Umíme informatiku

Programování v Pythonu pro SŠ

Metodické poznámky pro učitele

Tento materiál podává přehled možných způsobů zapojení cvičení v systému Umíme informatiku při výuce programování na středních školách.

Pojetí výuky programování v Pythonu

Naším cílem není poskytnout metodiku či materiály pro „univerzálně správný“ způsob výuky programování. Ten totiž neexistuje. Vhodné pojetí výuky závisí na okolnostech: vstupní znalosti žáků, jejich motivace a ambice, časová dotace výuky. Dále popisujeme tři základní možnosti pojetí výuky, přičemž pochopitelně existuje mnoho dalších smysluplných variací. Pro každou ze zmíněných variant zmiňujeme, jakým způsobem lze Umíme informatiku při výuce využít.

Ochutnávka programování

Charakteristiky pojetí:

- Důraz na obecné principy algoritmického myšlení, minimální důraz na specifika konkrétního programovacího jazyka.
- Typicky je provázané s blokovým programováním. Kromě návaznosti na bloky lze využívat i integrovaný styl, tj. střídání blokového a textového programování.
- Cílem je základní vyzkoušení „opravdového programovacího jazyka“, aby si žáci udělali představu, jak programování vypadá a mohli se rozhodnout, zda se mu chtějí více věnovat.
- Smysluplné i s malou časovou dotací.

Možnosti využití Umíme informatiku:

- Výuku lze do velké míry stavět na využití Umíme informatiku, které nabízí blokové programování i Python v provázané podobě, typicky např. u želví grafiky.
- V tomto přístupu lze vystačit i s programováním v prohlížeči, jak ho nabízí Umíme informatiku.

Pevné základy

Charakteristiky pojetí:

- Důraz na důkladné pochopení základních konceptů při programování v Pythonu (výrazy, cykly, podmínky, funkce).
- Vhodné pro žáky, kteří mají v plánu se programováním více zabývat (studium na VŠ, budoucí profese, vlastní programátorské projekty).
- Procvičování především formou řešení úloh pracujících s čísly a krátkými texty. Méně atraktivní než jiná pojetí, ale pro vážné zájemce o programování je nezbytné si tímto projít.
- Krom zjednodušeného editoru v prohlížeči je vhodné pracovat i v některém z editorů pro Python (např. `IDLE`, `Thonny`, `pyCharm`).

Možnosti využití Umíme informatiku:

- Úvodní seznámení s koncepty probíhá pomocí cvičení v Umíme informatiku, následně žáci tvoří větší programy ve standardním editoru.
- Ve výuce se používá primárně standardní editor, Umíme informatiku slouží jako doplněk pro samostatnou práci (rozcvička, opakování, domácí úloha).

Projektové pojetí

Charakteristiky pojetí:

- Při tomto pojetí se snažíme žáky dovést až k prakticky zajímavému výstupu.
- Tento přístup je atraktivnější a je snadnější žáky motivovat.
- Nevýhodou je povrchnější projití témat. Často je nezbytné alespoň část výuky pojmout „tutoriálovým stylem“, kde žáci jen kopírují kroky či doplňují dílčí části kódu a nedá se očekávat, že plně pochopí všechny použité koncepty.
- Příklady projektů:
  - Vývoj počítačové hry za využití knihovny `PyGame`.
  - Vývoj aplikací s uživatelským rozhraním za využití knihovny `tkinter`.
  - Práce s multimédii (vytváření či úpravy obrazu, zvuku, videa).

Možnosti využití Umíme informatiku:

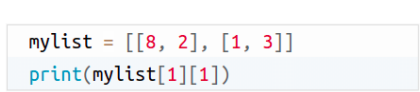
- Dobré zvládnutí konceptů, které jsou pro projekt potřeba.
- Získání základního povědomí o konceptech, které nejsou v projektu dostatečně silně zastoupeny.

Cvičení v Umíme informatiku

Umíme informatiku nabízí několik forem procvičování, přičemž každé z nich má své specifické způsoby využití.

Základní procvičení syntaxe a čtení kódu

Ve cvičeních *Rozhodovačka*, *Psaná odpověď*, *Pexeso* žák vybírá z nabízených možností, případně píše na klávesnici krátkou odpověď (např. jedno číslo). Úkolem je typicky rozhodnout, co udělá zadaný kód. Další otázky se týkají základní syntaxe příkazů a jejich použití. Tato cvičení jsou vhodná především při prvním seznámení s novými programátorskými koncepty.



Cvičení *Porozumění* má stále jednoduché ovládání formou výběru z možností. Zde jsou k dispozici vypracované příklady – funkce v rozsahu několika řádků. Ke každé takové funkci je připravena série několika otázek, které kontrolují pochopení kódu.

Interaktivní cvičení

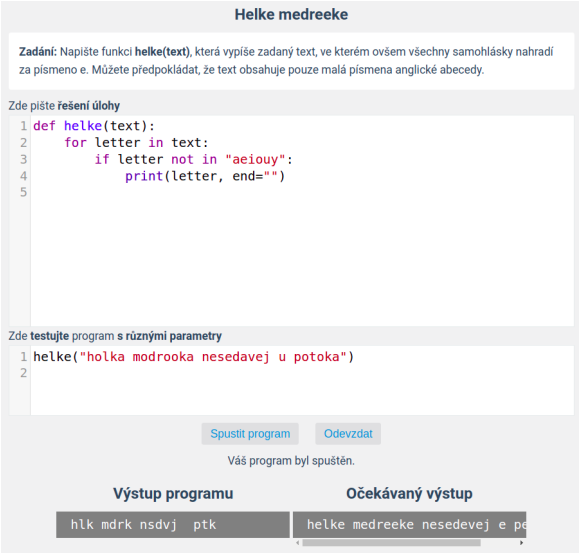
V těchto cvičeních žáci píší vlastní programy, které mohou přímo v prohlížeči spustit a vidět výsledek. Úlohy se automaticky vyhodnocují. Žáci tak dostávají bezprostřední zpětnou vazbu a učitel může snadno vidět, kteří žáci zvládli kterou úlohu vyřešit.

Základní interaktivní programování v Pythonu se zaměřuje na práci s čísly a krátkými texty (řetězci): žáci píší funkce, které pro zadaný vstup produkují požadovaný výstup. Každá úloha má skrytá testovací data, na kterých se vyhodnocuje správnost řešení.

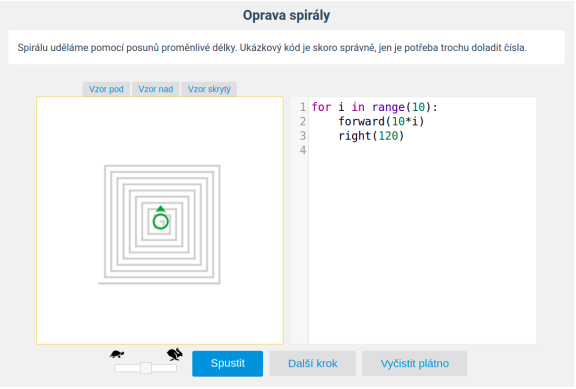
V úloze *Python želva* se kreslí obrázky pomocí principu želví grafiky. V Umíme informatiku je želví grafika dostupná i formou blokového programování a lze ji tak využít k plynulému přechodu od bloků k textovému zápisu.

V obou cvičeních jsou zadání členěna to tematických sad. Úlohy jsou uspořádány podle obtížnosti, úvodní úlohy typicky obsahují předpřipravený kód, který stačí mírně upravit.

Interaktivní programování



Python želva



Úkolovka

Toto cvičení spočívá ve zpracování informací a vyřešení zadaného úkolu (např. určit počet výskytů nejčastějšího písmena v zadaném textu). Jako odpověď se do systému zadává pouze výsledek. Úkoly jsou koncipovány tak, že nejdou řešit „z hlavy“. Známý zdroj tohoto typu úloh je Project Euler (`projecteuler.net`), ten však nabízí poměrně náročné úlohy. Zadání v Umíme informatiku jsou cílená na začátečníky.

Úkoly lze řešit v libovolném programovacím jazyce (nebo třeba i v tabulkovém editoru), jsou však koncipovány tak, aby vedly na procvičení užitečných konceptů v Pythonu.

Vstupní předpoklady

Pro úspěšné zvládnutí programování v Pythonu je vhodné, aby žáci měli dobré základy, na kterých mohou stavět.

Velmi užitečná je zkušenost s **blokovým programováním**, ve kterém je možné si vyzkoušet klíčové koncepty jako podmínky, cykly a proměnné. V blokovém programování odpadají problémy s textovým zápisem a žáci se tak mohou soustředit na samotné pochopení principů. Textově psaný zápis programů nabízí řadu výhod, ale také nástrah. Má smysl se do něj pouštět, až když žáci zvládají bez větších potíží blokové programování. Umíme informatiku nabízí mnoho příkladů na blokové programování se širokou škálou obtížnosti. Pokročilejší příklady v úlohách Želví grafika a Plošinovka nabízí dostatečnou výzvu i pro středoškolské žáky.

Při programování využíváme **základní logické spojky** (negace, a, nebo). Ty sice můžeme přirozeně vyučovat dohromady s využitím podmínek v Pythonu, ale tím se zvyšuje zátěž na žáky. Má smysl základní práci s logickými spojkami procvičit nezávisle na Pythonu, viz např. následující cvičení:

`https://www.umimeinformatiku.cz/cviceni-logika-spojky-tabulky`

Méně zřejmý, ale velmi užitečný vstupní předpoklad je **dělitelnost** (celočíselné dělení, dělení se zbytkem, význam pojmů „být dělitelný“, „prvočíslo“). Mnoho příkladů z úvodního programování nějakým způsobem staví na dělitelnosti. Typický příklad je úkol „výpis prvočísel“, který lze řešit krátkým programem (několika různými způsoby) a lze na něm ilustrovat a procvičit mnoho základních konceptů. I v mnoha programech, které na první pohled s dělitelností nesouvisí, se používá operace dělení se zbytkem (např. různé šifry nebo výpisy řetězců). Pokud žáci nemají v dělitelnosti jasno, snadno se pak při programování na těchto věcech zaseknou. Příklady k rychlému procvičení nebo zopakování dělitelnosti lze najít na Umíme matiku.

Přehled konceptů

Výuka programování je komplikovaná mimo jiné tím, že programování vyžaduje použití více vzájemně provázaných konceptů. Ať už tyto dílčí koncepty zařadíme do výuky v libovolném pořadí, nikdy to není úplně ideální. Navíc samotná volba konceptů a vhodná hloubka jejich pokrytí je diskutabilní.

Procvičování v rámci Umíme informatiku je organizováno do poměrně typické struktury základních konceptů. Pro každý koncept je na webu uveden stručný přehled hlavních principů a konkrétní příklady v Pythonu. Tyto soupisy neslouží jako samonosný výukový materiál, poskytují ale užitečný stručný přehled a připomenutí nejdůležitějších bodů. S připravenými podklady lze pracovat poměrně flexibilně podle potřeb zvoleného stylu výuky.

Proměnné a číselné výrazy

Základní použití proměnných, konstrukce aritmetických výrazů (kombinace konstant a proměnných), výpis pomocí `print`. Základ, který není moc atraktivní, ale je nezbytný.

`https://www.umimeinformatiku.cz/programovani-promenne-a-ciselne-vyrazy`

Logické výrazy

Použití aritmetických operací (rovnost, porovnání) a logických spojek (and, or,

not).

https://www.umimeinformatiku.cz/programovani-logicke-vyrazy

Podmíněný příkaz (if)

Použití podmíněného příkazu (`if`) a jeho různých variant (`if-else`, `if-elif-else`).

https://www.umimeinformatiku.cz/programovani-podmineny-prikaz-if

Cyklus for

S `for` cyklem lze pracovat na několika různých úrovních a může být vhodné postup výuky rozdělit: 1) Základní konstantní počet opakování bloku příkazů, bez využití řídicí proměnné. 2) Využití řídicí proměnné, která nabývá hodnot z intervalu čísel (základní `range`). 3) Obecné využití `for` cyklu (nejen přes intervaly čísel, ale i přes řetězce či seznamy).

https://www.umimeinformatiku.cz/programovani-cyklus-for

Cyklus while

Cyklus s podmínkou typicky dělá větší problémy než cyklus s konstantním počtem opakování, je tedy vhodné mu věnovat samostatnou pozornost.

https://www.umimeinformatiku.cz/programovani-cyklus-while

Vnořené cykly

Použití vnořených cyklů nepřináší žádný nový syntaktický prvek v Pythonu. Po algoritmické stránce však toto téma dělá žákům problémy a je vhodné mu věnovat dostatek pozornosti.

https://www.umimeinformatiku.cz/programovani-vnorene-cykly

Funkce

U tématu funkcí můžeme hned od začátku používat funkce bez návratové hodnoty („procedury“), které jsou poměrně intuitivní, zvyšují přehlednost kódu a usnadňují testování. Plnohodnoté funkce včetně návratové hodnoty již jsou náročnější a vyžadují důkladnější procvičení.

https://www.umimeinformatiku.cz/programovani-funkce

Řetězce

Řetězce sdílí se seznamy společný prvek „indexování“, který není jednoduchý (mj. i kvůli neintuitivnímu indexování od nuly) a je dobré mu věnovat dostatek pozornosti. Řetězce jsou více intuitivní a je tedy vhodnější téma indexování nejprve procvičit na nich.

https://www.umimeinformatiku.cz/programovani-retezce

Seznamy

Seznamy se v Pythonu od řetězců liší měnitelností a nabízí bohatší využití. Specifické podtéma, které si zaslouží speciální pozornost, jsou vnořené seznamy (např. reprezentace 2D mřížky pomocí seznamu seznamů).

https://www.umimeinformatiku.cz/programovani-seznamy

Slovníky

Slovníky jsou velice praktická datová struktura, která je však již na hraně úvodního programování. Není snadné je smysluplně procvičovat na krátkých programech; typické příklady, které to umožňují, jsou „převodník do/z Morseovy abecedy“ a „frekvenční analýza“.

https://www.umimeinformatiku.cz/programovani-slovniky

Pojmenování proměnných a funkcí

Již od úvodní výuky programování je dobré vést žáky k používání smysluplných názvů proměnných a funkcí. Není vhodné odkládat dobrý styl na později („až se naučí základy“). Pokud žáci začnou proměnné a funkce pojmenovávat nevhodným stylem (`a, b, c, fun1, fun2, fun3`), vytvoří si špatné návyky, kterých se pak těžko zbavují. Navíc neschopnost pojmenovat proměnnou je často indikace nedomyšleného postupu. Důsledná volba dobrých názvů usnadňuje samotné programování a ladění.

Doporučujeme pro názvy proměnných a funkcí používat **angličtinu**, čehož se drží i připravené příklady a ukázkové kódy v Umíme informatiku. Toto doporučení má několik důvodů:

- Pokud budou žáci s programováním dále pokračovat a budou pracovat ve větším týmu, budou časem muset přejít na čistou angličtinu. Použití českých názvů je nevhodný návyk, kterého se špatně zbavuje.
- Klíčová slova jazyka Python jsou anglicky. Pokud používáme české názvy proměnných a funkcí, dochází k míchání jazyků, které je trochu matoucí. Často také dochází k tomu, že některé proměnné mají názvy česky, jiné anglicky.
- U českých názvů proměnných není jasné, zda je psát v Pythonu s diakritikou či nikoliv (oboje má své nevýhody).

Použití angličtiny má pochopitelně svoje nevýhody: může žáky brzdit, protože nemají dostatečnou slovní zásobu, příp. dokonce může vést k tomu, že začnou raději používat jednopísmenné názvy proměnných. Je tedy pochopitelně nutné vždy zvážit konkrétní okolnosti a jazykovou vybavenost žáků.

Další zdroje

Zdrojů k úvodní výuce programování v Pythonu existuje velké množství. Zde uvádíme pouze několik ukázek s důrazem na ty, které bezprostředně souvisí s autory Umíme informatiku.

- *Učíme se programovat v jazyce Python 3*, překlad knihy *How to think like a computer scientist – Learning with Python* (Jeffrey Elkner, Allen B. Downey and Chris Meyers).

  `http://howto.py.cz/index.htm`

  Volně dostupný výukový text, jde o překlad jedné z klasických anglicky psaných učebnic.
- *Programování v jazyce Python pro střední školy*, Andrej Blaho, Ľubomír Salanci, Václav Šimandl.

  `https://www.imysleni.cz/ucebnice/zaklady-programovani-v-jazyce-python-pro-stredni-skoly`

  Výukové materiály z projektu `imysleni.cz`, které nabízí detailní metodiku. Kladou však důraz na specifické pojetí výuky (práce s obrázky a grafickým prostředím).
- Slidy k předmětu *Úvod do programování a algoritmizace* na FI MU.

  `https://www.fi.muni.cz/~xpelanek/IB113/`

  Jde o úvodní předmět na VŠ cílený na začátečníky a neinformatiky. Výuka využívá mj. cvičení z Umíme informatiku. Posluchači typicky mají již nějaké zkušenosti ze SŠ a jsou motivovaní, takže předmět má větší tempo, než jaké je reálné

při běžné výuce na SŠ. Dostupné slidy z předmětu ale mohou poskytnout inspiraci.

- *Programátorská cvičebnice*, Radek Pelánek, Computer Press, 2012.
  Sbírka příkladů od jednoho z autorů Umíme informatiku. Obsahuje náměty i na složitější příklady (vhodné jako úkoly třeba pro talentované žáky).
- *Code Quality Defects across Introductory Programming Topics*, Tomáš Effenberger, Radek Pelánek, Technical Symposium on Computer Science Education, 2022.
  Odborná publikace založená na analýze dat z Umíme informatiku. Obsahuje přehled konkrétních nedostatků, které se často vyskytují v odevzdaných řešeních, což může posloužit jako užitečná inspirace pro výuku.

Umíme informatiku
https://www.umimeinformatiku.cz/

Autorský kolektiv:

- Metodika a tvorba zadání pro Python: doc. Mgr. Radek Pelánek, Ph.D.; RNDr. Tomáš Effenberger, Ph.D.
- Návrh a realizace cvičení: Mgr. Petr Jarušek, Ph.D.; Mgr. Tomáš Vejpustek