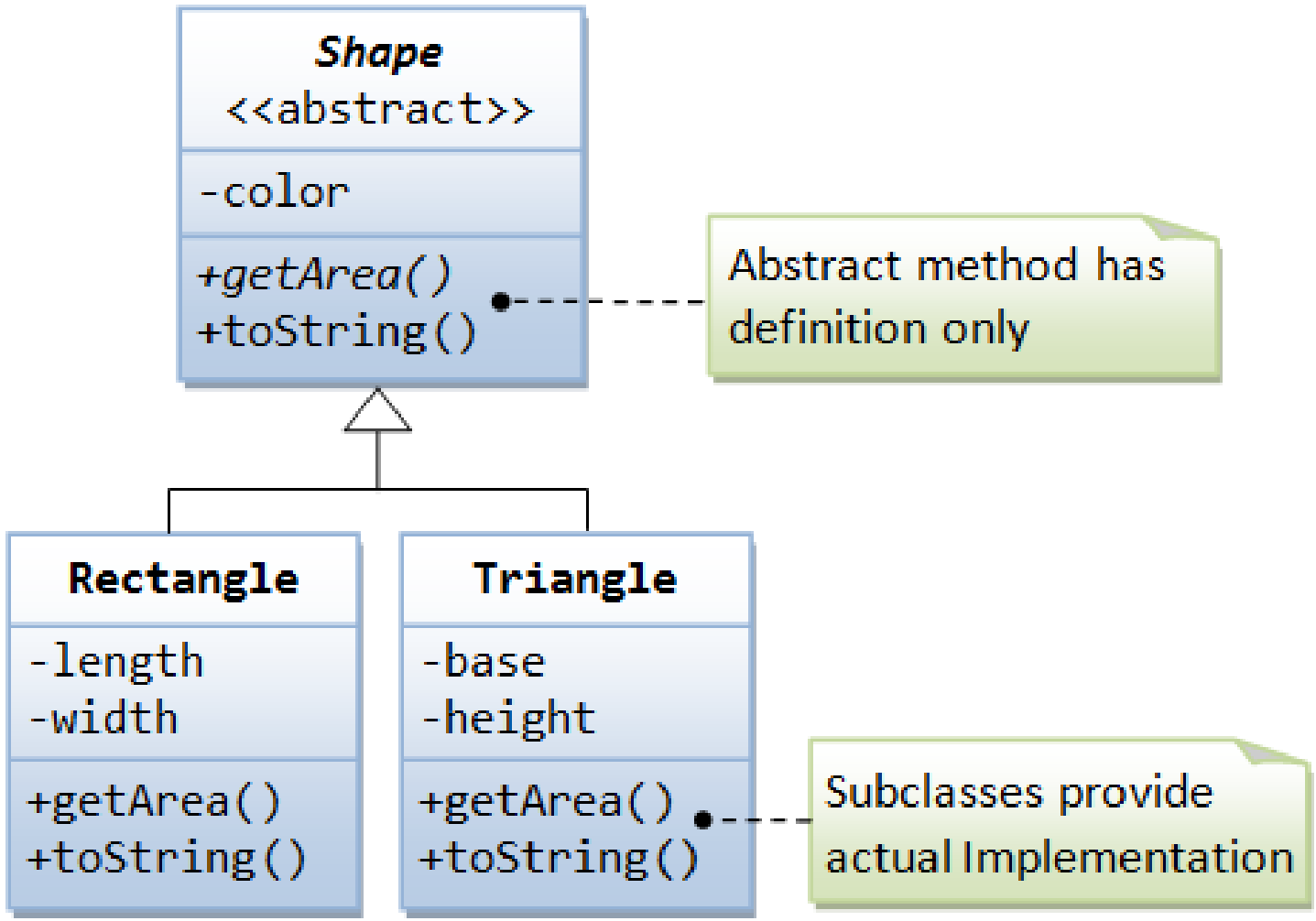
python

Abstrakcia Tried v Pythone

- Je **pokročilý programovací koncept**, ktorý sa **používa na definovanie tried**, ktoré **nemôžu byť priamo inštancované**, ale **slúžia** ako **základ pre ďalšie triedy**
- **Umožňuje definovať metódy** a **vlastnosti**, ktoré **musia byť implementované** v **odvodených triedach**
- Tým **sa zabezpečuje dodržiavanie určitého rozhrania** alebo **kontraktu**

Abstrakcia (Abstraktné Triedy)

- Pod abstrakciou sa vo všeobecnosti chápe zameranie sa na **kľúčové vlastnosti** nejakého prvku reálneho sveta (alebo aj nereálneho)
- Napríklad tvary, ktoré chceme vykresľovať



Previous topic
contextlib — Utilities for **with**-statement contexts

Next topic
atexit — Exit handlers

This Page
Report a Bug
Show Source

abc — Abstract Base Classes

Source code: Lib/abc.py

This module provides the infrastructure for defining abstract base classes (ABCs) in Python, as outlined in **PEP 3119**; see the PEP for why this was added to Python. (See also **PEP 3141** and the `numbers` module regarding a type hierarchy for numbers based on ABCs.)

The `collections` module has some concrete classes that derive from ABCs; these can, of course, be further derived. In addition, the `collections.abc` submodule has some ABCs that can be used to test whether a class or instance provides a particular interface, for example, if it is hashable or if it is a mapping.

This module provides the metaclass `ABCMeta` for defining ABCs and a helper class `ABC` to alternatively define ABCs through inheritance:

class `abc.`**ABC**

A helper class that has `ABCMeta` as its metaclass. With this class, an abstract base class can be created by simply deriving from `ABC` avoiding sometimes confusing metaclass usage, for example:

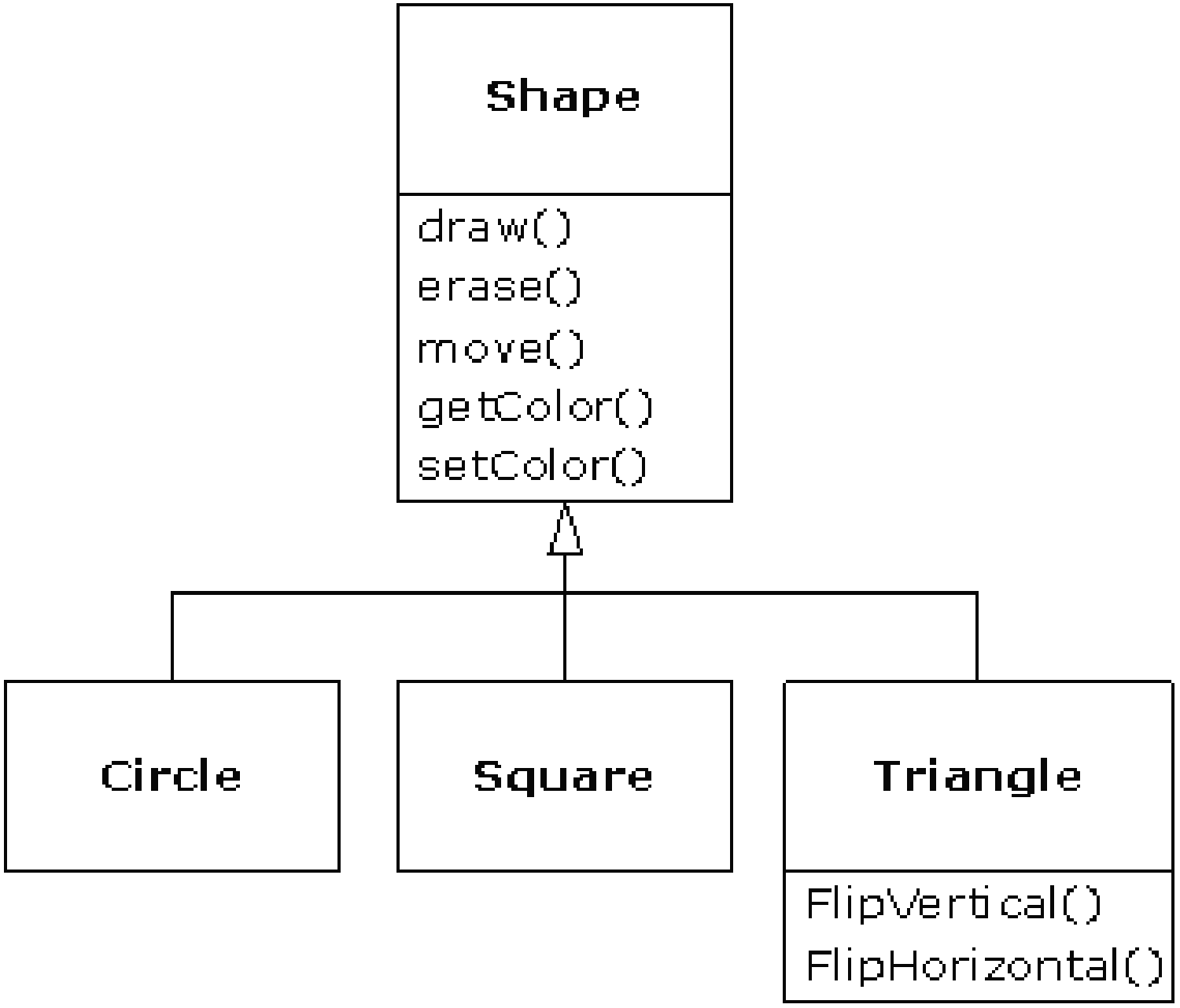
```
from abc import ABC

class MyABC(ABC):
    pass
```

Note that the type of `ABC` is still `ABCMeta`, therefore inheriting from `ABC` requires the usual precautions regarding metaclass usage, as multiple inheritance may lead to metaclass conflicts. One may also define an

```python
from abc import ABC, abstractmethod

class Vozidlo(ABC):
    @abstractmethod
    def pohyb(self):
        pass

class Auto(Vozidlo):
    def pohyb(self):
        print("Auto sa pohybuje po ceste.")

class Lod(Vozidlo):
    def pohyb(self):
        print("Loď sa plaví po vode.")

# Vytvorenie inštancií a volanie ich metód
auto = Auto()
auto.pohyb()  # Vypíše: Auto sa pohybuje po ceste.

lod = Lod()
lod.pohyb()  # Vypíše: Loď sa plaví po vode.
```

```python
from abc import ABC, abstractmethod
class AbstraktnaTrieda(ABC):
    @abstractmethod
    def abstraktna_metoda(self):
        pass

# Táto trieda nemôže byť inštancovaná, pretože obsahuje abstraktnú metódu
# abstraktna_instancia = AbstraktnaTrieda() # Toto by spôsobilo chybu
class KonkretnaTrieda(AbstraktnaTrieda):
    def abstraktna_metoda(self):
        print("Implementácia abstraktnej metódy v konkrétnej triede")

# Teraz môžeme inštancovať KonkretnaTrieda, pretože implementuje všetky abstraktné metódy
konkretna_instancia = KonkretnaTrieda()
konkretna_instancia.abstraktna_metoda() # Vypíše: Implementácia abstraktnej metódy v konkrétnej triede
```

Dedičnosť a Abstrakcia



Abstraktné Triedy

Use the `abc` module to create abstract classes. Use the `abstractmethod` decorator to declare a method abstract, and declare a class abstract using one of three ways, depending upon your Python version.

In Python 3.4 and above, you can inherit from `ABC`. In earlier versions of Python, you need to specify your class's metaclass as `ABCMeta`. Specifying the metaclass has different syntax in Python 3 and Python 2. The three possibilities are shown below:

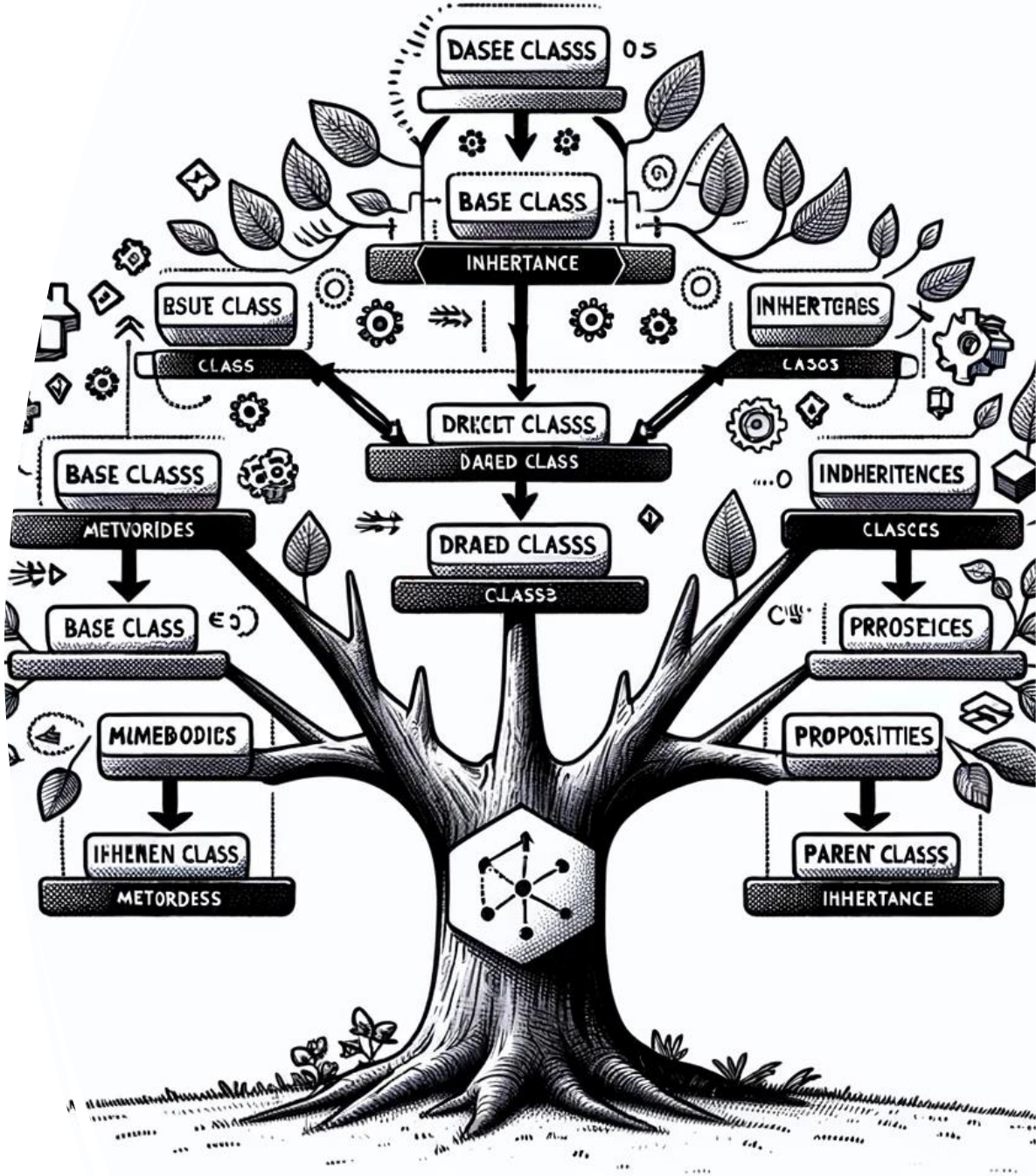
```
# Python 3.4+
from abc import ABC, abstractmethod
class Abstract(ABC):
    @abstractmethod
    def foo(self):
        pass
```

```
# Python 3.0+
from abc import ABCMeta, abstractmethod
class Abstract(metaclass=ABCMeta):
    @abstractmethod
    def foo(self):
        pass
```

```
# Python 2
from abc import ABCMeta, abstractmethod
class Abstract:
    __metaclass__ = ABCMeta

    @abstractmethod
    def foo(self):
        pass
```

Úlohy Abstrakcia

1. Pochopenie abstrakcie
2. Používanie abstrakcie
3. Osvojenie si výhod abstrakcie



Zapúzdrenie (Enkapsulácia)

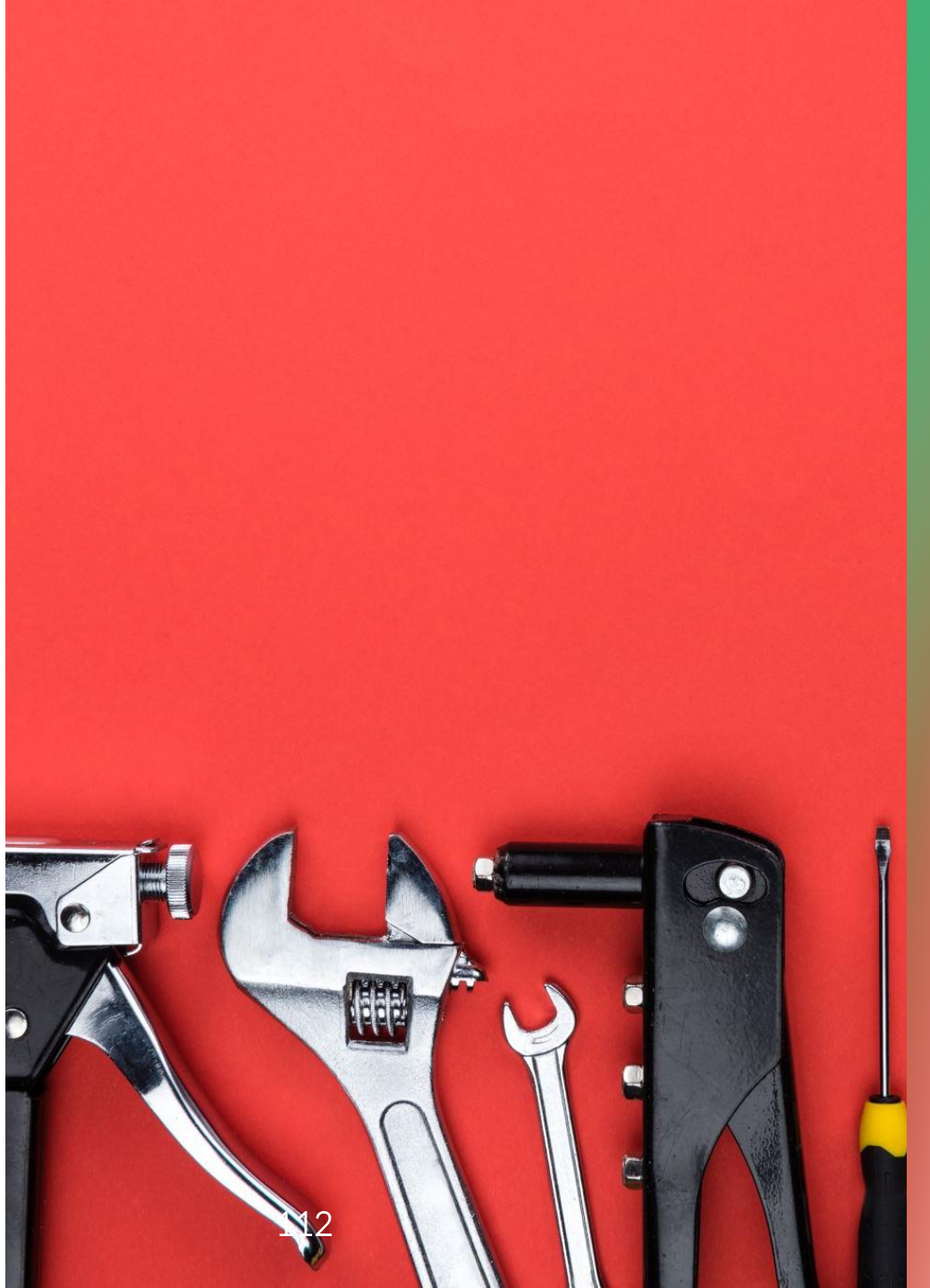
python

Zapúzdrenie

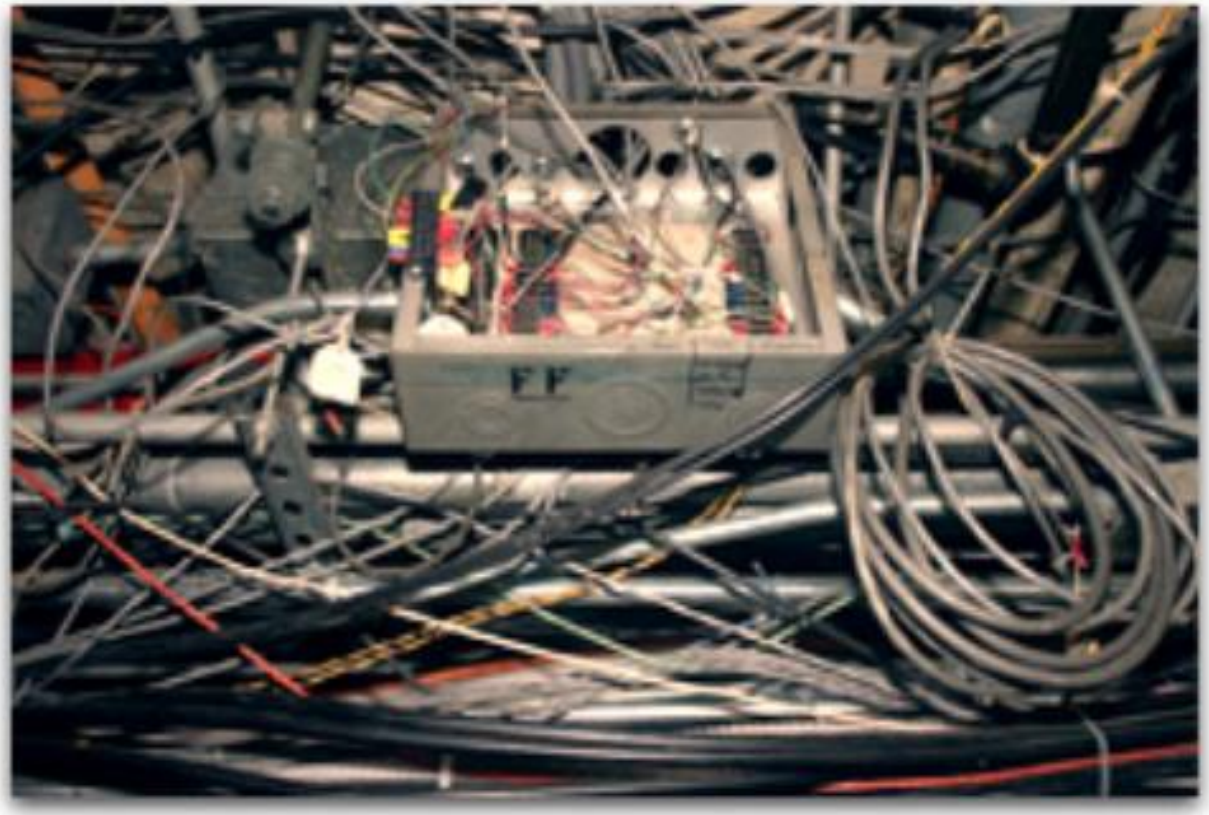
- Jeden z **kľúčových konceptov (OOP)**
- **Umožňuje skryť vnútorné detaily implementácie triedy** a **vystaviť** len tie **metódy** a **atribúty**, ktoré sú **potrebné pre jej použitie**
- **Interné stavy objektu sú chránené** pred **vonkajším prístupom**, a tak sa **zabraňuje neautorizovaným zmenám**
- V praxi sa zapúzdrenie dosahuje **pomocou privátnych a chránených atribútov a metód**

Zapúzdrenie

- **V Pythone** sa privátne atribúty zvyčajne označujú pomocou **2 podčiarkovníkov (__atribut) pred ich menom**
- **Python nemá striktný mechanizmus na vynútenie privátnosti**, ale táto **konvencia signalizuje ostatným programátorom**, že tieto **atribúty by sa nemali priamo pristupovať** alebo **meniť mimo definície triedy**



Zapúzdrenie (Enkapsulácia)

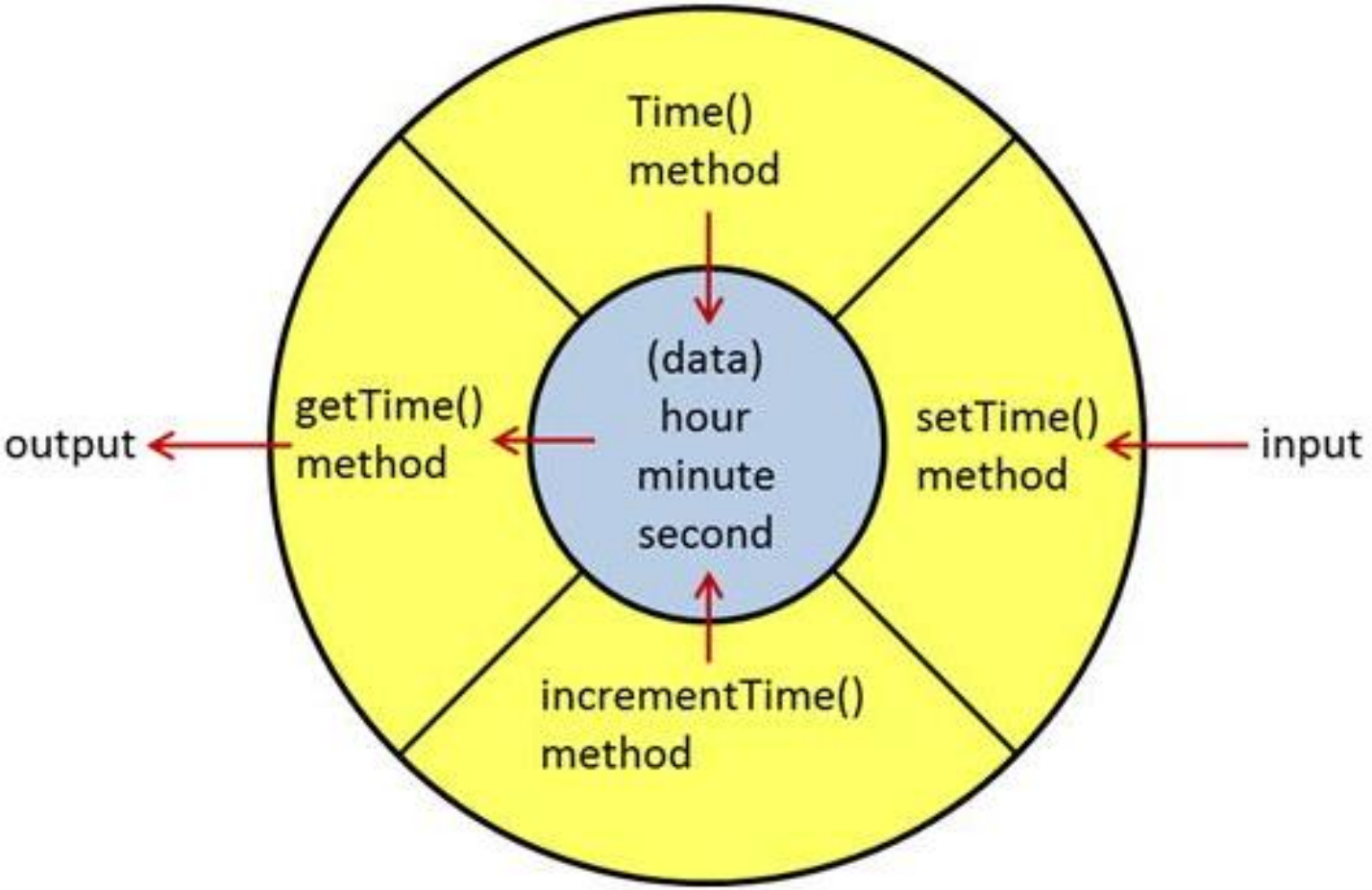


Jedno z týchto riešení je o niečo bezpečnejšie

```python
class ElektronickeZariadenie:
    def __init__(self, nazov, rok_vydania):
        self.nazov = nazov  # Verejný atribút
        self.__rok_vydania = rok_vydania  # Privátny atribút
        self.__zapnute = False  # Privátny atribút, indikuje, či je zariadenie zapnuté
    def zapnut(self):
        if not self.__zapnute:
            self.__zapnuté = True
            print(f"{self.nazov} je teraz zapnuté.")
        else:
            print(f"{self.nazov} už je zapnuté.")
    def vypnut(self):
        if self.__zapnute:
            self.__zapnute = False
            print(f"{self.nazov} je teraz vypnuté.")
        else:
            print(f"{self.nazov} už je vypnuté.")
    def ziskat_informacie(self):
        return f"{self.nazov}, Rok vydania: {self.__rok_vydania}"

# Použitie triedy
zariadenie = ElektronickeZariadenie("Smartphone XYZ", 2024)
zariadenie.zapnut()  # Vypíše: Smartphone XYZ je teraz zapnuté
print(zariadenie.získat_informácie())  # Vypíše: Smartphone XYZ, Rok vydania: 2024
zariadenie.vypnut()  # Vypíše: Smartphone XYZ je teraz vypnuté
```

```python
class Kniznica:
    def __init__(self):
        self.__knihy = {}  # Privátny slovník na uchovávanie kníh
    def pridat_knihu(self, názov, autor):
        if nazov in self.__knihy:
            print(f"Kniha '{nazov}' už existuje v knižnici.")
        else:
            self.__knihy[nazov] = autor
            print(f"Kniha '{nazov}' bola pridaná do knižnice.")
    def odstranit_knihu(self, názov):
        if nazov in self.__knihy:
            del self.__knihy[nazov]
            print(f"Kniha '{nazov}' bola odstránená z knižnice.")
        else:
            print(f"Kniha '{nazov}' sa nenašla v knižnici.")
    def vyhladat_knihu(self, nazov):
        if nazov in self.__knihy:
            autor = self.__knihy[nazov]
            print(f"Kniha '{nazov}' od autora '{autor}' je dostupná.")
        else:
            print(f"Kniha '{názov}' sa nenašla v knižnici.")

# Použitie triedy Knižnica
moja_kniznica = Kniznica()
moja_kniznica.pridat_knihu("Veľký Gatsby", "F. Scott Fitzgerald")
moja_kniznica.vyhladat_knihu("Veľký Gatsby")  # Vypíše: Kniha 'Veľký Gatsby' od autora 'F. Scott Fitzgerald' je dostupná.
moja_kniznica.odstranit_knihu("Veľký Gatsby")
```

Zapúzdrenie (Enkapuslácia)



Úlohy

Zapúzdrenie

1. Pochopenie zapúzdrenia (enkapsulácie)
2. Používanie zapúzdrenia
3. Osvojenie si výhod zapúzdrenia

Zabudované (Vstavané) Atribúty Tried

Zabudované (Vstavané) Atribúty Triedy

1. **__dict__**: Slovník obsahujúci menný priestor triedy.
2. **__doc__**: Dokumentačný reťazec triedy alebo žiadny, ak nie je definovaný
3. **__name__**: Názov triedy
4. **__module__**: Názov modulu, v ktorom je trieda definovaná. Tento atribút je "__main__" v interaktívnom režime
5. **__bases__**: Možná prázdna n-tica obsahujúca základné triedy v poradí ich výskytu v zozname základných tried

Zabudované Funkcie (built-in functions)

		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	

Atribút __dict__

```
0.1s
```

```python
class Kurz:
    '''Vitajte na kurze Pythone'''

    def __init__(self):
        print("Metóda __init__ konštruktor")

# Tlač hodnoty atribútu Built-in __dict__ pre triedu Kurz
print(Kurz.__dict__)
```

```
{'__module__': '__main__', '__doc__': 'Vitajte na kurze Pythone', '__init__': <function Kurz.__init__ at 0x7f26df4eb430>,
```

Atribút __doc__

```
0.1s
```

```python
class Kurz:
    '''Vitajte na kurze Pythone'''

    def __init__(self):
        print("Metóda __init__ konštruktor")

# získanie dokumentácie pre použitie triedy Kurz
# Vstavaný atribút triedy __doc__
print(Kurz.__doc__)
```

```
Vitajte na kurze Pythone
```

Atribút __name__

```
0.1s
1 class Kurz:
2     '''Vitajte na kurze Pythone'''
3 
4     def __init__(self):
5         print("Metóda __init__ konštruktor")
6 
7 # Získanie názvu triedy pomocou vstavaného atribútu triedy __name__
8 print(Kurz.__name__)
```

```
Kurz
```

Atribút __module__

```
0.1s
1 class Kurz:
2     '''Vitajte na kurze Pythone'''
3
4     def __init__(self):
5         print("Metóda __init__ konštruktor")
6
7 # získanie modulu triedy pomocou vstavaného atribútu triedy
  __module__
8 print(Kurz.__module__)
```

```
__main__
```

Atribút __bases__

```
0.1s
```

```python
class Kurz:
    '''Vitajte na kurze Pythone'''

    def __init__(self):
        print("Metóda __init__ konštruktor")

# získanie bázových tried (nadtriedy) pomocou vstavaného atribútu triedy __bases__
print(Kurz.__bases__)
```

```
(<class 'object'>,)
```

Polymorfizmus (Mnohotvárnosť)

python

DID YOU KNOW

WHAT IS POLYMORPHISM ?

"A boy starts LOVE with the word FRIENDSHIP but girl ends LOVE with the same word FRIENDSHIP. Word is the same but attitude is different.This beautiful concept of OOP is nothing but POLYMORPHISM..."

-Broken Heart Programmer

Polymorfizmus Operátorov

```
1. a = 5
2. b = 6
3. print(a*b)


4. c = "Adam"
5. d = 3
6. print(c * d)
```

Polymorfizmus Operátorov

```
1. a = 5
2. b = 6
3. print(a+b)


4. c = "Adam"
5. d = "Sangala"
6. print(c + " " + d)
```

Polymorfizmus Defs/Funckií/Metód

```
1.  # Jednoduchý polymorfizmus
2.  # Method Overloading (Preťaženie metod)
3.  nazov_kurz = "Python OOP"
4.  # Počet znakov
5.  print(len(nazov_kurz))
6.  # Počet poloziek
7.  print(len([1,2,3,4,9]))
8.  print(len({"mama":"eva", "otec":"jozef"}))


9.  def sum(a, b, c = 0):
10.     return a+b+c
11. print(sum(1, 2))
12. print(sum(1, 2, 5))
```

Polymorfizmus Tried a ich Metód

- Umožňuje **zmeniť nejakú metódu rodičovskej triedy** u potomka
- Zmenenú metódu jednoducho u potomka **znova definujeme podľa potrieb potomka**
- Ak Python vykonáva nejakú metódu napr. speak(), tak sa toto vykonávanie prispôsobí (adaptuje) tej inštancii, ktorá túto metódu zavolala
  1. Pes → Woof
  2. Kačka → Quack
  3. Mačka → Meow

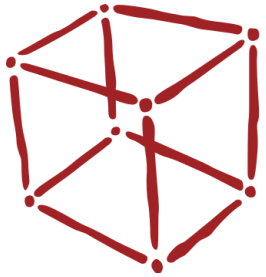
Ylvis - The Fox (What Does The Fox Say?) [Official music video HD]

Polymorfizmus (Dynamický)


```
1. Class RodicovskaTrieda:
2.    atributy...
3.    def metoda1(self):
4.          ...
5.    def metoda2(self):
6.          ...
```

```
1. Class
   DetskaTrieda(RodicovskaTrieda):
2.    atributy...
3. # Metoda sa predefinuje podľa
   potrieb potomka->polymorfizmus
4.    def metoda1(self):
5.          ...
6. # Nové metódy a atribúty
7. atributy
8. metody
```

Aké IDE mám použiť?



NetBeans

wxPython

Chceme úplne všetko!

ANACONDA®

Collaboration

Flexibility

Code
Completion

```
print("Sorry, we are down for maintenance")
print("We'll be back shortly")
```

Individual Edition is now

ANACONDA DISTRIBUTION

The world's most popular open-source Python distribution platform

Anaconda Distribution

Download

For Windows

Python 3.9 • 64-Bit Graphical Installer • 510 MB

Get Additional Installers

Open Source

Access the open-source software you need for projects in any field, from data visualization to robotics.

User-friendly

With our intuitive platform, you can easily search and install packages and create, load, and switch between environments.

Trusted

Our securely hosted packages and artifacts are methodically tested and regularly updated.

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Learning

Community

Documentation

Developer Blog

Applications on base (root) Channels Refresh

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch

JupyterLab
1.2.6
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated
Launch

Qt Console
↗ 4.6.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

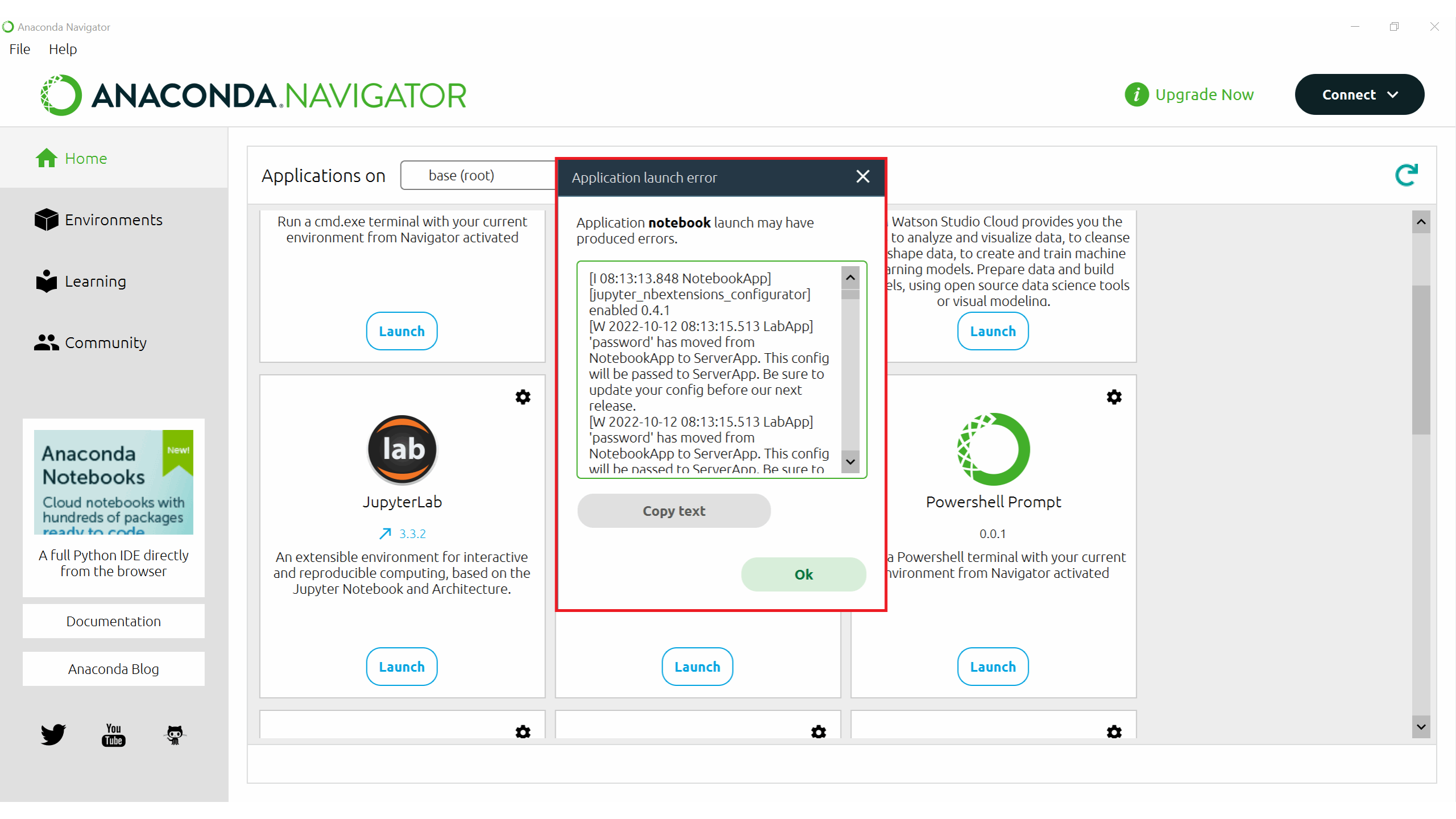
Spyder
↗ 4.0.1
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

VS Code
1.52.1
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install

Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install


Anaconda Navigator
File
Help
ANACONDA.NAVIGATOR
Upgrade Now
Connect
Home
Environments
Learning
Community
Anaconda Notebooks
New!
Cloud notebooks with hundreds of packages ready to code
A full Python IDE directly from the browser
Documentation
Anaconda Blog
Applications on
base (root)
Run a cmd.exe terminal with your current environment from Navigator activated
Launch
Application launch error
Application notebook launch may have produced errors.
[I 08:13:13.848 NotebookApp] [jupyter_nbextensions_configurator] enabled 0.4.1
[W 2022-10-12 08:13:15.513 LabApp] 'password' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your config before our next release.
[W 2022-10-12 08:13:15.513 LabApp] 'password' has moved from NotebookApp to ServerApp. This config
Copy text
Ok
Watson Studio Cloud provides you the
to analyze and visualize data, to cleanse
shape data, to create and train machine
arning models. Prepare data and build
els, using open source data science tools
or visual modelina.
Launch
lab
JupyterLab
3.3.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Powershell Prompt
0.0.1
a Powershell terminal with your current
nvironment from Navigator activated
Launch

Learn Python, JavaScript, C, C++, and Java

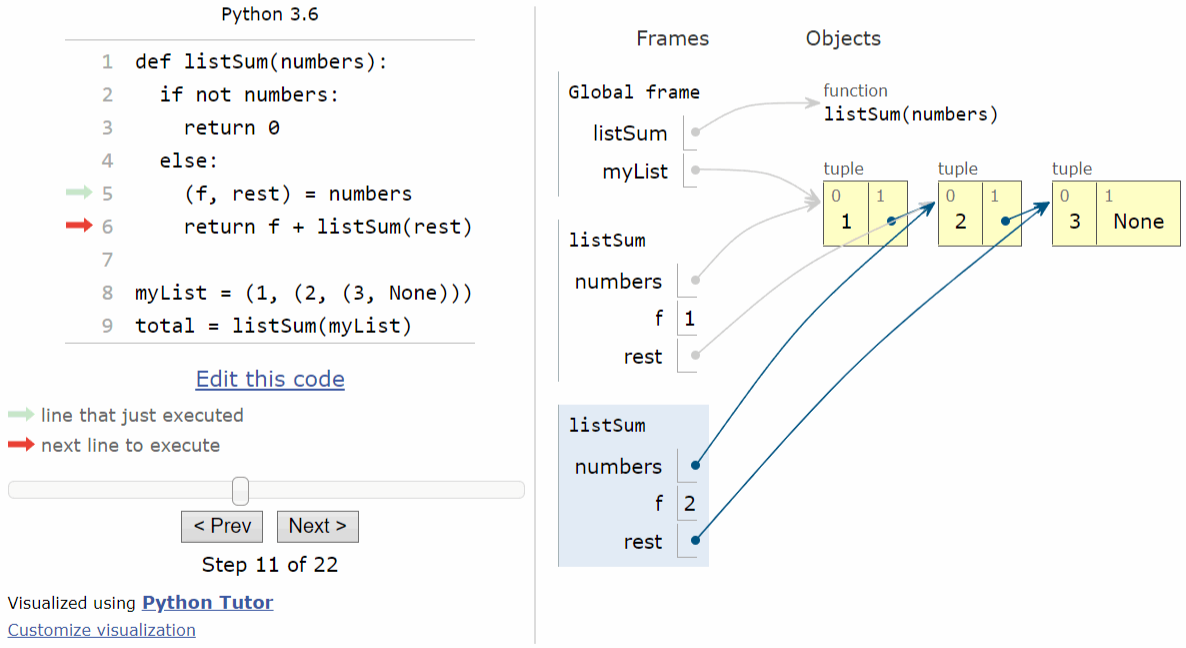
This tool helps you learn Python, JavaScript, C, C++, and Java programming by visualizing code execution. You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start writing and visualizing code now

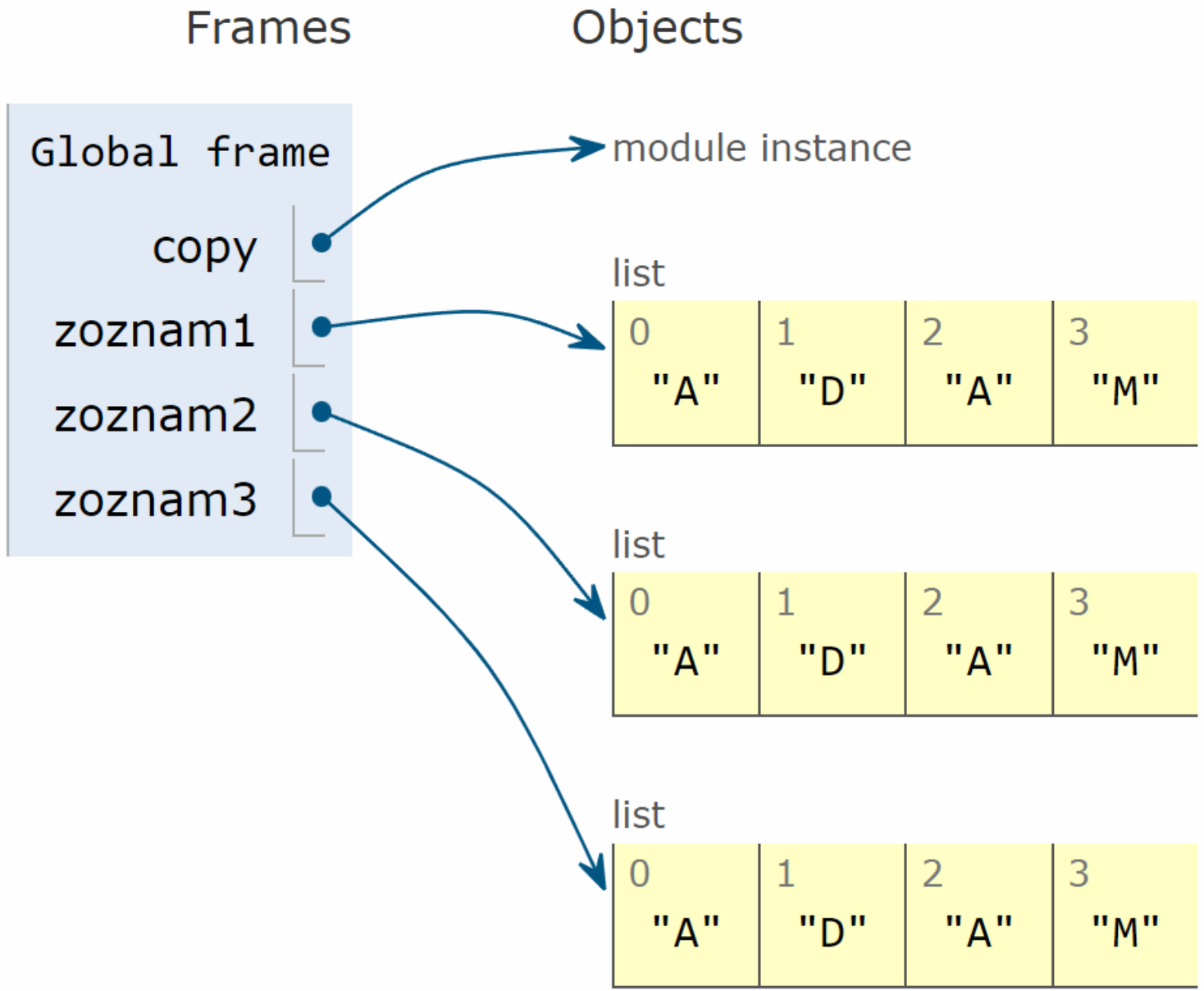
Related services: **JavaScript Tutor**, **C Tutor**, **C++ Tutor**, **Java Tutor**

Over ten million people in more than 180 countries have used Python Tutor to visualize over 100 million pieces of code. It's the most widely-used program visualization tool for computing education.

You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:



Modul copy


Frames
Objects
Global frame
copy
zoznam1
zoznam2
zoznam3
module instance
list
0 "A"
1 "D"
2 "A"
3 "M"
list
0 "A"
1 "D"
2 "A"
3 "M"
list
0 "A"
1 "D"
2 "A"
3 "M"

Microsoft Store

python

Aplikácie

Hranie

Knižnica

Pomocník

“python”

Filtre

Všetky oddelenia | Aplikácie | Hry

Python 3.9
Aplikácie • Developer tools
★★★★★ 4
Nainštalované

Python 3.8
Aplikácie • Developer tools
Bezplatné

Python 3.10
Aplikácie • Developer tools
Bezplatné

Python 3.7
Aplikácie • Developer tools
★★★★★ 1
Bezplatné

Learn Django and Python by GoLearningBus
Aplikácie • Books & reference
Bezplatné

WiBit.Net :: Programming in Python
Aplikácie • Education
0,99 €

计算机二级 Python 考试题库
Aplikácie • Education
Bezplatné

Data Science with Python
Aplikácie • Education

Python Programs
Aplikácie • Education

Learn Python
Aplikácie • Books & reference

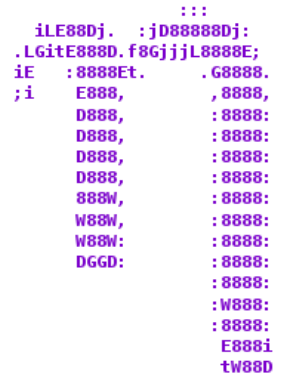
Introduction to Python Programming by...
Aplikácie • Books & reference

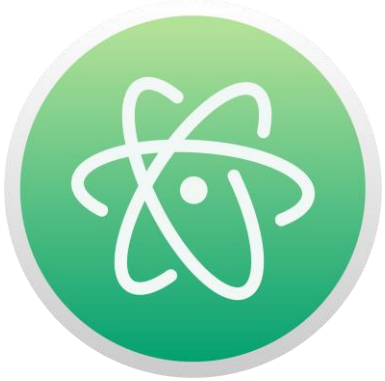
Python Playground
Aplikácie • Developer tools

Python Programming Manual For Absolute...
Aplikácie • Utilities & tools

Python User Tutorial
Aplikácie • Utilities & tools

Aký Editor mám použiť?





IDE ≠ editor

Dashboard

Welcome, itacademysk

CPU Usage: 0% used – 0.00s of 100s. Resets in 1 hour, 56 minutes More Info

File storage: 0% full – 100.0 KB of your 512.0 MB quota More Info

Upgrade Account

Recent
Consoles

You have no recent consoles.

New console:

$ Bash >>> Python More...

Version

2.7 3.7 3.8

Recent
Files

/home/itacademysk/.bashrc

/home/itacademysk/.gitconfig

/home/itacademysk/.profile

/home/itacademysk/.pythonstartup.py

/home/itacademysk/.vimrc

+ Open another file Browse files

Recent
Notebooks

Your account does not support Jupyter Notebooks. Upgrade your account to get access!

All
Web apps

You don't have any web apps.

Open Web tab

MySQL

Postgres

Initialize MySQL

Let's get started! The first thing to do is to initialize a MySQL server:

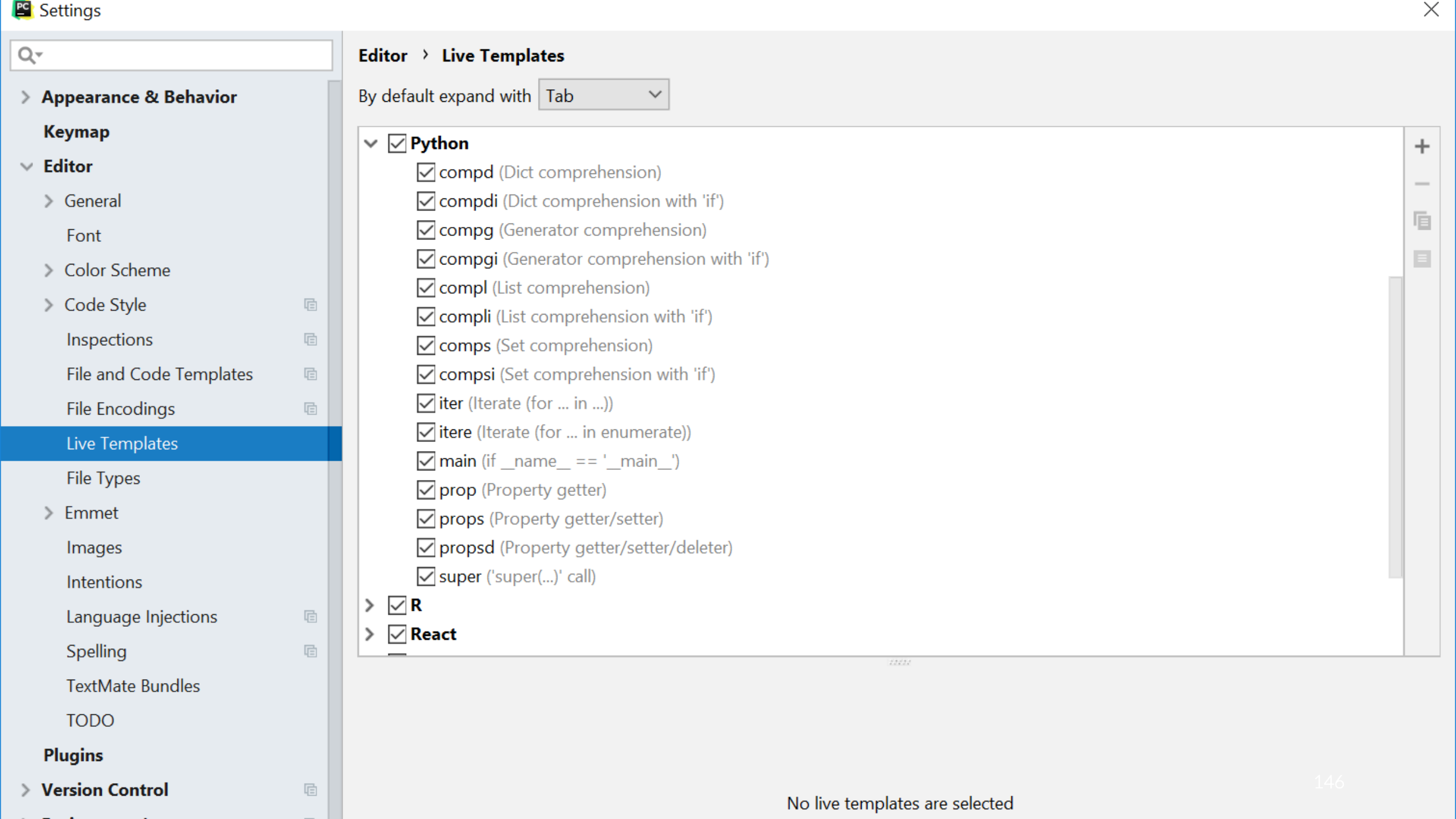
Enter a new password in the form below, and note it down: you'll need it to access the databases once you've created them. You will only need to do this once.

New password:

Confirm password:

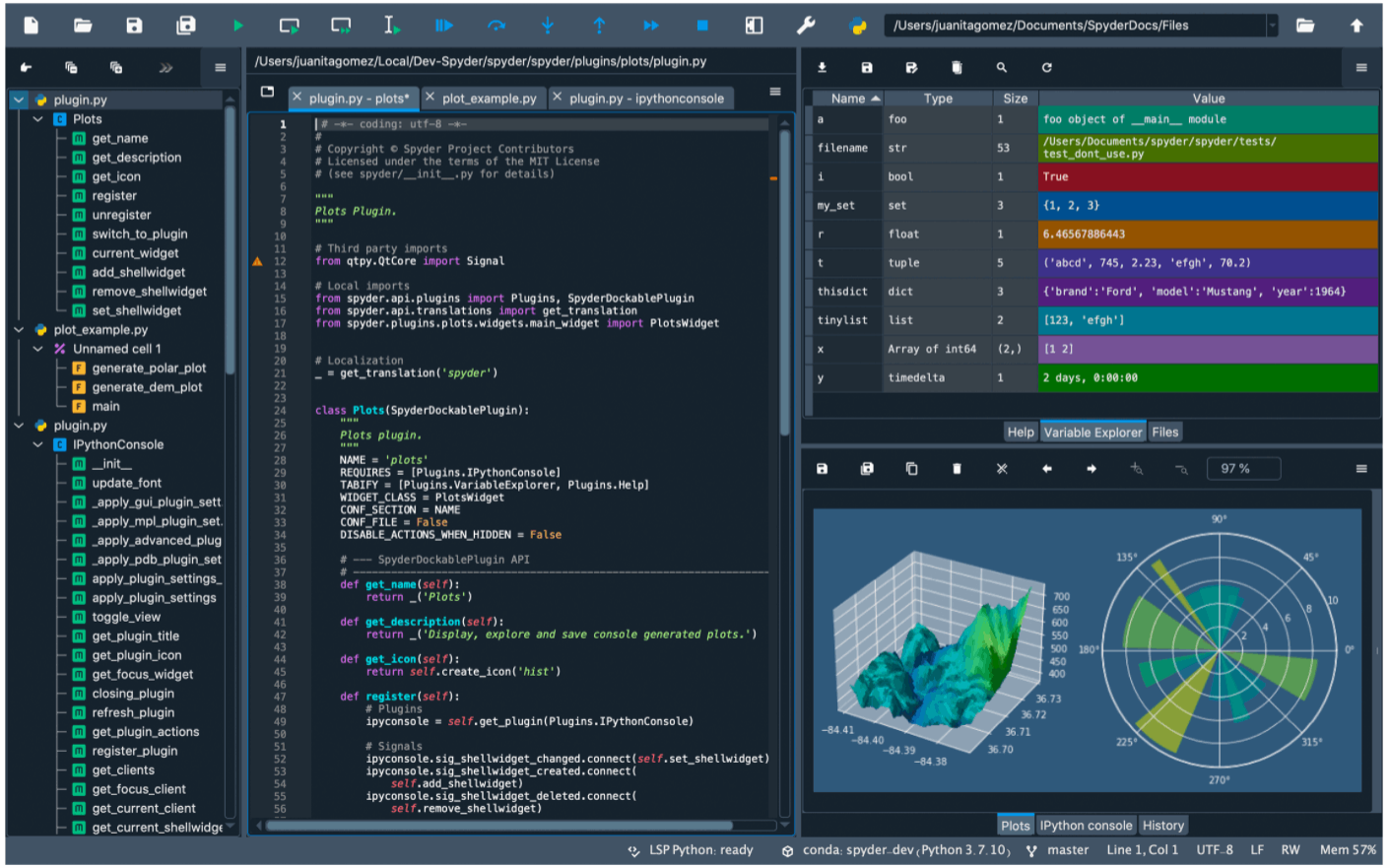
Initialize MySQL

This should be different to your main PythonAnywhere password, because it is likely to appear in plain text in any web applications you write.


Settings
Appearance & Behavior
Keymap
Editor
General
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Emmet
Images
Intentions
Language Injections
Spelling
TextMate Bundles
TODO
Plugins
Version Control
Editor › Live Templates
By default expand with Tab
Python
compd (Dict comprehension)
compdi (Dict comprehension with 'if')
compg (Generator comprehension)
compgi (Generator comprehension with 'if')
compl (List comprehension)
compli (List comprehension with 'if')
comps (Set comprehension)
compsi (Set comprehension with 'if')
iter (Iterate (for ... in ...))
itere (Iterate (for ... in enumerate))
main (if __name__ == '__main__')
prop (Property getter)
props (Property getter/setter)
propsd (Property getter/setter/deleter)
super ('super(...)' call)
R
React
No live templates are selected

spyder

The Scientific Python Development Environment



VERSION

Spyder 5

Search ...

WELCOME

QUICKSTART

INSTALLATION GUIDE

▶ INTRO VIDEOS

▶ PANES IN DEPTH

▶ SPYDER PLUGINS

▶ TROUBLESHOOTING

▶ WORKSHOPS

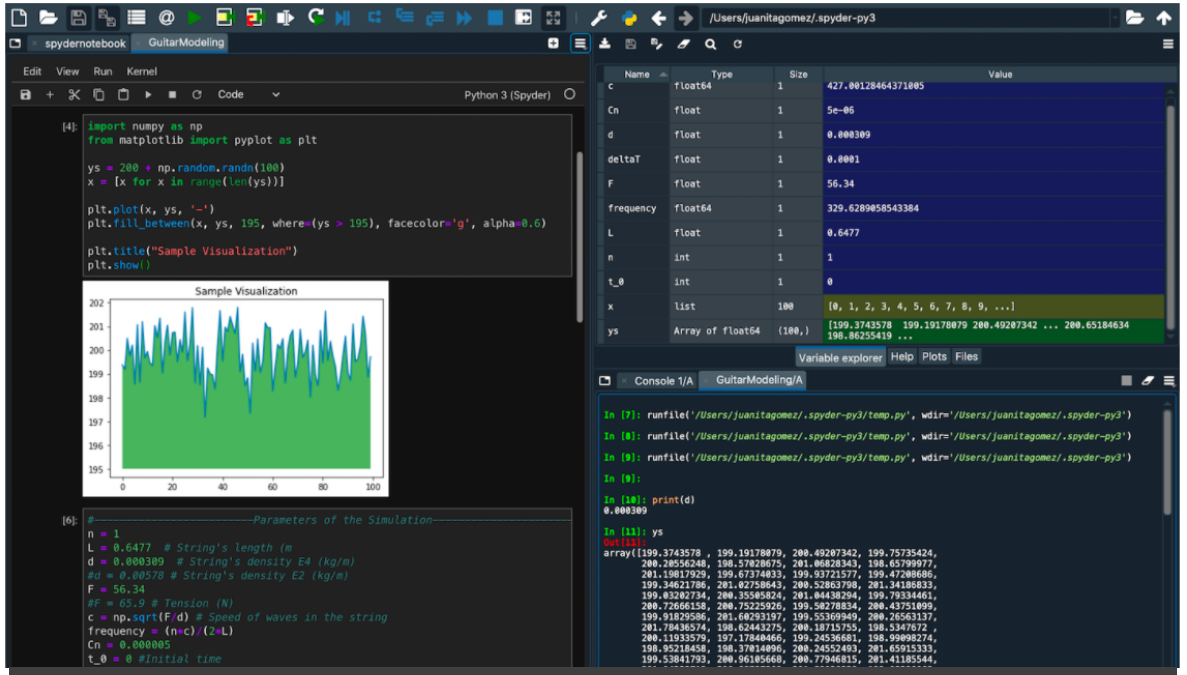
FAQ

Spyder Notebook

Warning

Currently, this plugin is still being ported to Spyder 5, and will likely not yet work or experience serious issues on this version of Spyder. A compatible version is expected soon. Thanks for your patience!

Spyder-notebook is a plugin that allows you to open, edit and interact with Jupyter Notebooks right inside Spyder.



conda install spyder-notebook -c spyder-ide

Edit this page

On this page

Installing the Notebook

Using the Notebook

Connecting an IPython Console

Additional Options

Kurz_Python_Den1.ipynb

Súbor Upraviť Zobraziť Vložiť Runtime Nástroje Pomocník Zmeny sa nedajú uložiť

+ Kód + Text Kopírovať na Disk

Útržky kódu

Filtrovať útržky kódu

- Adding form fields
- Camera Capture
- Cross-output communication
- display.Javascript to execute Jav...
- Downloading files or importing da...

Adding form fields Vložiť

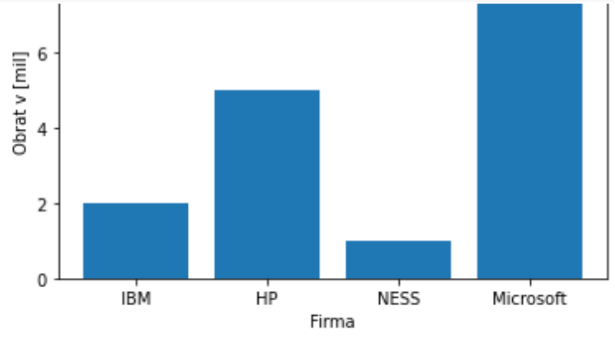
Forms example

Forms support multiple types of fields with type checking including sliders, date pickers, input fields, dropdown menus, and dropdown menus that allow input.

```
#@title Example form fields
#@markdown Forms support many type

no_type_checking = ''  #@param
string_type = 'example'  #@param
slider_value = 142  #@param {type
number = 102  #@param {type: "numb
date = '2010-11-05'  #@param {type
pick_me = "monday"  #@param ['mond
select_or_input = "apples" #@param
#@markdown ---
```

Zobraziť zdrojový zápisník

```
tiene = True
# r g b, c m y k, w
farby_vlastne = ["black","pink", "b", "#CCCC00"]

print(type(y))
print(y)

plt.pie(y, labels = menovky, startangle = 0, explode = vykrojenie, shadow = tiene, colors = farby_vlastne)
plt.legend(title = "Produkty ABC s.r.o.")
plt.title("Analyza predaja produktov za Q1-2022")
plt.show()

x1 = np.array(["IBM", "HP", "NESS", "Microsoft"])
y1 = np.array([2, 5, 1, 9])

plt.bar(x1, y1)
plt.title("Porovnanie IT firiem 2022")
plt.xlabel("Firma")
plt.ylabel("Obrat v [mil]")
plt.show()

nahoda = np.random.normal(100, 10, 200)
print(nahoda)
plt.hist(nahoda)
plt.show()
```




```
[ 79.76069196  99.4155264  114.29926387 101.33767141  88.49106384
 111.70288892  91.32702578 102.53587004 108.38846479 114.34889501
  98.79114202 117.40488367  89.26174251  94.12100639 101.96805716
```

0 s dokončené

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\miros

welcome | untitled0 | Python_I_Za...

Edit View Run Kernel

Code | Python 3 (Spyder)

Kurz Python - 1. deň

Miroslav Reiter | miroslav.reiter@it-academy.sk | https://www.linkedin.com/in/miroslav-reiter/
Kurz Python | https://www.it-academy.sk/kurz/python/ | https://github.com/miroslav-reiter

Komentáre, kódovanie, tlač a docstringy

[1]:
```
# -*- coding: utf-8 -*-
# Toto je komentár (jednoriadkový)
"""Toto je docstring (document string)"""

# Pozor na nespravne zalomovanie riadku (Enter)
# SyntaxError: EOL while scanning string literal
# Emotikony https://emojipedia.org/
print("Python je fajnovy jazyk!")
print(__doc__)
print("🖱 🎲 🔮")
print("Co bolo skorej? --> ", min(['\N{CHICKEN}', '\N{EGG}']))
```

```
Python je fajnovy jazyk!
Toto je docstring (document string)
🖱 🎲 🔮
Co bolo skorej? -->  🐔
```

Premenné a typy

```
...
```

```
Milujem Python
Milujem Python
Milujem Python

Nazov produktu je: Hypoteka pre mladych 2021
Splatka vasej hypoteky je: 600 Eur
Urokova sadzba je: 1.5 % p.a.
Je k dispozicii: False
```

Editor | Notebook

Name | Type | Size | Value

Variable explorer | Help | Plots | Files

Console 1/A

```
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]:
```

IPython console | History

LSP Python: ready | Kite: ready (no index) | conda: base (Python 3.8.5) | main [86] | Line 8, Col 1 | UTF-8 | CRLF | RW | Mem 47% | CPU 17% | 09:26

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\miros\.spyder-py3

C:\Users\miros\AppData\Local\Temp\kite_tutorial.py

temp.py kite_tutorial.py

```
# Welcome to...
#
#          `hmy+.                  ://:
#        .mMMMMMNho:`              NMMm
#       :NMMMMMMMMMMds/.`          NMMm             :ss:
#      +NMMMMMMMMMMMMMMMMmy+       NMMm            -MMMM-   ---
#    `oMMMMMMMMMMMMMMMMMMMMMo      NMMm             /ss/   :MMM+
#   `yMMMMMMMMNshmNMMMMMMMN`       NMMm                    /MMM+
#  .dMMMMMMMMm/hmhssydmMMM+        NMMm     `/yhhy. shhy ohmMMMmhhhh.  ./ydmmmdho-
#  omMMMMMMMd/mMMMMMmhsosy`        NMMm  .omMMmo.  mMMN odmMMMmdddd. omMNdsoshNMNy`
#   .+dMMMMy/mMMMMMMMMMm-          NMMm-yNMMh/`    mMMN   /MMM+     sMMN:`   `:NMMy
#     `-ymo/NMMMMMMMMMMd           NMMMNMMN/       mMMN   :MMM+     MMMNddddddNMMN
#        ``hMMMMMMMMMMM:           NMMm+mMMNs.     mMMN   :MMM+     MMMh//////////:
#          `:yNMMMMMMMh            NMMm `/dMMNy-   mMMN   :MMM+  `. sMMNo`    `-:
#            .+mMMMMM-             NMMm   `/dMMNy- mMMN   .MMMNddNN/ +NMMNdhydNNMs
#              `:yMMMy             yhhs     `/hhhh shhs    :ymmmdho:  `/sdmmmmhs/`
#                 `om.

""" Kite is your Python programming copilot. Kite will try to show you the
    right information at the right time as you code to prevent you from context
    switching out of your current line of thought.

    This tutorial will teach you how to use all of Kite's core features. You
    should be able to learn everything in 5 minutes.

    If you get stuck at any point, please visit https://help.kite.com/ or file
    an issue at https://github.com/kiteco/issue-tracker.
"""



""" PART 0: BEFORE WE START ============================================

    Spyder will by default try to start the Kite backend when the editor first
    starts. You can change this behavior by opening settings, clicking on
    "Completion and linting", "Advanced", and then changing Kite's "Start Kite
    Engine on editor startup" setting.

    Look for the Kite indicator in the bottom left corner of Spyder's status
    bar — It will tell you if Kite is ready and working. If the indicator reads
    "not running", then you'll have to start the Kite Engine manually before
    proceeding with the rest of this tutorial.
"""
```

Name	Type	Size	Value
a	int	1	5
b	str	1	Karol

Variable explorer Help Plots Files

Console 1/A

```
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/miros/.spyder-py3/temp.py', wdir='C:/Users/miros/.spyder-py3')

In [2]: runfile('C:/Users/miros/.spyder-py3/temp.py', wdir='C:/Users/miros/.spyder-py3')
5
Karol

In [3]:
```

IPython console History

Štart

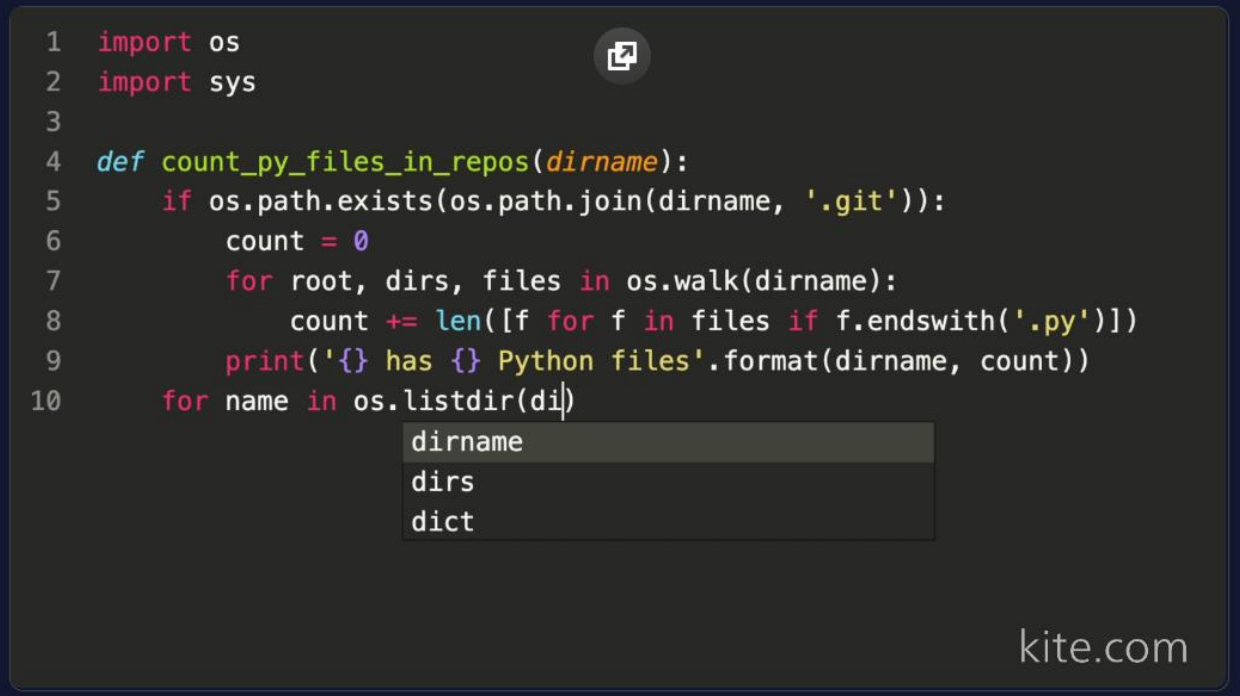
LSP Python: ready Kite: ready conda: base (Python 3.8.5) main [86] Line 19, Col 1 UTF-8-GUESSED CRLF RW Mem 41% CPU 6% 21:57

kite

Code Faster. Stay in Flow.

Kite adds AI powered code completions to your code editor, giving developers superpowers.

Download for Free

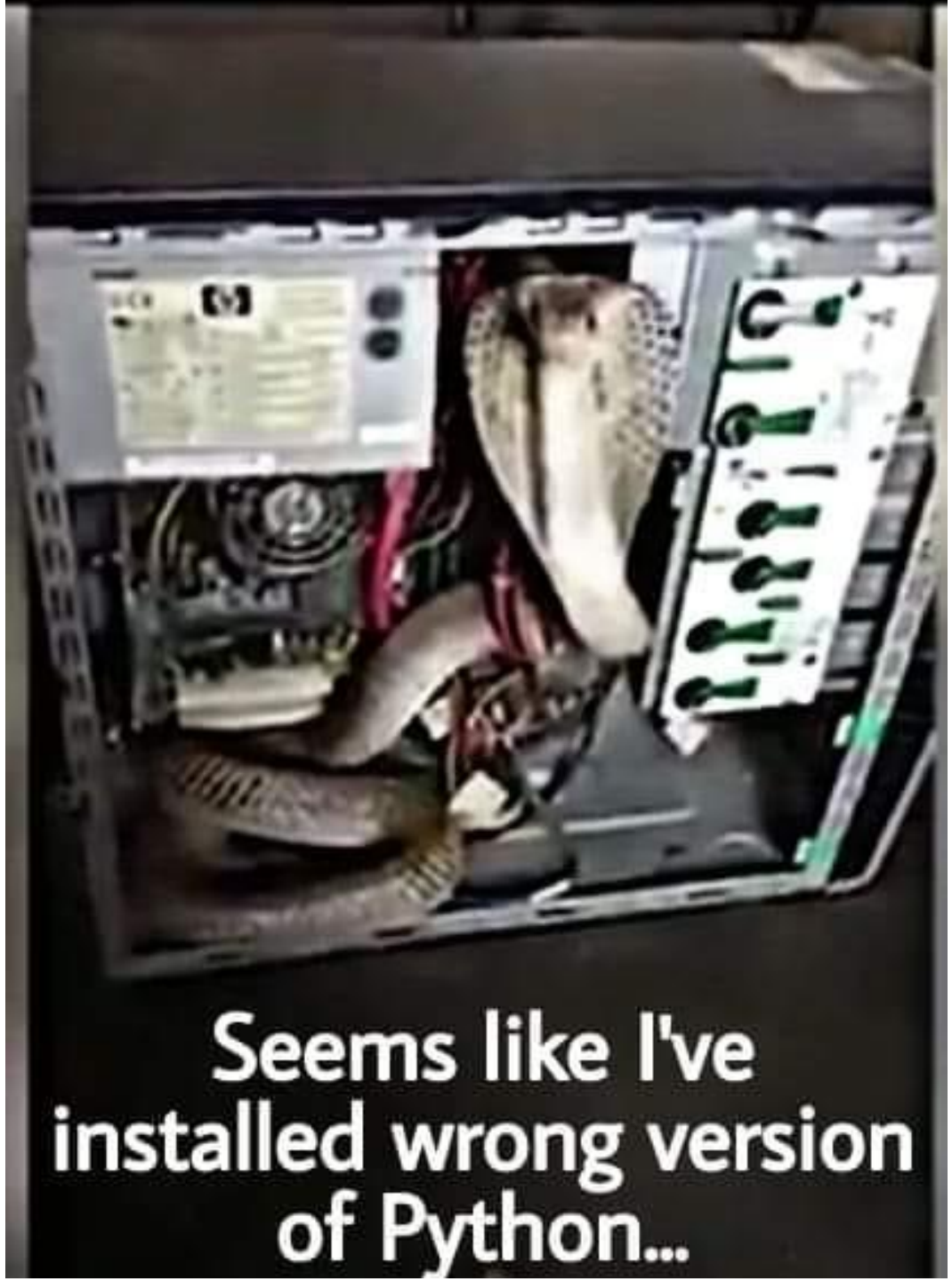


```
1  import os
2  import sys
3
4  def count_py_files_in_repos(dirname):
5      i
```

C:\Windows\System32\cmd.exe - pip install matlib

```
Microsoft Windows [Version 10.0.16299.785]
(c) 2017 Microsoft Corporation. Všetky práva vyhradené.

C:\Windows\System32>cd C:\Program Files (x86)\Python37-32\Scripts

C:\Program Files (x86)\Python37-32\Scripts>pip instal xlwt
ERROR: unknown command "instal" - maybe you meant "install"

C:\Program Files (x86)\Python37-32\Scripts>pip install xlwt
Collecting xlwt
  Downloading https://files.pythonhosted.org/packages/44/48/def306413b25c3d01753603b1a222a011b8621aed27cd7f89cbc27e6b0f4/xlwt-1.3.0-py2.py3-none-any.
whl (99kB)
    100% |████████████████████████████████| 102kB 826kB/s
Installing collected packages: xlwt
Could not install packages due to an EnvironmentError: [WinError 5] Access is denied: 'c:\\program files (x86)\\python37-32\\Lib\\site-packages\\xlwt
'
Consider using the `--user` option or check the permissions.

You are using pip version 10.0.1, however version 18.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Program Files (x86)\Python37-32\Scripts>pip install matlib
Collecting matlib
```

PIP a easy install



sys Variables

argv	Command line args
builtin_module_names	Linked C modules
byteorder	Native byte order
check_interval	Signal check frequency
exec_prefix	Root directory
executable	Name of executable
exitfunc	Exit function name
modules	Loaded modules
path	Search path
platform	Current platform
stdin, stdout, stderr	File objects for I/O
version_info	Python version info
winver	Version number

sys.argv for *$ python foo.py bar -c qux --h*

sys.argv[0]	foo.py
sys.argv[1]	bar
sys.argv[2]	-c
sys.argv[3]	qux
sys.argv[4]	--h

os Variables

altsep	Alternative sep
curdir	Current dir string
defpath	Default search path
devnull	Path of null device
extsep	Extension separator
linesep	Line separator
name	Name of OS
pardir	Parent dir string
pathsep	Patch separator
sep	Path separator

Note Registered OS names: "posix", "nt", "mac", "os2", "ce", "java", "riscos"

Class Special Methods

__new__(cls)	__lt__(self, other)
__init__(self, args)	__le__(self, other)
__del__(self)	__gt__(self, other)
__repr__(self)	__ge__(self, other)
__str__(self)	__eq__(self, other)
__cmp__(self, other)	__ne__(self, other)
__index__(self)	__nonzero__(self)
__hash__(self)	
__getattr__(self, name)	
__getattribute__(self, name)	
__setattr__(self, name, attr)	
__delattr__(self, name)	
__call__(self, args, kwargs)	

String Methods

capitalize() *	lstrip()
center(width)	partition(sep)
count(sub, start, end)	replace(old, new)
decode()	rfind(sub, start ,end)
encode()	rindex(sub, start, end)
endswith(sub)	rjust(width)
expandtabs()	rpartition(sep)
find(sub, start, end)	rsplit(sep)
index(sub, start, end)	rstrip()
isalnum() *	split(sep)
isalpha() *	splitlines()
isdigit() *	startswith(sub)
islower() *	strip()
isspace() *	swapcase() *
istitle() *	title() *
isupper() *	translate(table)
join()	upper() *
ljust(width)	zfill(width)
lower() *	

Note Methods marked * are locale dependant for 8-bit strings.

List Methods

append(item)	pop(position)
count(item)	remove(item)
extend(list)	reverse()
index(item)	sort()
insert(position, item)	

File Methods

close()	readlines(size)
flush()	seek(offset)
fileno()	tell()
isatty()	truncate(size)
next()	write(string)
read(size)	writelines(list)
readline(size)	

Indexes and Slices (of *a=[0,1,2,3,4,5]*)

len(a)	6
a[0]	0
a[5]	5
a[-1]	5
a[-2]	4
a[1:]	[1,2,3,4,5]
a[:5]	[0,1,2,3,4]
a[:-2]	[0,1,2,3]
a[1:3]	[1,2]
a[1:-1]	[1,2,3,4]
b=a[:]	Shallow copy of a

Datetime Methods

today()	fromordinal(ordinal)
now(timezoneinfo)	combine(date, time)
utcnow()	strptime(date, format)
fromtimestamp(timestamp)	
utcfromtimestamp(timestamp)	

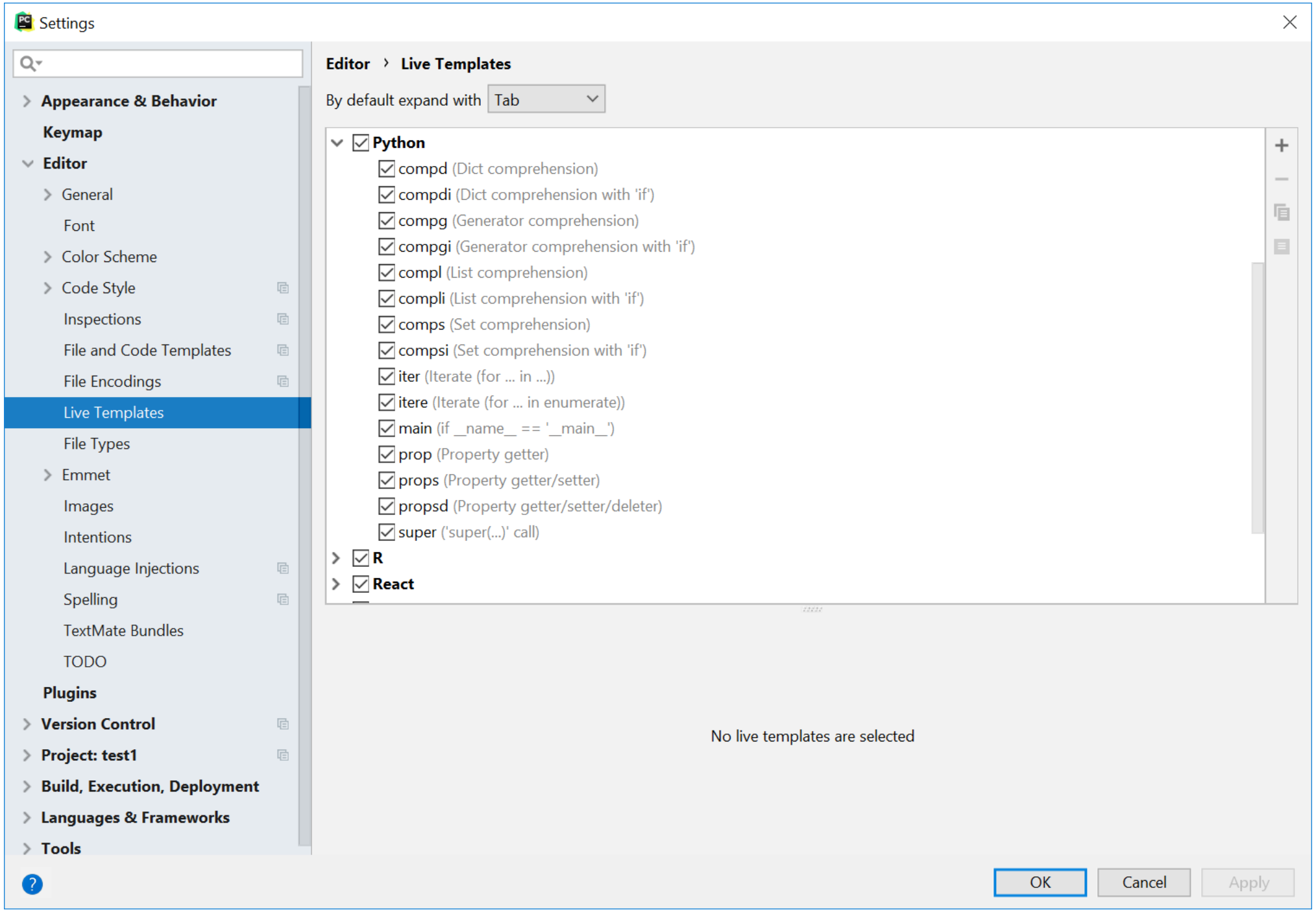
Time Methods

replace()	utcoffset()
isoformat()	dst()
__str__()	tzname()
strftime(format)	

Date Formatting (strftime and strptime)

%a	Abbreviated weekday (Sun)
%A	Weekday (Sunday)
%b	Abbreviated month name (Jan)
%B	Month name (January)
%c	Date and time
%d	Day (leading zeros) (01 to 31)
%H	24 hour (leading zeros) (00 to 23)
%I	12 hour (leading zeros) (01 to 12)
%j	Day of year (001 to 366)
%m	Month (01 to 12)
%M	Minute (00 to 59)
%p	AM or PM
%S	Second (00 to 61 [4])
%U	Week number [1] (00 to 53)
%w	Weekday [2] (0 to 6)
%W	Week number [3] (00 to 53)
%x	Date
%X	Time
%y	Year without century (00 to 99)
%Y	Year (2008)
%Z	Time zone (GMT)
%%	A literal "%" character (%)

1. Sunday as start of week. All days in a new year preceding the first Sunday are considered to be in week 0.
2. 0 is Sunday, 6 is Saturday.
3. Monday as start of week. All days in a new year preceding the first Monday are considered to be in week 0.
4. This is not a mistake. Range takes account of leap and double-leap seconds.


Settings
Appearance & Behavior
Keymap
Editor
General
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Emmet
Images
Intentions
Language Injections
Spelling
TextMate Bundles
TODO
Plugins
Version Control
Project: test1
Build, Execution, Deployment
Languages & Frameworks
Tools
Editor › Live Templates
By default expand with Tab
Python
compd (Dict comprehension)
compdi (Dict comprehension with 'if')
compg (Generator comprehension)
compgi (Generator comprehension with 'if')
compl (List comprehension)
compli (List comprehension with 'if')
comps (Set comprehension)
compsi (Set comprehension with 'if')
iter (Iterate (for ... in ...))
itere (Iterate (for ... in enumerate))
main (if __name__ == '__main__')
prop (Property getter)
props (Property getter/setter)
propsd (Property getter/setter/deleter)
super ('super(...)' call)
R
React
No live templates are selected
OK
Cancel
Apply

Asercia

Testovanie programu

Princíp raise-if

Ak je výraz True, pokračuje sa vo vykonávaní príkazov

Ak je výraz False, vyvolá sa výnimka AssertionError

```
def kelvinNaFahrenheit(teplota):
    assert (teplota >= 0),"Menej ako absolutna 0!"
    return ((teplota-273)*1.8)+32

print kelvinNaFahrenheit(100)
print int(kelvinNaFahrenheit(500.55))
print kelvinNaFahrenheit(-5)
```

Spracovanie výnimiek

- Try
- Except

- Try
- Finally

- Try
- Except
- Finally

```
try:
    fh = open("testfile", "w")
    fh.write("Toto je moj testovaci subor...")
except IOError:
    print "Chyba: nemozem najst subor alebo citat data"
else:
    print "Obsah uspesne zapisany"
    fh.close()
```

Otváranie a zatváranie súborov

```
f = open('data.txt')
try:
    data = f.read()
finally:
    f.close()
```

```
with open('data.txt') as f:
    data = f.read()
```

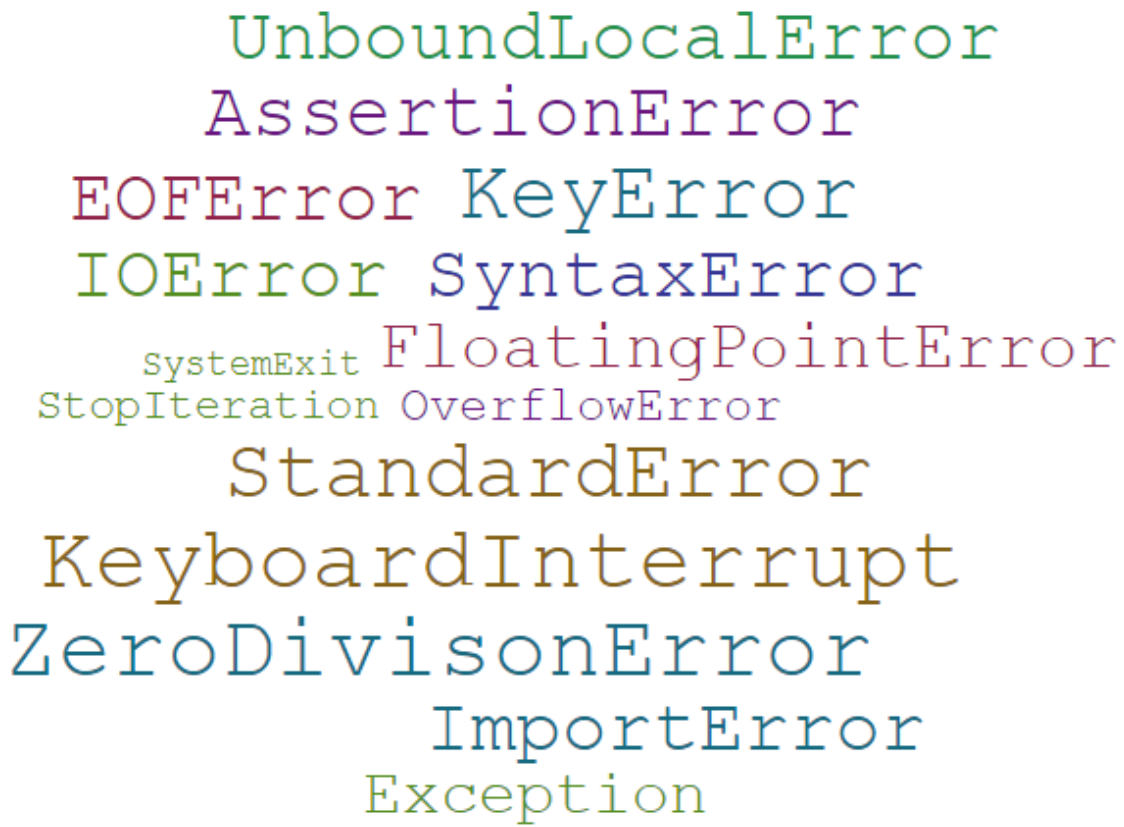
IT ACADEMY

Odstraňovanie súborov

```
try:
    os.remove('somefile.tmp')
except OSError:
    pass


with ignored(OSError):
    os.remove('somefile.tmp')
```

IT ACADEMY

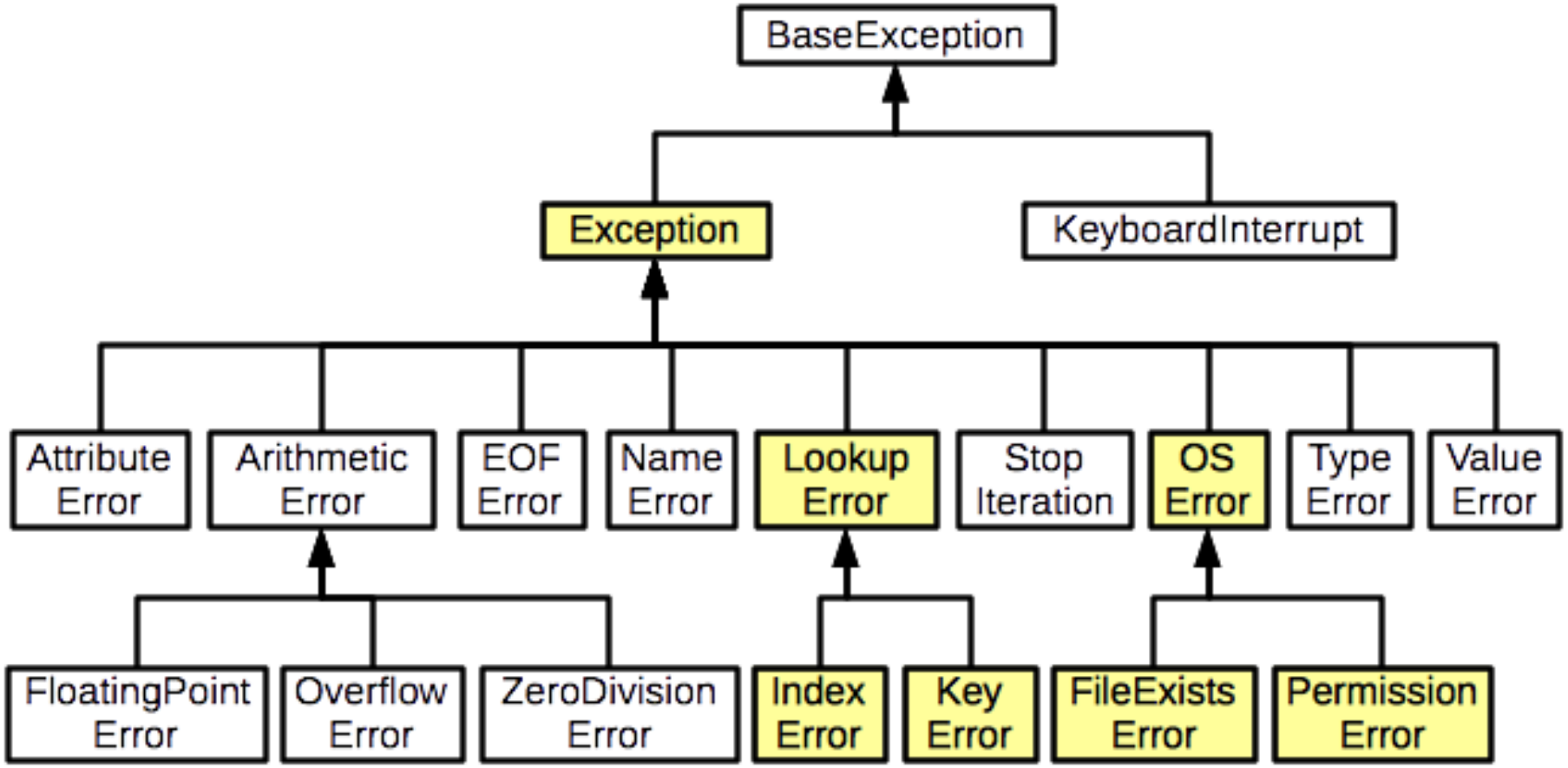
Spracovanie výnimiek




```
def vypocitajCenuDPH(cena):
    try:
        return int(cena)
    except ValueError as exVal:
        print "Argument neobsahuje cislo"
        print exVal
##          print exVal.message
    except TypeError as exType:
            print "Nezadal si argument..."

print vypocitajCenuDPH("xyz")
print vypocitajCenuDPH(100)
print vypocitajCenuDPH(None)
print vypocitajCenuDPH()
```

IT ACADEMY

Diagram tried výnimiek



IT ACADEMY

```
Multiexcept: except (exception1, exception2) as e
```

Premenné triedy

```
class Zamestnanci:
    pocetZam = 0

    def __init__(self, meno, plat):
        self.meno = meno
        self.plat = plat
        Zamestnanci.pocetZam += 1

    def zobrazPocet(self):
        print "Pocet zamestnancov %d" % Zamestnanci.pocetZam

    def zobrazZamestnancov(self):
        print "meno : ", self.meno,  ", plat: ", self.plat
```

IT ACADEMY

Iterátory

Sekvencia automaticky vytvorí iterátor:

- **for** i **in** sekvencia: urob_nieco(i)

Čo je ekvivalentné:

```
1.  m = iter(sekvencia)
2.  while 1:
3.      try:
4.          i = m.next()
5.      except StopIteration:
6.          break
7.      urob_nieco(i)
```

Iterator	Arguments	Results	Example
`count()`	start, [step]	start, start+step, start+2*step, ...	`count(10) --> 10 11 12 13 14 ...`
`cycle()`	p	p0, p1, ... plast, p0, p1, ...	`cycle('ABCD') --> A B C D A B C D ...`
`repeat()`	elem [,n]	elem, elem, elem, ... endlessly or up to n times	`repeat(10, 3) --> 10 10 10`

Getre a setre

```
class Zamestnanec:
    pocetZam = 0

    def __init__(self, meno, plat):
        self.meno = meno
        self.plat = plat
        Zamestnanec.pocetZam += 1

    def zobrazPocet(self):
        print "Pocet zamestnancov %d" % Zamestnanec.pocetZam

    def zobrazZamestnancov(self):
        print "meno : ", self.meno,  ", plat: ", self.plat


zam1 = Zamestnanec("Adam", 2000)
zam2 = Zamestnanec("Eva", 5000)
```

```
##print zam1.vek
zam1.vek = 7   # Vytvor atribut vek
print zam1.vek
zam1.vek = 8   # Modifikuj atribut vek
print zam1.vek
del zam1.vek   # Vymaz atribut vek

print "Ma zamestnanec vek:", hasattr(zam2, "vek")
setattr(zam2, "vek", 8) # vy
print getattr(zam2, "vek")
delattr(zam2,"vek")
```

IT ACADEMY

Príkaz del

Zabudované atribúty triedy

- **__dict__**: Dictionary containing the class's namespace.
- **__doc__**: Class documentation string or none, if undefined.
- **__name__**: Class name.
- **__module__**: Module name in which the class is defined. This attribute is "__main__" in interactive mode.
- **__bases__**: A possibly empty tuple containing the base classes, in the order of their occurrence in the base class list.

IT ACADEMY

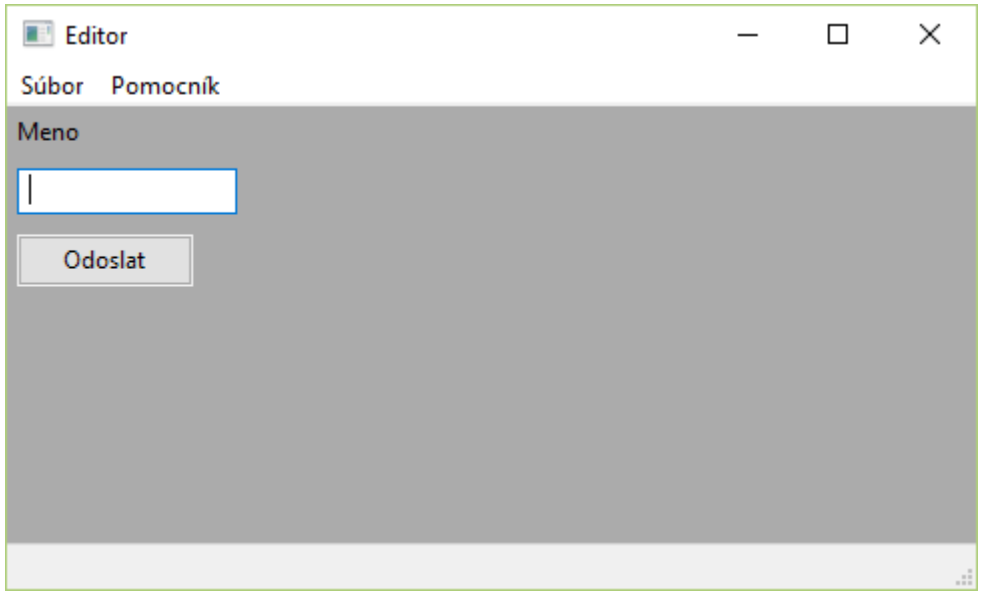
Knižnice

```
import math as matematika

def vypisAhoj():
    print "Ahoj novy svet"

meno = "Karol"
```

```
>>> vypisAhoj()
Ahoj svet
>>> meno
'Karol'
>>> import kniznica
>>> vypisAhoj()
Ahoj svet
>>> kniznica.vypisAhoj()
Ahoj novy svet
>>> reload(kniznica)
<module 'kniznica' from 'C:/Users/Miroslav/Desktop\kniznica.pyc'>
>>> vypisAhoj()
Ahoj svet
```

IT ACADEMY

GUI




```
if __name__ == '__main__':
    app = wx.App(0)
    frame = frmEditor(None)
    frame.Show()
    app.MainLoop()
```

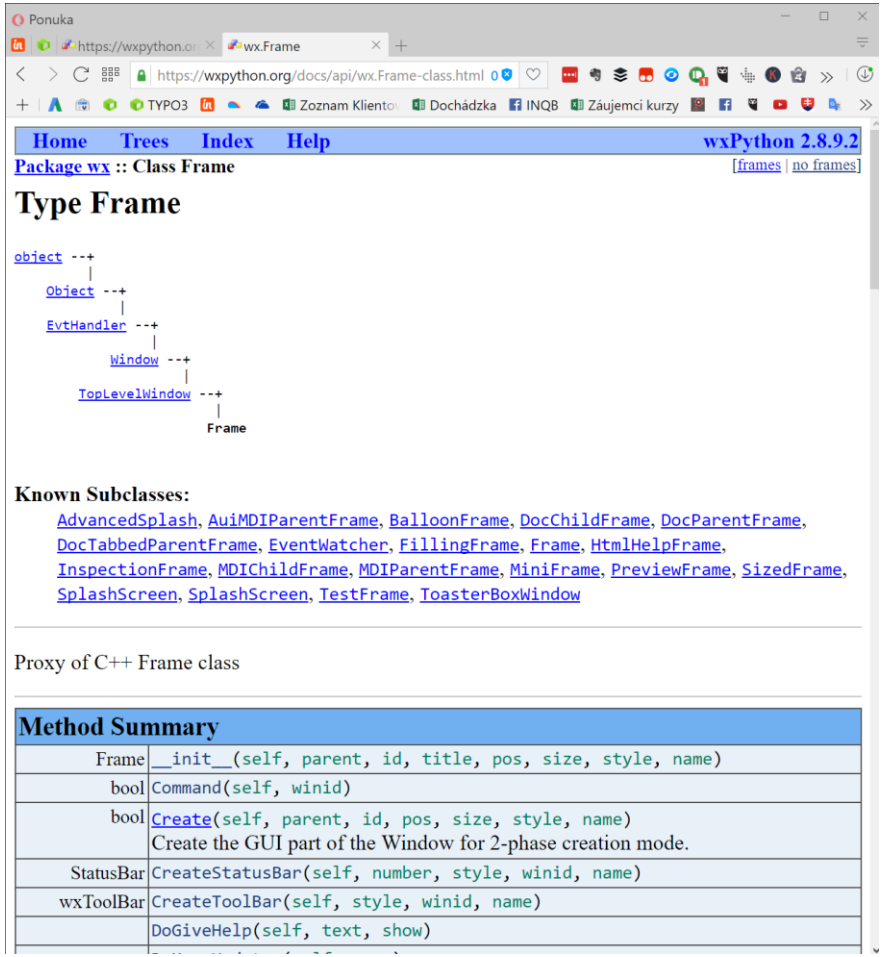
IT ACADEMY

Zámky/locky

```
# Make a lock
lock = threading.Lock()


# Old-way to use a lock
lock.acquire()
try:
    print 'Critical section 1'
    print 'Critical section 2'
finally:
    lock.release()


# New-way to use a lock
with lock:
    print 'Critical section 1'
    print 'Critical section 2'
```

IT ACADEMY

Dokumentácia



IT ACADEMY

Dôležité Zdroje

Slovensko a Česko

- Albatrosmedia
- Kopp
- Grada
- Wolters Kluwer
- BEN
- Veda

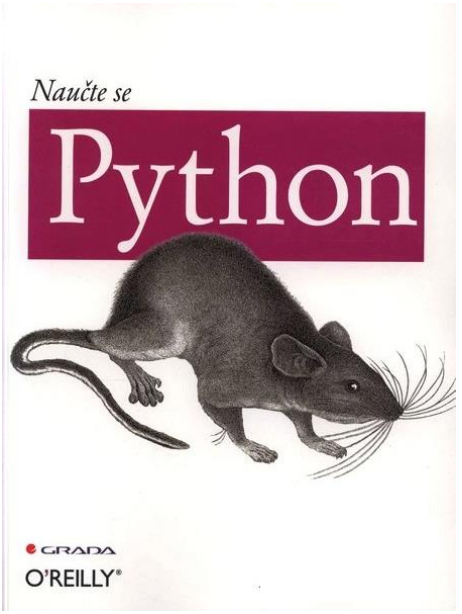
Zahraničie

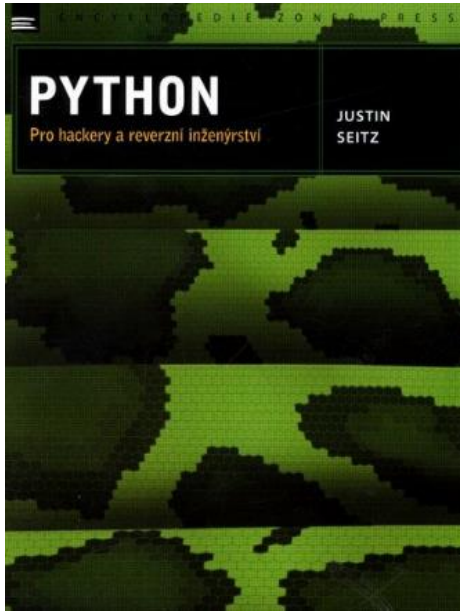
- O'Reilly
- Manning
- Packt
- Apress
- Wiley
- No Starch Press

YouTube Tutorialy

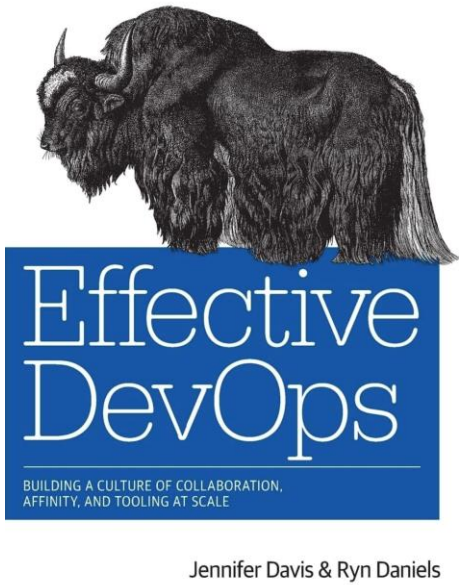
- IT Academy

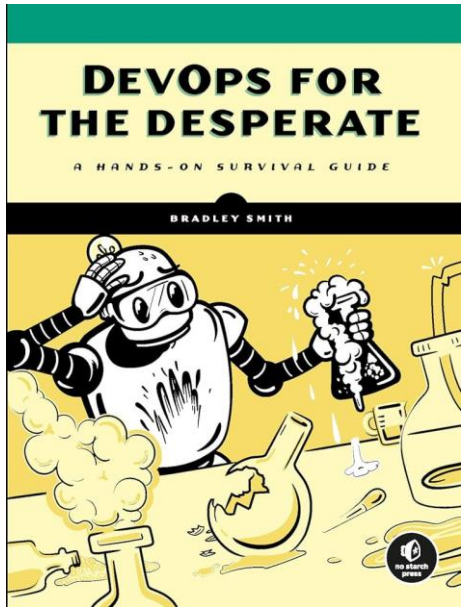
IT Academy

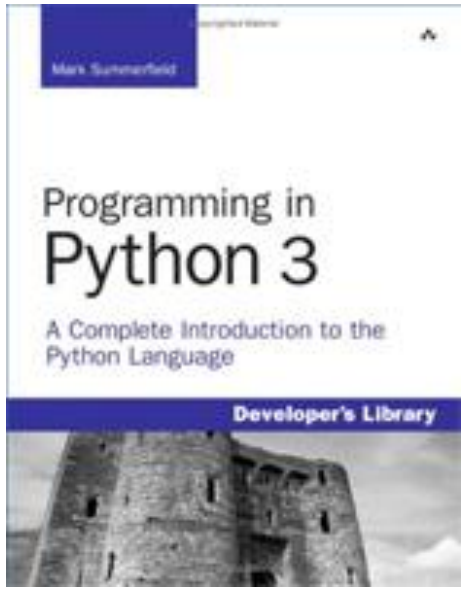


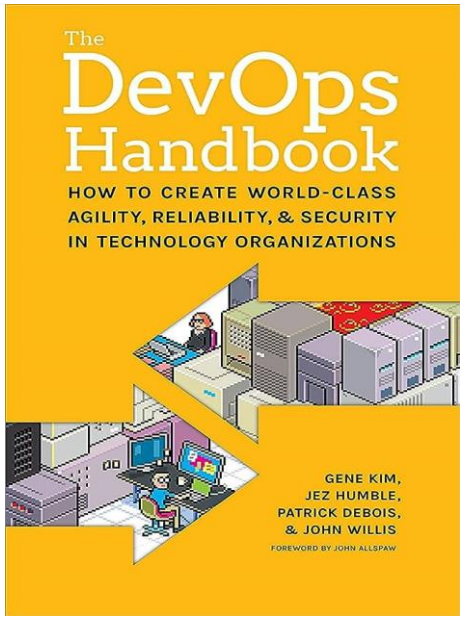


Čo sa oplatí/neoplatí prečítať SK/CZ?









Čo sa oplatí/neoplatí prečítať EN?

Čo Odporúčam si Pozrieť?

1. https://docs.python.org/3/
2. https://realpython.com/tutorials/best-practices/
3. https://google.github.io/styleguide/pyguide.html
4. https://docs.python.org/3/
5. http://python2013.input.sk/19prednaska
6. https://realpython.com/python3-object-oriented-programming/
7. https://jeffknupp.com/blog/2014/06/18/improve-your-python-python-classes-and-object-oriented-programming/
8. https://overiq.com/python-101/inheritance-and-polymorphism-in-python/
9. https://www.javatpoint.com/python-oops-concepts
10. https://www.programiz.com/python-programming/object-oriented-programming

TOP Klávesové Skratky

TOP Klávesové Skratky

Práca s IDE

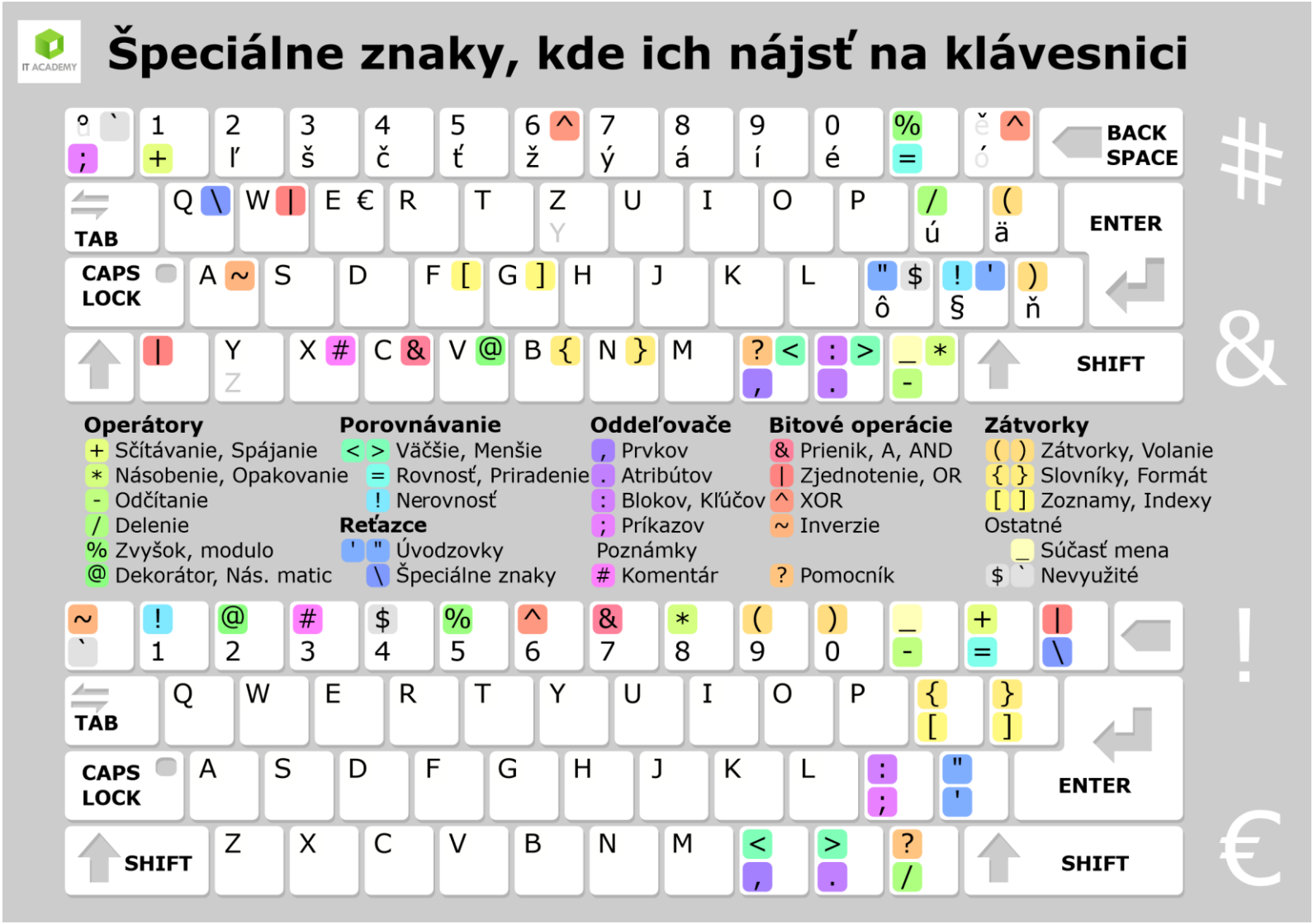
• Ctrl + D	Delete zmaž riadok
• **Ctrl + Space**	Asistent kódu
• **Ctrl + /**	Komentáre
• Ctrl + A	Označ všetko
• **Alt + /**	Dokonči slovo
• Ctrl + F	Hľadanie a náhrady
• Ctrl + Shift + F	Kompakt režim
• Ctrl + Shift + S	Ulož všetko

Práca s browserom

• Ctrl + T	Vytvor nový tab
• Ctrl + W	Zatvor aktuálny tab
• Ctrl + Shift + W	Zatvor všetky taby
• **Ctrl + Shift + T**	Otvor posledný tab
• Ctrl + Shift + J/F12	Web console
• **F11**	Fullscreen

ALT + Ins Code Generator

Efektívne Používanie Klávesnice



Hľadať na Facebooku

Vývojári

Verejná skupina · 7,4 tis. členov

+ Pozvať | Zdieľať | Člen

Diskusia | Ľudia | Podujatia | Médiá | Súbory

Napíšte niečo...

Reel | Fotka/video | Anketa

Informácie

Skupina softvérových vývojárov. Táto skupina by mala byť miestom, kde sa môžu českoslovenkí vývojári vzájomne spoznať, vyžiadať si konštruktívnu... **Zobraziť viac**

Verejná
Členov skupiny a ich príspevky bude vidieť ktokoľvek.

Programátori

Verejná skupina · 11,1 tis. členov

+ Pozvať | Člen

Diskusia | Vybrané | Ľudia | Podujatia | Médiá | Súbory

Napíšte niečo...

Reel | Fotka/video | Anketa

Informácie

Táto skupina slúži na dohadzovanie si kšeftíkov a pre hľadačov programátorov / vývojárov.

Verejná
Členov skupiny a ich príspevky bude vidieť ktokoľvek.

Viditeľná

IT ACADEMY

Home

PUBLIC
Questions
Tags
Users

COLLECTIVES
Explore Collectives

FIND A JOB
Jobs
Companies

TEAMS
Create free Team

Tags

A tag is a keyword or label that categorizes your question with other, similar questions. Using the right tags makes it easier for others to find and answer your question.

Show all tag synonyms

python

Popular | Name | New

python

Python is a multi-paradigm, dynamically typed, multi-purpose programming language. It is designed to be quick to learn, understand, and...

1870168 questions — 695 asked today, 6387 this week

python-3.x

USE ONLY IF YOUR QUESTION IS VERSION-SPECIFIC. For questions about Python programming that are specific to version 3+ of the language...

303562 questions — 93 asked today, 836 this week

python-2.7

Python 2.7 is the last major version in the 2.x series, and is no longer maintained since January 1st 2020. Use the generic [python] tag on all Python...

94965 questions — 24 asked this week, 106 this month

python-requests

USE ONLY FOR THE PYTHON REQUESTS LIBRARY. Requests is a full-featured Python HTTP library with an easy-to-use, logical API.

18697 questions — 8 asked today, 57 this week

python-imaging-library

The Python Imaging Library (PIL) provides the Python language with a de-facto standard foundation for image work. PIL's API is lightweight but...

7883 questions 5 asked today, 38 this week

wxpython

wxPython is a Python wrapper for the cross-platform C++ GUI API wxWidgets.

7047 questions — 7 asked this week, 14 this month

ipython

IPython is a feature-rich interactive shell for Python, and provides a kernel for frontends such as IPython Notebook and Jupyter Notebook.

6886 questions — 5 asked this week, 26 this month

python-3.6

Version of the Python programming language released in December 2016. For issues specific to Python 3.6. Use more generic [python] and [python-3....

5602 questions — 11 asked this week, 24 this month

python-asyncio

to be used for the asyncio Python package which provides mechanisms for writing single-threaded concurrent code. The asyncio package provides...

5492 questions — 29 asked this week, 125 this month

python-import

For questions about importing modules in Python

5119 questions — 11 asked this week, 47 this month

python-multiprocessing

multiprocessing is a package that supports spawning processes using an API similar to the threading module in python programming language.

4036 questions — 12 asked this week, 46 this month

python-3.7

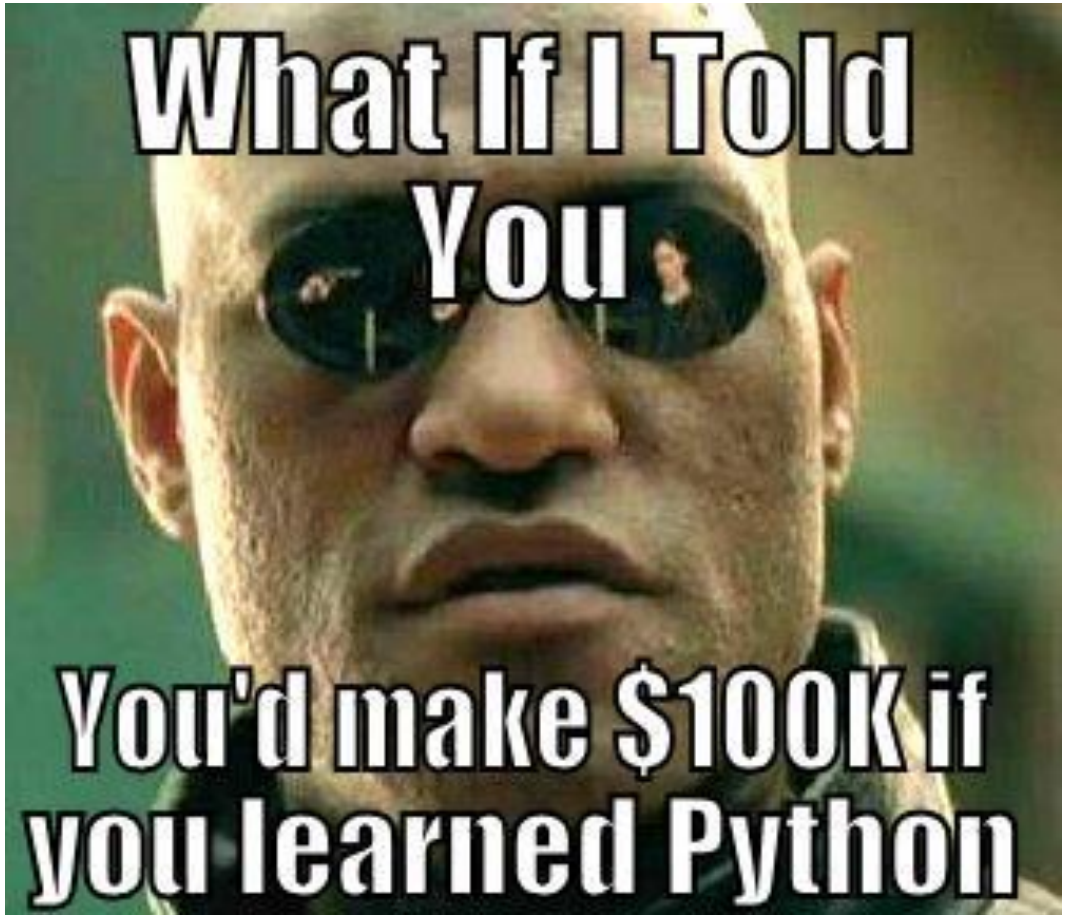
Version of the Python programming language released in June 27, 2018. For issues that are specific to Python 3.7. Use the more generic [python] and...

4034 questions — 5 asked this week, 21 this month


NOVINKY
PROGRAM
EDUSUMMIT
SPONZORI
KÓDEX SPRÁVANIA
ABOUT
KÚP SI LÍSTOK
PYCON SK 2022
Bratislava
KÚP SI LÍSTOK

Zen filozofia Pythonu

1. Krásny je lepší než škaredý
2. Explicitný je lepší ako implicitný
3. Jednoduchý je lepší ako zložitý
4. Zložitý je lepší ako komplikovaný
5. Plochý je lepší ako vnorený
6. Riedky je lepší ako hustý
7. Na čitateľnosti záleží
8. Praktickosť vyhráva nad čistotou




```
import this
```

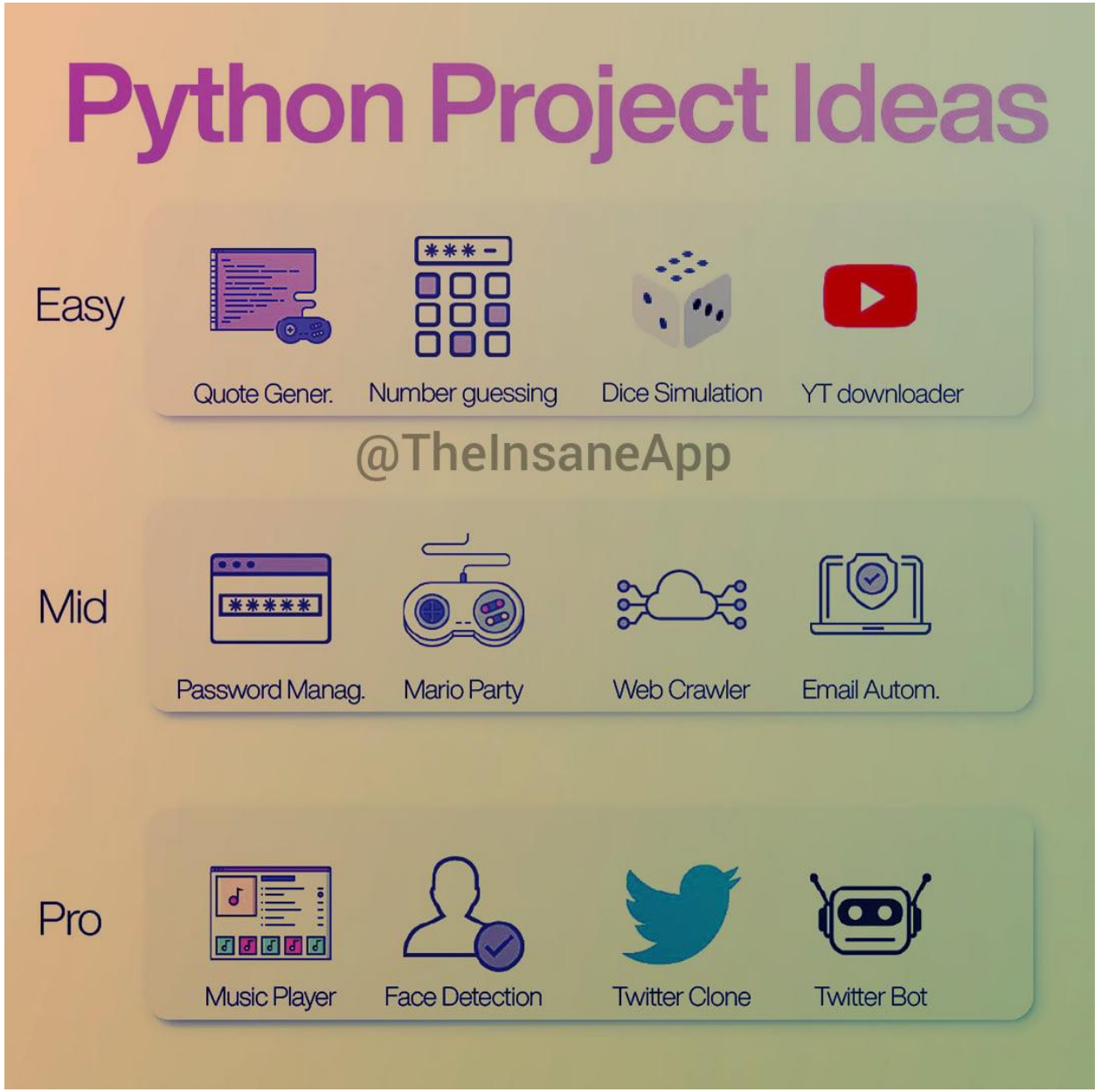
Čaká nás krásna budúcnosť

```
>>> from __future__ import braces
SyntaxError: not a chance (<pyshell#13>, line 2)
>>>
```

IT ACADEMY

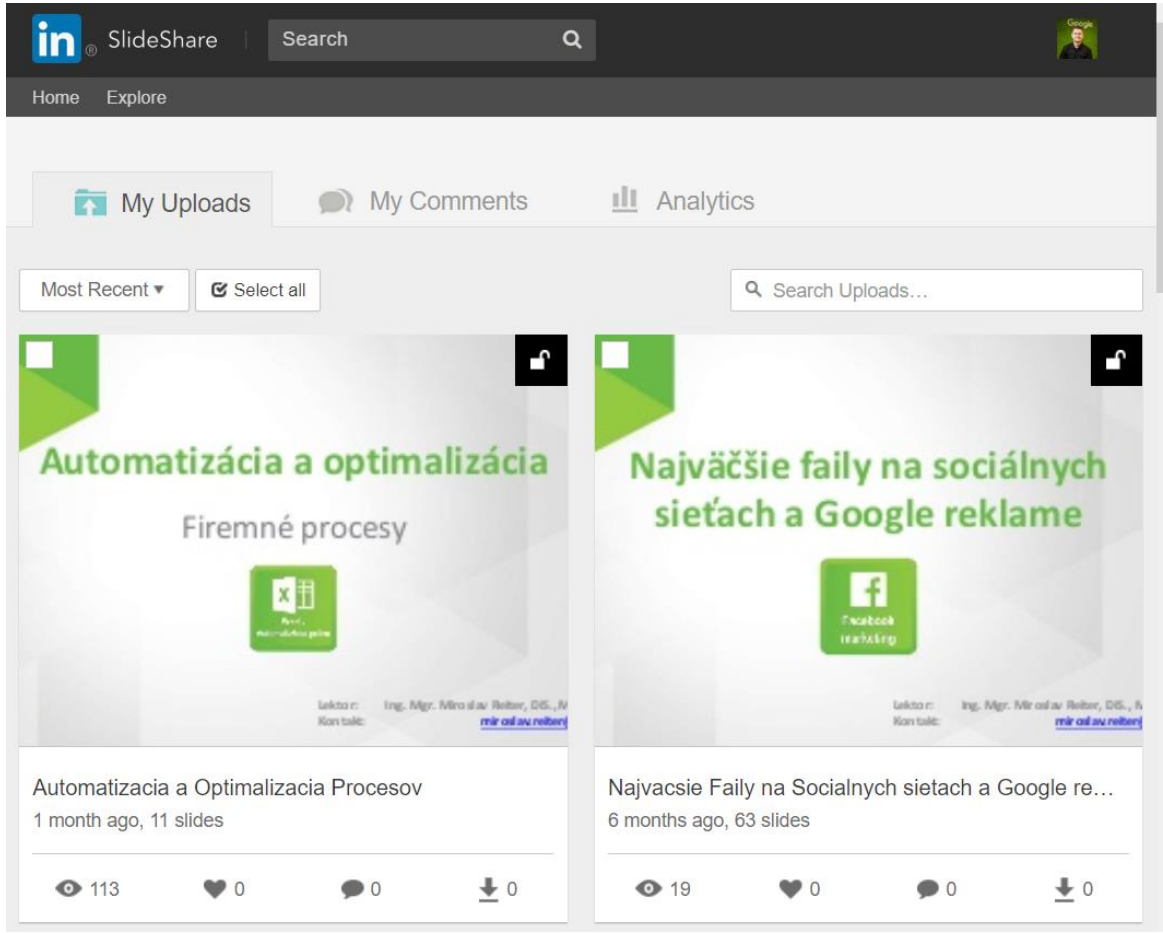
No future {} a ;

Inšpirácia projekty



Ako skončíme?

1. **Pridaj** si ma na **LinkedIn**
   - www.linkedin.com/in/miroslav-reiter
2. **Materiály po prednáške**
   - https://1drv.ms/p/s!AlrLrycbTQ1a19sfc1MmNNnYMaluWA?e=FTUlTc



IT ACADEMY

Spracovanie a vizualizácia dát v Pythone

Základy dátovej analýzy, spracovanie a vizualizácia dát v programovacom jazyku Python.
Prerekvizitou tohto kurzu sú základné zručnosti v programovacom jazyku Python. Pokiaľ ste absolvovali predošlý kurz s názvom „Základy programovacieho jazyka Python“ prípadne „Objektovo orientované programovane v Pythone“, určite spĺňate základné prepoklady pre absolvovanie tohto kurzu.
Opäť (ako v predšlých kurzoch) budeme pracovať v prostredí Jupyter Notebook, ktorý si môžete nainštalovať aj doma na svojom osobnom počítači prostredníctom GUI Anaconda Navigator.

1. **Základný balík NumPy:**
   - Nainštalovanie knižnice NumPy
   - Vytvorenie NumPy polí
   - Dátové typy a operácie s NumPy poľami
   - Indexovanie a prechádzanie NumPy polí
   - Univerzálne NumPy funkcie
   - Spracovanie a filtrovanie NumPy polí
   - Zhrnutie nových znalostí

2. **Vizualizácia dát:**
   - Nainštalovanie knižnice pre vizualizáciu dát
   - Úvod do vizualizácie v knižniciach Matplotlib a Seaborn
   - Vizualizácia dát na rozličných príkladoch v spomínaných knižniciach
   - Prispôsobenie(Customization) výstupov grafov podľa našej potreby

3. **Spracovanie dát:**
   - Nainštalovanie a import Pandas knižnice
   - Vytváranie Pandas dataframov
   - Načítanie súborov (.txt, .xlsx, .csv)
   - Spracovanie a analýza dát zo súborov(.txt, .xlsx, .csv)
   - Operácie s dátami
   - Vytváranie grafov
   - Zhrnutie

Trvanie:
10 hodín (2 dni)
Cena:
€0,-
Kategória:
Python
Registrácia (počet prihlásených):
18. - 19. 5. 2022 (43/50)

Zadarmo

1. Kurzy SAV
2. Kurzy Grow with Google
3. YouTube kanál IT Academy

Platené

Moje kurzy na www.vita.sk

https://www.youtube.com/c/IT-AcademySK

Ako sa s nami Spojiť?

ADRESA: IT Academy, s. r. o.

Budova KOLOSEO prízemie
Tomášikova 50/A

831 04 Bratislava

WEB: www.it-academy.sk

E-MAIL: info@it-academy.sk

TELEFÓN: +421 917 095 406

Ako Vieme Pomôcť?

#Copywriting #Školenia #Zamestnanci #Pomáhame #Rast

#Projekty #Certifikácie #Kurzy #Tréningy #Vzdelávanie

#PPC Kampane #Elearning #Mentoring #Konzultácie #Online

#Programovanie #Vývoj #Marketing #Reklama #Prenájom Techniky

Domov
Shorts
Odbery

Moje
História

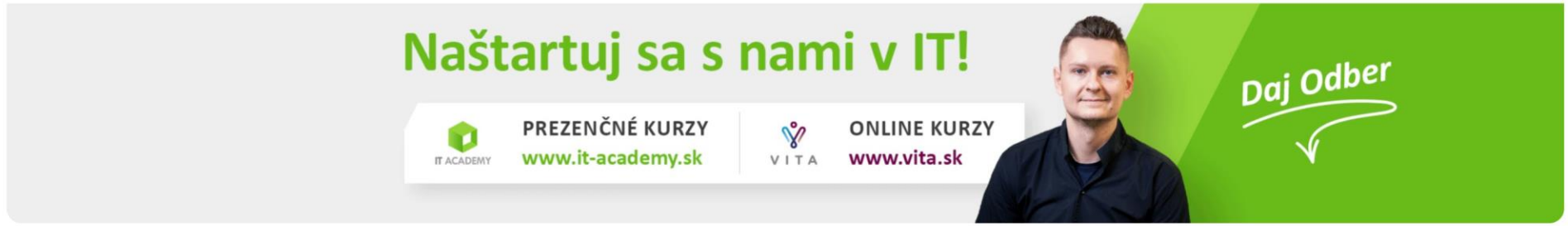
Ak chcete k videám pridať označenie páči sa mi, komentovať alebo sa prihlásiť na odber, musíte sa prihlásiť.

Prihlásiť sa

Preskúmať
Trendy
Hudba
Filmy
Hry
Šport

Prehliadať kanály

Viac zo služby YouTube
YouTube Premium
YouTube Music
YouTube Kids



IT Academy

@IT-Academy · 7,02 tis. odberateľov · 938 videí

Online certifikované kurzy a školenia IT, marketingu a manažmentu.

vita.sk a 8 ďalších odkazov

Odoberať

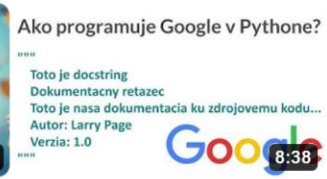
Domov Videá Shorts Naživo Zoznamy Komunita

Videá

Prehrať všetko

Online kurz Data Science a Jazyk R - Jemný Úvod do...

7 zhliadnutí • pred 1 hodinou



Online kurz Python - Ako programuje Google v Pytho...

51 zhliadnutí • pred 1 dňom

Online Kurz Microsoft Excel Grafy a Typy Grafov

61 zhliadnutí • pred 2 dňami

Online Kurz Počítačové Siete - Protokol HTTP(s), Stavové...

166 zhliadnutí • pred 3 dňami

Online Kurz Efektívna Komunikácia - Názory a...

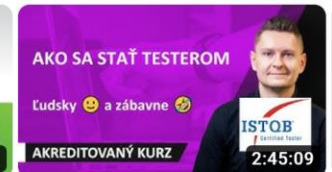
60 zhliadnutí • pred 3 dňami

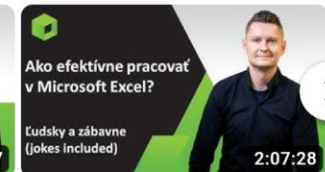
Online kurz Microsoft Outlook - Ako na...

142 zhliadnutí • pred 5 dňami

Obľúbené videá

Prehrať všetko





Dajte odber na IT Academy

➡ WWW.YOUTUBE.COM/C/IT-ACADEMYSK ⬅

IT ACADEMY