13. ročník
Žilina 1. – 3. 7. 2025
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.fri.uniza.sk
OSS Conf

online OSSConf 2025: 1–13

PYTHON AKO PRIATEĽ LaTeX-U?

ALEŠ KOZUBÍK (SK)

Abstrakt. Dôležitým prvkom sadzby odborného textu je vkladanie zdrojového kódu do dokumentu a možnosťou jeho spustenia a zobrazenia výstupu. Tento príspevok v krátkosti predstavuje možnosti balíčka `pythontex`. Tento článok je rozšírenou verziou príspevku prezentovaného na konferencii. Autor ilustruje možnosti spolupráce systému LaTeX s jazykom Python. Upozorňuje tiež na niektoré ťažkosti súvisiace so špecifikami jazyka, použitého v dokumente.

Kľúčové slová. LaTeX, Python, sadzba, zdrojový kód.

PYTHON AS A FRIEND OF LaTeX?

Abstract. An important element of typesetting a professional text is inserting source code into the document and the possibility of running it and displaying its outcome. This contribution briefly presents the capabilities of the `pythontex` package. The article is an extended version of a paper presented at the conference. The author illustrates the cooperation of the LaTeX system with the Python language. It also highlights some difficulties related to the specifics of the language used in the document.

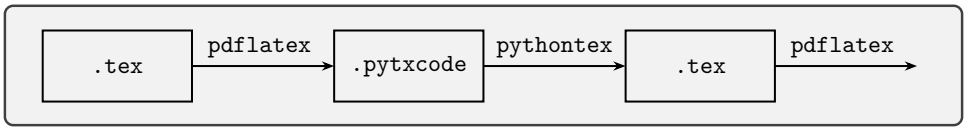
Keywords. LaTeX, Python, typesetting, source code.

Úvod

Dôležitým aspektom tvorby dokumentov je možnosť vkladania zdrojového kódu v nejakom programovacom jazyku s možnosťou jeho spustenia v dokumente a zobrazením výsledku. Vzhľadom na narastajúcu popularitu programovania v jazyku Python je prirodzené pátrať po možnosti spolupráce dokumentu sádzaného v LaTeX-u a programovania v Pythone. Odpoveďou je balíček `pythontex`, ktorého autorom je Geoffrey Poore, ktorý má na svedomí tiež balíčky `fvextra`, ktorý rozširuje `fancyvrb`, `latex2pydata`, a najmä znovuoživenie balíčka `minted`, umožňujúceho syntax highlight zdrojových kódov v rôznych jazykoch s využitím knižnice `Pygments`. Tento balíček sme si stručne predstavili počas konferencie.

Vzhľadom na obmedzený rozsah príspevkov v tlačenej verzii zborníka tu prinášame širšiu verziu, ktorá obsahuje viaceré ukážky a námety spoluráce skriptov v jazyku Python a systému LaTeX. Tiež podrobnejšie upozorníme na niektoré potiaže pri sadzbe dokumentu v slovenskom príp. českom jazyku, ktoré majú svoje špecifiká pri označovaní niektorých elementárnych funkcií. Ukazuje sa, že nie všetky typografické zásady, najmä pokiaľ sa týka sadzby matematiky, sú v balíčku plne akceptované.

Balíček vkladáme do preambuly dokumentu obvyklým spôsobom, použitím príkazu `\usepackage{pythontex}`. Samotné spracovanie dokumentu s použitím tohto balíčka je pomerne jednoduché a prebieha v troch krokoch podľa nasledujúcej schémy.



To znamená, že zdrojový súbor dokumentu s názvom `meno.tex` preložíme pomocou `pdflatex`, pričom okrem obvyklých pomocných súborov sa nám v aktuálnom pracovnom adresári objaví súbor `meno.pytxcode`. Tento súbor sa následne spracuje príkazom `pythontex meno` a ak spracovanie prebehne bez chybových hlášok, ešte raz súbor skompilujeme pomocou `pdflatex`. Za zmienku stojí aj to, že pokiaľ v dokumente nedôjde ku zmene pythonovského kódu, tak pri opakovanej kompilácii stačí len kompilácia pomocou `pdflatex` a procedúru s utilitou `pythontex` nie je potrebné opakovať. Dodajme, že po ukončení podpory Pythonu 2 je niekedy pri použití Pythonu 3 potrebné používať príkaz `pythontex3`. Týka sa to najmä prípadov, kedy udržiavate vo svojom počítači viaceré staršie verzie interpretu jazyka Python a nemáte nastavený `alias` na novšie verzie Pythonu.

1. Prehľad príkazov a prostredí

Balíček `pythontex` poskytuje 7 príkazov, ktoré je možné použiť v rámci bežného textu v dokumentoch LaTeX-u:

- `\py` vyhodnocuje pythonovský výraz a do textu vysádže jeho hodnotu. Napríklad príkaz `\py{2**6-15}` vloží do dokumentu hodnotu 49.
- `\pyc` vykoná pythonovský kód, ale výsledok sa zapíše do `stdout` a do textu sa hodnota nevysádže. Pre zobrazenie výsledku je treba použiť funkciu `print`. Napríklad `\pyc{print(2**10-1000)}` dáva ako výsledok hodnotu 24 .
- `\pyb` vykoná pythonovský kód, ale do textu sa vysádže samotný kód. Napríklad `\pyb{2**10-1000}` dáva v texte výsledok `2**10-1000`. Tu nepomôže ani funkcia `print`, lebo vo výsledku ako výstup dostaneme pythonovský kód `print(2**10-1000)`. Pre vytlačenie výslednej hodnoty je potrebné použiť príkazy `\printpythontex` alebo `\stdoutpythontex`. Potom skutočne zobrazíme výslednú hodnotu 24 alebo 24 .
- `\pys` podporuje premenné a substitúciu výrazov.
- `\pyv` umožňuje tlač pythonovského kódu s vyznačenou syntaxou highlight, ale samotný kód sa nevykonáva. Ak do nášho dokumentu vložíme napríklad príkaz

`\pyv{print(2**10+1000)}`, tak sa zobrazí `print(2**10+1000)`. Avšak tento kód jazyka Python sa nevyková a príkaz `\stdoutpythontex` zobrazí výstup **?? PythonTeX ??**.

– `\pygment` všeobecne pre tlač zdrojového kódu.

– `\pycon` sprístupňuje premenné z predchádzajúceho bloku `pyconsole`.

Okrem týchto príkazov je k dispozícii 7 prostredí:

– `pycode` kód v tomto prostredí sa vykoná, ale nevysádže.

– `pyblock` kód sa vykoná, ale zobrazí sa len zdrojový kód so syntaxhighlight. Pre zobrazenie výstupu treba použiť príkaz `\printpythontex`.

– `pysob` pre substitúcia výrazov a premenných.

– `pyverbatim` pre sadzbu zdrojového kódu, ale bez jeho spustenia. Rovnaké ako `\begin{pygments}{python}...\end{pygments}`.

– `pygments` všeobecne pre sadzbu zdrojového kódu.

– `pyconsole` simuluje interaktívnu konzolu Python-u, t. j. vykonáva príkazy nasledujúce za `>>>`.

– `pythontexcustomcode` kód sa vykoná rovnako ako v `pycode`, ale tlač nie je zobrazená.

2. Vykonanie pythonovského kódu v `pycode` príkazov v `py` a `pyc`

Výpočty je možné vykonávať buď na riadku pomocou makra `\py` alebo v blokovom prostredí ako sú `pycode` alebo `pyblock`.

2.1. Výpočet v riadku s makrom `\py`

Príkaz `\py` odošle vložený kód do interpretu jazyka Python, ten vráti reťazec obsahujúci výsledok spracovania kódu. Tak napríklad príkaz `\py{2**4+2}` vloží do riadku výslednú hodnotu 18. Pripomeňme, že výsledok sa zobrazí až po kompletnej procedúre podľa uvedenej schémy. Po prvej kompilácii sa namiesto výsledku objaví symbol dvoch otáznikov „??“. Príkaz `\py` neumožňuje priradenie hodnôt do premenných, takže napríklad zápis `\py{a=2}` je neplatný. Príčinou je fakt, že priradenie nemožno previesť na reťazec.

Podobne ako `\py` pracuje aj makro `\pyc`, avšak rozdiel je v tom, že len interpretuje kód a pre zobrazenie výsledku musíme použiť funkciu `print`. V tomto prípade teda musíme písať `\pyc{print(2+2)}` aby sme dostali 4. Príkaz `\pyc` však môžeme využiť na priradenie hodnoty do premennej. Rovnaký výsledok by sme teda dosiahli sekvenciou príkazov `\pyc{a=2+2}\py{a}`. Zobrazíme tak rovnaký výsledok 4. Pre úplnosť dodajme, že ak použijeme funkciu `print` v príkaze`\py`, tak dostaneme výsledok 4 None. Zobrazenie `None` je dané tým, že `\py` chce vyhodnotiť výraz, ktorý v tomto prípade nie je zadaný.

Pri použití makra `\pyc` je potrebné venovať pozornosť výslednému výstupu. Ak totiž na konci vety vložíme `\pyc{print(2**100)}`, tak ako výsledok dostaneme hodnotu 1267650600228229401496703205376 . Tu si všimnime medzeru medzi výsledkom a bodkou na konci vety, ktorá je na tomto mieste ponechaná úmyselne, pre zobrazenie chyby v sadzbe. Pre odstránenie tohto nedostatku sa ponúkajú dve riešenie. Buď za výpočet vložíme dve záporné medzery `\!\!` alebo problém vyriešime priamo v pythonovskom kóde použitím reťazca `\endinput`. Do textu teda vložíme `print(str(2**100) + r'\endinput')`. Tak dostaneme výsledok s bodkou bezprostredne za vypočítanou hodnotou 1267650600228229401496703205376.

2.2. Zobrazenie kódu jazyka Python pomocou `\pyb`

Ak chceme namiesto spustenia a vyhodnotenia kódu dosiahnuť jeho zobrazenie, a to aj s farebným zvýraznením syntaxe, použijeme príkaz `\pyb`. Tak môžeme vložiť do riadku napríklad kód `print(2+2)`. Tento príkaz funguje presne naopak, ako príkaz `\pyc`. Kód sa síce vykoná, nezobrazí sa však výsledok ale samotný kód. Môžeme tak napríklad definovať dve hodnoty premenných `a=10;b=3` a následne tieto hodnoty využiť v príkaze `\py`. Takže po vložení `\py{a**b}` dostaneme v dokumente výsledok 1000.

2.3. Vykonanie jednoduchého kódu v prostredí `pycode`

Ako už bolo spomenuté, prostredie `pycode` slúži na vykonanie pythonovského kódu, bez jeho zobrazenia v dokumente. Ak teda do prostredia `pycode` vložíme nasledujúci zdrojový kód:

```
print(r'\begin{center}')
print(r'\textit{Odkaz od jazyka Python!} ')
print(r'\end{center}')
print("[pycode] Buď pozdravený, \LaTeX{}! ")
```

tak ako výsledok dostaneme tento výstup:

Odkaz od jazyka Python!
[pycode] Buď pozdravený, LaTeX!

Významnou vlastnosťou prostredia `pycode` je, podobne ako pri príkaze `\pyb`, že premenné, ktoré boli definované v rámci tohto prostredia môžeme používať aj ďalej v dokumente, mimo tohto prostredia. Ak napríklad v prostredí `pycode` zavedieme dve premenné:

```
a=2; b=3
```

tak v dokumente môžeme príkazom `\py{a+b}` vložiť hodnotu 5.

2.4. Vykonanie kódu na pythonovskej konzole

Niekedy potrebujeme do dokumentu vložiť ukážku výstupu priamo na konzole prostredia Pythonu. Správanie interaktívnej pythonovskej konzoly emuluje prostredie `pyconsole`. Ak vložíme napríklad nasledujúci jednoduchý kód v jazyku Python:

```
\begin{pyconsole}
x=5
y=3
u=x**y
v=y**x
u;v
u+v
\end{pyconsole}
```

tak ako výstup v dokumente dostávame zobrazenie priebehu výpočtu na interaktívnej pythonovskej konzole:

```
>>> x=5
>>> y=3
>>> u=x**y
>>> v=y**x
>>> u;v
125
243
>>> u+v
368
```

Poznamenajme, že premenné definované v prostredí `pyconsole` sú dostupné aj mimo tohto prostredia. Môžeme tak vypočítať napríklad súčet `u+v` pomocou príkazu `\pycon{u+v}`. Dostaneme výsledok 368.

Tiež si všimnime, že farba výsledkov, zobrazených na konzole je štandardne nastavená ako šedá (grey). To je možné predefinovať v preambule pridaním riadku:

```
\setpygmentspygopt[console]{style=bw}
```

Toto nastavenie ale nebude mať vplyv na farebnosť výstupu príkazu `pycon`.

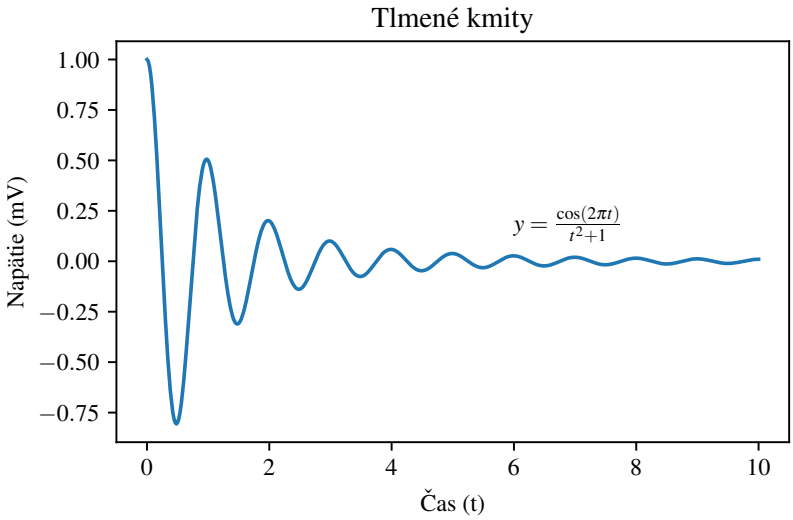
3. Grafy funkcií s knižnicou `matplotlib`

Užitočnou funkcionalitou je možnosť vkladania grafov funkcií, vytvorených pomocou knižnice `matplotlib` do obrázkov v dokumente. Pre tieto potreby si musíme načítať aj

knižnicu `numpy`, čo asi skúsených pythonistov neprekvapuje. Príslušný kód, ktorý vložíme do prostredia `pycode` potom bude vyzerať takto:

```
from matplotlib.pyplot import *
from numpy import *
def f(t):
     return cos(2*pi*t)/(t**2+1)
t=linspace(0,10,500)
y=f(t)
clf()
figure(figsize=(5,3))
rc("text", usetex=True)
plot(t,y)
title(r'Tlmene kmity')
text(6,0.15,r"$y=\frac{\cos (2 \pi t)}{t^2+1}$")
xlabel("Cas (t)")
ylabel("Napatie (mV)")
savefig("tlmene.pdf",bbox_inches="tight")
print(r"\begin{figure}")
print(r"\includegraphics[width=0.8\textwidth]{tlmene.pdf}")
print(r"\vspace{-1.5em}")
print(r"\caption{Graf funkcie z Python kódu.}\label{obr:graf}")
print(r"\end{figure}")
```

Výsledok vidíme na obrázku 1.



Obr. 1. Graf funkcie z Python kódu.

4. Symbolické výpočty s knižnicou sympy

Symbolické výpočty sú neoddeliteľnou súčasťou sadzby matematických textov. Tu s výhodou využijeme knižnicu `sympy` jazyka Python. Na tomto mieste ilustrujeme len skrátenú verziu tabuľky integrálov a derivácií elementárnych funkcií:

$$\frac{d}{dx}\sin(x) = \cos(x) \qquad \int \sin(x)\,dx = -\cos(x)$$

$$\frac{d}{dx}\cos(x) = -\sin(x) \qquad \int \cos(x)\,dx = \sin(x)$$

$$\frac{d}{dx}\operatorname{tg}(x) = \operatorname{tg}^2(x) + 1 \qquad \int \operatorname{tg}(x)\,dx = -\ln(\cos(x))$$

$$\frac{d}{dx}\operatorname{asin}(x) = \frac{1}{\sqrt{1-x^2}} \qquad \int \operatorname{asin}(x)\,dx = x\operatorname{asin}(x) + \sqrt{1-x^2}$$

$$\frac{d}{dx}\operatorname{atan}(x) = \frac{1}{x^2+1} \qquad \int \operatorname{atan}(x)\,dx = x\operatorname{atan}(x) - \frac{\ln(x^2+1)}{2}$$

$$\frac{d}{dx}\sinh(x) = \cosh(x) \qquad \int \sinh(x)\,dx = \cosh(x)$$

$$\frac{d}{dx}\ln(x) = \frac{1}{x} \qquad \int \ln(x)\,dx = x\ln(x) - x$$

Do dokumentu, opäť do prostredia `pycode`, vložíme nasledujúci kód:

```
from sympy import *
var('x')

# Definujeme zoznam funkcií do tabuľky
funcs=['sin(x)','cos(x)','tan(x)','asin(x)','atan(x)','sinh(x)','log(x)']

print(r'\begin{alignat*}{3}')
for func in funcs:
     myderiv='Derivative('+func+',x)'
     myint='Integral('+func+',x)'
     print(r'&'+latex(eval(myderiv))+'='+
             latex(eval(myderiv+'.doit()'))+r'&\hspace{3em}')
     print(r'&'+latex(eval(myint))+'='+
             latex(eval(myint+'.doit()'))+ r'\\[.75em]')
print(r'\end{alignat*}')
```

Všimnime si určitú disproporciu medzi zaužívaným označením niektorých funkcií v matematickej symbolike, kde napríklad funkciu tangens označujeme v slovenčine alebo češtine (ale aj v rozšírenejších svetových jazykoch ako sú francúzština, španielčina alebo zastaralo nemčina) ako $\mathrm{tg}\,x$, kým syntax jazyka Python vyžaduje `tan(x)`, čo sa prejaví aj vo výstupe matematických vzťahov. V anglosaskej symbolike matematické označenie zodpovedá implementovaným funkciám, t. j. tan. Túto nezrovnalosť v symbolike môžeme odstrániť predefinovaním pomocou `\renewcommand\tan{\textrm{tg}}`, resp. pomocou `\renewcommand\tan{\mathrm{tg}}` v preambule LaTeX-ovského súboru. Pri zápise je vhodné pred funkčnú hodnotu vložiť nejakú medzeru (najlepšie zúženú), napr. `\tan\,x`. Vhodnejšie je použiť príkaz `\renewcommand\tan{\mathop{\rm tg}\nolimits}` a potom funguje štandardne `\tan{x}`. Takto sme aj v tomto článku docielili adekvátne označenie funkcie tangens alebo pretypovanie prirodzeného logaritmu z pôvodného log na zaužívané ln. Pri cyklometrických funkciách už toto pretypovanie na našu symboliku také jednoduché nie je. Súvisí to so spôsobom, akým sa preberá výstup výpočtov.

Problémom je postup, akým PythonTeX spracúva pyhonovské kódy v dokumente. V pracovnom adresári vzniká podadresár `pythontex-files-meno`, kde `meno` je názov zdrojového dokumentu. Tu ku každému pythonovskému kódu v dokumente vzniká pomocný súbor `py_default_default_cislo` s príponou `stdout`, kde sa čísla postupne zvyšujú. Kým pre tangens a prirodzený logaritmus sa v týchto súboroch používa `\tan` resp. `\log`, tak pre cyklometrické funkcie je použitý príkaz `\operatorname`. Zo znamená príkazy `\operatorname{asin}`, `\operatorname{acos}` a `\operatorname{atan}`, ktoré už nie je možné pretypovať jednoduchým použitím `\renewcommand`. Neostáva teda iné, ako manuálne prehľadať všetky pomocné súbory resp. spracovať si nejaký skript, ktorý rozparsuje a prepíše obsah týchto pomocných súborov.

Ďalším nedostatkom vo výstupe je zápis symbolu pre diferenciál, ktorý by mal byť správne sádzaný vzpriameným rezom písma, teda $\mathrm{d}x$ a nie dx. Za zmienku stojí aj to, že zátvorky pri jednoduchých argumentoch funkcií strácajú z matematického hľadiska zmysel.

Knižnica `SymPy` obsahuje aj užitočnú funkciu `latex()`, ktorú oceníme najmä pri automatizácii inak zdĺhavých úloh. Tak namiesto manuálneho zapisovania jednotlivých krokov postupu integrácie, alebo kopírovania postupných výpočtov z externého programu (napríklad OSS systému počítačovej algebry `wxmaxima`), ich používateľ môže automaticky generovať priamo v LaTeX-u. Ilustrujeme si to na výpočte dvojného integrálu

$$\iint x \sin{(x+y)}\,\mathrm{d}x\,\mathrm{d}y.$$

Namiesto dlhého manuálneho prepisu integrácie, kde v prvom kroku integrujeme metódou per partes podľa premennej x a v druhom kroku vykonáme integráciu podľa premennej y, vložíme do dokumentu jednoduchý pythonovský kód.

V našom prípade môže mať príslušný kód nasledujúci tvar:

```
\begin{sympycode}
x, y = symbols('x, y')
f = x * sin(x+y)
krok1 = Integral(f, x, y)
krok2 = Integral(Integral(f, x).doit(), y)
krok3 = krok2.doit()
\end{sympycode}

\begin{align*}
\sympy{krok1} &= \sympy{krok2} \\ &= \sympy{krok3}
\end{align*}
```

Dostaneme výpočet dvojného integrálu v tvare:

$$\begin{aligned}\iint x\sin\left(x+y\right)dx\,dy &= \int\left(-x\cos\left(x+y\right)+\sin\left(x+y\right)\right)\,dy\\ &= -x\sin\left(x+y\right)-\cos\left(x+y\right)\end{aligned}$$

Takisto riešenie rovníc môžeme zautomatizovať pomocou funkcie `solve` z knižnice `sympy`. Do prostredia `sympycode` vložme napríklad nasledujúci kód:

```
x = symbols('x')
myeq = Eq(3*x**3+4*x**2+2*x-4,0)
print('Korene rovnice ')
print(latex(myeq, mode='inline'))
print(' sú')
print(latex(solve(myeq), mode='inline'))
```

Ako výstup dostaneme vyriešenú rovnicu.

Korene rovnice $3x^3+4x^2+2x-4=0$ sú $\left[2/3,\ -1-i,\ -1+i\right]$.

Aj tu sa stretáme s určitou typografickou nezrovnalosťou. Imaginárna jednotka by mala byť správne sádzaná vzpriameným rezom písma, čo nie je dodržané. Tentoraz však ide o problém samotnej knižnice `SymPy`. Ak by sme nepoužili funkciu `latex()`, dostali by sme výsledné riešenie v tvare [2/3, -1 - I, -1 + I]. Ako vidíme, imaginárna jednotka je nesprávne označená verzálkou `I`. Ak použijeme funkciu `latex()`, táto len transformuje verzálku na minuskulu, avšak nie vo vzpriamenom reze.

Nie všetky algebraické rovnice však dokážeme vyriešiť analyticky s explicitným vyjadrením koreňov. Neraz musíme použiť numerické metódy riešenia. To prináša ďalšie nezrovnalosti v označovaní imaginárnej jednotky. Numerický výpočet vykonáme v prostredí `pylabcode`. V prípade numerického riešenia tej istej rovnice rovnice s knižnicou `NumPy` dostaneme výsledok v nasledujúcej podobe:

Korene rovnice $3x^3 + 4x^2 + 2x - 4 = 0$ sú:

$$x_1 = (-1.0000000000000002 + 1.0000000000000002j),$$
$$x_2 = (0.6666666666666664 + 0j),$$
$$x_3 = (-1.0000000000000002 - 1.0000000000000002j).$$

Ako vidíme, tu sa pre označenie imaginárnej jednotky zachovalo označenie j, bežne používané v rámci základnej syntaxe jazyka Python, na rozdiel od matematiky, kde je používané i. Ak to zhrnieme, máme tu ako výsledok tri rôzne varianty označenia imaginárnej jednotky, čo je istá nekonzistentnosť. Tieto ťažkosti však vyplývajú zo samotnej implementácie uvedených pythonovských knižníc, nie sú záležitosťou PythonTEX-u.

5. Výpis zdrojového kódu

Okrem už skôr prezentovaných výpisov zdrojového kódu, kde sme využili možnosti balíčka `tcolorbox`, prináša PythonTEX aj vlastné prostredie pre zobrazovanie zdrojového kódu. Jeho vVýhodou je, že môžeme doplniť čísla riadkov a v texte dokumentu sa potom na tieto číslované riadky odvolávať. Náš kód pre tabuľku derivácií a integrálov potom môžeme prezentovať v prostredí `sympyblock` takto:

Kód generujúci tabuľku derivácií a integrálov

```
from sympy import *

var('x')

# Definujeme zoznam funkcií do tabuľky
funcs=['sin(x)','cos(x)','tan(x)','asin(x)','atan(x)',
       'sinh(x)','log(x)']

print(r'\begin{alignat*}{3}')
for func in funcs:
     myderiv='Derivative('+func+',x)'
     myint='Integral('+func+',x)'
     print(r'&'+latex(eval(myderiv))+'='+
             latex(eval(myderiv+'.doit()'))+r'&\hspace{3em}')
     print(r'&'+latex(eval(myint))+'='+
             latex(eval(myint+'.doit()'))+ r'\\')
print(r'\end{alignat*}')
```

Ako vidíme z výsledného zápisu kódu, toto prostredie pre listingy má okrem číslovania niekoľko ďalších užitočných vlastností. V prvom rade je to schopnosť rozdelenia orámovania a vloženého kódu na zlome stránky, čo značne uľahčuje umiestnenie výpisu do dokumentu. Prostredie automaticky farebne zvýrazňuje syntax jazyka. Ako voliteľné

parametre prostredia je možné nastaviť parametre číslovania, štýl orámovania a do záhlavia rámu umiestniť nadpis. V našom prípade teda zápis prostredia aj s voliteľnými parametrami vyzeral takto:

```
\begin{sympyblock}[][numbers=left,frame=single,framesep=5mm,
        label=Kód generujúci tabuľku derivácií a integrálov]
```

Pozornému čitateľovi isto neuniklo, že prostredie má dva voliteľné argumenty. To je všeobecnou vlastnosťou všetkých príkazov a prostredí PythonTeX-u. Prvý voliteľný argument určuje reláciu, v ktorej sa kód vykonáva. Štandardne sa celý kód a obsah blokov vykonáva v rámci jednej relácie Pythonu a celý obsah konzoly sa vykonáva v rámci samostatnej relácie. V mnohých prípadoch je takéto správanie žiaduce kvôli kontinuite, ktorú poskytuje. Niekedy však môže byť užitočné izolovať nejaký nezávislý kód v samostatnej relácii. Napríklad dlhý výpočet by sa mohol umiestniť do vlastnej relácie, takže by sa spustil iba po úprave jeho kódu, nezávisle od iného kódu. PythonTeX poskytuje takúto funkcionalitu prostredníctvom používateľom definovaných relácií. Všetky príkazy a prostredia akceptujú názov relácie ako voliteľný argument.

Všetky prostredia PythonTeX-u akceptujú aj druhý voliteľný argument. Ten pozostáva z nastavení pre LaTeX-ový balíček `fancyvrb` (Fancy Verbatims), ktorý PythonTeX používa na sadzbu kódu. Tieto nastavenia umožňujú prispôsobenie vzhľadu kódu. Ako sme ilustrovali v našej ukážke, blok kódu môže byť napríklad ohraničený farebným rámčekom s názvom a zároveň môžu byť zahrnuté čísla riadkov. Viac sa o balíčku `fancyvrb` možno dozvedieť v manuálovej dokumentácii [10].

6. Dynamická sadzba tabuliek

Balíček PythonTeX s výhodou použijeme pri dynamickej sadzbe tabuliek, kde obsah jednotlivých buniek je viazaný na nejaké výpočty. V prepojení s príkazom cyklu jazyka Python môžeme zostaviť napríklad tabuľku mocnín prirodzených čísiel. Príslušný pythonovský kód pre sadzbu tabuľky môžeme zobraziť pomocou prostredia `pyblock` v nasledujúcej podobe:

```
zaciatok,koniec=1,8
print(r'\renewcommand\arraystretch{1.3}')
print(r'\begin{tabular}{|*{6}{r|}}')
print(r'\hline')
print(r'$n$ &$n^2$ &$n^3$ &$n^4$ &$n^5$ &$n^6$ \\ \hline')
for n in range (zaciatok,koniec+1):
    print(n,'&',n**2,'&',n**3,'&',n**4,'&',n**5,'&',n**6,r'\\')
print(r'\hline')
print(r'\end{tabular}')
```

Výslednú tabuľku mocnín potom do nášho dokumentu vložíme pomocou príkazu `\printpythontex`.

n	n^2	n^3	n^4	n^5	n^6
1	1	1	1	1	1
2	4	8	16	32	64
3	9	27	81	243	729
4	16	64	256	1024	4096
5	25	125	625	3125	15625
6	36	216	1296	7776	46656
7	49	343	2401	16807	117649
8	64	512	4096	32768	262144

7. Záver

Aj keď balíček `pythontex` prináša pri sadzbe v slovenskom alebo českom jazyku určité komplikácie pri finalizácii výstupu, je možné ho hodnotiť ako veľkú pomoc pri vytváraní dokumentov obsahujúcich väčšie množstvo výpočtov. Navyše, pri publikovaní v angličtine, čo sa stalo pri vedeckej práci štandardom je takmer bezproblémové. Pretože ide o kombináciu dvoch nástrojov, začiatočníkom pri vytváraní dokumentov v systéme LaTeX je možné odporučiť učebnice [8] alebo [5]. O čosi podrobnejšie zoznámenie so systémom ponúka [4] od autora typografického systému TeX, v [2] alebo [6].

Pre záujemcov neznalých jazyka Python (ako aj samotný autor príspevku sa v tomto smere hodnotí ako večný začiatočník) možno odporučiť [9] alebo [3]. Niektoré ďalšie nápady a inšpirácie pre samotný balíček `pythontex` potom ponúka [1] a podrobný popis manuál k balíčku [7].

Veľkou výhodou tohto balíčka je, že spolupracuje aj s inými jazykmi, nielen s Pythonom, ako by mohol evokovať jeho názov. Ako sa uvádza v manuáli [7], voliteľným argumentom `usefamily` je možné definovať prostredie rozličných programovacích jazykov. Od verzie 0.12 podporuje pythontex jazyky Ruby a Julia. Od verzie 0.13 pribudlo rozšírenie o Octave. Verzia 0.15 potom priniesla ďalšie rozšírenie o Burn shell `bash` a Rust. Konečne od verzie 0.17 je dostupné rozšírenie pre jazyky R, Perl, resp. Perl 6 a JavaScript.

Poďakovanie. Príspevok vznikol s prispením grantu KEGA 019ŽU-4/2023 „Inovatívne učenie matematiky s podporou Open Source“ podporeného Slovenskou kultúrno-edukačnou grantovou agentúrou.

Literatúra

[1] ESSLINGER, B. & STUDENTS: *PythonTeX and LaTeX – Familiarize us with PythonTeX in order to build the CrypTool Book*, online dokument `https://www.cryptool.org/download/ctb/PythonTex-by-Examples.pdf`.

[2] GRÄTZER, G.: *Text and Math Into LaTeX*, 6th ed., Springer Nature Switzerland AG, Cham, 2024, ISBN 978-3-031-55280-9, e-book: `https://doi.org/10.1007/978-3-031-55281-6`.

[3] HUNT, J.: *A Beginners Guide to Python 3 Programming*, 2nd ed., Springer Nature Switzerland AG, Cham, 2023, ISBN 978-3-031-35121-1, e-book: https://doi.org/10.1007/978-3-031-35122-8.

[4] KNUTH, D. E.: *The TeXbook*, Volume A of *Computers and Typesetting*, Addison-Wesley Publishing Company (1984), ISBN 0-201-13448-9.

[5] KOPKA, H. – DALY, P. W.: *LaTeX – Podrobný průvodce*, Brno, Computer Press, 2004, ISBN 80-722-6973-9.

[6] ÖCHSNER, M. – ÖCHSNER, A.: *Advanced LaTeX in Academia*, Springer Nature Switzerland AG, Cham, 2021, ISBN 978-3-030-88955-5, e-book: https://doi.org/10.1007/978-3-030-88956-2.

[7] POORE, G. M.: *The pythontex package*, dokumentácia balíčka PythonTeX, dostupné online http://mirrors.ctan.org/macros/latex/contrib/pythontex/pythontex.pdf.

[8] RYBIČKA, J.: *LaTeX pro začátečníky*, Brno, KONVOJ 2003, ISBN 80-7302-049-1.

[9] SCHÄFER,CH.: *Quickstart Python*. Springer essentials, Wiesbaden, Germany, 2019, ISBN 978-3-658-33551-9, e-book: https://doi.org/10.1007/978-3-658-33552-6

[10] Van ZANDT, T. – GIROU, D. – RAHTZ, S. and VOSS, H.: *The fancyvrb package: Fancy Verbatims in LaTeX*, http://www.ctan.org/pkg/fancyvrb.

Kontaktná adresa

RNDr. Aleš Kozubík, PhD., Katedra matematických metód, Fakulta riadenia a informatiky, Žilinská univerzita, Univerzitná 8215/1, 010 26 Žilina, Slovenská republika,
E-mailová adresa: alesko@frcatel.fri.uniza.sk