PROGRAMOVÁNÍ V JAZYCE PYTHON

pro střední školy

Andrej Blaho
Ľubomír Salanci
Václav Šimandl

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Poděkování

Jiří Dvořák, Vladimír Přibyl, Pavel Roubal, Sylva Jarošová, Eva Kospachová, Hana Šandová, Petr Šavelka, Radek Šmíd, Jaroslav Vik, Kateřina Bartošová, Tomáš Dragon, Daniel Hošek, Ondřej Rusek, Eliška Šiponová

Jihočeská univerzita
v Českých Budějovicích
University of South Bohemia
in České Budějovice

PROGRAMOVÁNÍ V JAZYCE PYTHON
pro střední školy

RNDr. Andrej Blaho, PhD.; doc. RNDr. Ľubomír Salanci, PhD.;
Mgr. Václav Šimandl, Ph.D.

Recenzenti:
doc. RNDr. Pavel Töpfer, CSc.; Mgr. Václav Dobiáš, Ph.D.

Vydavatel:
Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta

Obálka:
Mgr. Pavel Pfauser

Rok vydání: 2020

Základní informace o vzdělávacím obsahu

- Učebnice je určena pro žáky 1. nebo 2. ročníku SŠ (kvinta nebo sexta osmiletého gymnázia)
- Učebnice se skládá z materiálů pro žáka a z materiálů pro učitele:
  - **Pracovní listy pro žáky** (zadání úloh, která v elektronické a/nebo tištěné podobě poskytnete žákům)
  - **Metodická příručka** pro učitele
- Rozsah učebnice je 20 lekcí. Každá lekce je věnována jednomu základnímu tématu (viz seznam lekcí níže)
- Každá lekce obsahuje základní a rozšiřující úlohy. Rozšiřující úlohy jsou označeny hvězdičkou a jsou určeny pro žáky, kteří mají chuť nad úlohou déle přemýšlet (jejich vyřešení tedy není nezbytné pro zvládnutí daného učiva)
- K učebnici je přiložena vzorová sada testových úloh

Seznam lekcí:

- Výrazy
- Proměnné
- Program
- Výpisy
- Kreslení
- Barvy
- Kreslení s proměnnými
- Podprogramy
- Náhoda
- Kreslení textu
- Program s opakováním
- Proměnná cyklu
- Výrazy v cyklu
- Elipsy a kruhy
- Kruhy a cykly
- Větvení
- Větvení a konstrukce
- Vnořené větvení
- Podprogram s parametrem
- Kreslení myší

Proč právě Python

Python je moderní programovací jazyk, jehož popularita stále roste. Jeho autorem je Guido van Rossum, který jej vymyslel v roce 1989. Jazyk Python je používán například v systémech firmy Google, v aplikacích YouTube, Dropbox, Mozilla, Quora, Facebook nebo na platformě Rasperry Pi.

Python je vyučován jako úvodní programovací jazyk na mnohých renomovaných univerzitách, jako je například MIT, Carnegie Mellon, Berkeley, Cornell, Caltech, Illinois,…

Mezi jeho výhody patří multiplatformnost, tedy schopnost běžet na různých platformách, mimo jiné pod operačními systémy MS Windows, Linux, nebo MacOS. Za další výhodu lze považovat jeho dostupnost, neboť je to freeware a též open source.

Hlavní vlastnosti jazyka Python:

- Velmi jednoduchá a dobře čitelná syntaxe
- Vzhledem k jeho vysoké interaktivitě je Python velmi vhodný i pro výuku programování
- Na rozdíl od staticky typovaných jazyků, ve kterých je potřeba dopředu deklarovat typy veškerých dat, je Python dynamicky typovaný, což znamená, že neexistují žádné deklarace
- Python obsahuje pokročilé rysy moderních programovacích jazyků, např. podporu práce s datovými strukturami nebo objektově orientované tvorby software
- Python je univerzální programovací jazyk, který poskytuje prostředky pro tvorbu moderních aplikací, jako je analýza dat, zpracování multimédií, síťové aplikace a podobně
- Python má rozsáhlou komunitu programátorů a expertů, kteří jsou ochotní svými radami pomoci i začátečníkům

Filozofie tohoto kurzu

- Kurz vychází z nejmodernějších trendů ve výuce programování, které podporují konstruktivistický přístup k budování vědomostí
- Preferujeme **samostatné objevování** konceptů žáky před výkladem učitele
- Žáci se **učí řešením** připravené sady úloh
- Každý žák pracuje svým **vlastním tempem**. Pokud chce, může si měnit pořadí úloh
- Doporučujeme, aby **žáci** během řešení úloh navzájem **komunikovali**, **radili si**, **vysvětlovali si** své postupy a řešení
- **Učitel** většinou do tohoto procesu **nezasahuje**
    - V rámci některých témat **může učitel shrnout** vybrané postupy, případně **dodatečně vysvětlit** náročnější koncepty
    - Doporučujeme, aby v závěrečné části hodiny učitel společně se žáky shrnul, co se během této hodiny naučili, co jim dělalo problémy
- **Tempo** probírané látky ponecháváme **na učiteli** – některé lekce může rozdělit do více hodin, případně může nějakou část lekce přenést do jiné hodiny
- **Pořadí témat** jsme zvolili na základě náročnosti témat **od jednodušších po složitější**. Jelikož je dle našeho názoru v jazyce Python problematika for cyklů jednodušší než problematika podmíněných příkazů, je řazeno téma cyklů před tématem podmíněných příkazů (zde nazývaných příkazy větvení). Je na učiteli, zda nabídnuté pořadí témat zachová, nebo jej změní podle svých zkušeností

- Protože tento kurz není **určen** jen **pro gymnázia**, ale i pro **střední odborné školy**, má tato učebnice několik **specifik**:
    - Je využívána **matematika** téměř výhradně na **úrovni základní školy**. Autorům kurzu se nezdá vhodné zatížit tento kurz náročnějšími tématy středoškolské matematiky, které by mohly být pro některé žáky úplně zbytečnou komplikací a přitom pro kvalitu výuky programování by neměly žádný přínos.
    - Učivo je žákům předkládáno po **malých krocích** a důraz je kladen na **procvičování** získaných programátorských kompetencí. Pokud se tempo probírané látky zdá učiteli vzhledem k potřebám a programátorským schopnostem dané skupiny žáků pomalé, může žákům doporučit některé procvičovací úlohy vynechat. Musí si však být jist, že daná úloha nezavádí nový koncept či nepoužívá daný koncept jiným způsobem než úlohy předchozí.
    - **Pokročilejší** programátorské **konstrukce** a **vylepšení**, které přímo nesouvisejí s cíli učebnice, jsou uváděny **pouze v metodických listech**. Je na zvážení učitele, zda je žákům vzhledem k jejich programátorským schopnostem prozradí, či nikoliv.

Jak učit programování

Stejně jako Jiří Vaníček, Ingrid Nagyová a Monika Tomcsányiová v jejich učebnici programování se i my domníváme, že výuka programování by měla být založena na aktivitě žáka a jeho interakci s počítačem. Proto si s jejich laskavým souhlasem dovolujeme prezentovat jejich pohled na to, Jak učit programování, Jak hodnotit a známkovat žáky, Jak vést výuku a Jak se připravovat na výuku.

Programování je praktická činnost, jejíž výsledek je na počítači ihned po spuštění programu vidět. **Počítač žákovi poskytuje zpětnou vazbu**, tedy zda programoval správně nebo s chybou. **Učitel se musí naučit této výhody využít**.

Zde uvádíme **základní doporučení**, jak učit, aby se **žák učil aktivně si vytvářet znalosti**. V dalším textu je vysvětlíme.

1. Nesdělujme žákům správná řešení.
2. Neučme definice a pamětní poznatky.
3. Obrňme se trpělivostí.
4. Očekávejme živé a rušné prostředí (a podporujme jej).
5. Hodnoťme praktické dovednosti.

1. Učitel by **neměl dělat tu chybu, že on sám říká správné řešení**. Měl by ponechat na počítači, aby to žákovi ukázal. Pokud učitel prochází lavicemi a narazí na chybný programový kód, lépe než „Toto máš špatně“ je říci „Spusť mi tento program“. Příště si žák program zkontroluje sám a učí se samostatnosti.

   Většinou **neexistuje jedno správné řešení**. Učitel by neměl bazírovat na tom, že jeho řešení nebo řešení z učebnice je jediné správné. Naopak oceňujme žáky, kteří přicházejí s novými, netradičními řešeními (i když správná nebudou). Pokud je žákovo řešení málo

obecné, nastavme situaci, v níž **nebude** fungovat, a zadejme žákovi, aby své řešení předvedl.

2. **Nemá valného smyslu učit žáky definice**, aby uměli pěkně odpovídat na otázky. To z nich programátory neudělá a jejich myšlení to nijak nerozvine. Je také **zbytečné na začátku hodiny vše znovu vysvětlovat**, opakovat, žáky vyvolávat. Oni chtějí programovat, tak je nechejme. Praktickou činností si to zopakují rychleji.

   Pokud žák některé příkazy z minulých hodin zapomněl, může nahlédnout do svých poznámek nebo realizovaných programů. Žákům můžeme už v prvních lekcích doporučit, aby si průběžně **vytvářeli „tahák“** obsahující používané příkazy. Když žák zápis některého příkazu zapomene, snadno si jej vyhledá.

   Máte-li možnost výuky informatiky ve dvou hodinách týdně, nedoporučujeme tzv. „dvouhodinovky“. Žáci za týden příkazy zapomenou spíše než v případě výuky rozdělené rovnoměrněji do dvou dnů v týdnu.

3. **Obrňme se trpělivostí**. Trvá to nějaký čas, než si žáci na Python a jeho syntaktická pravidla zvyknou. Pak ale jakoby najednou celý ten systém pochopí a začnou pracovat více automaticky, efektivněji a samostatněji. Žáci nemusí věci ihned pečlivě zvládnout. Není třeba být v tomto směru úzkostlivý; žáci pochopí věci postupně.

4. **Očekávejme živé a rušné prostředí**. Při programování a při práci ve skupině je to přirozené. Pokud je učitel zvyklý, že při psaní a kreslení na počítači je ve třídě ticho, pak může být překvapen, že žáci si chtějí a potřebují něco vysvětlovat, ukazovat, projevují nahlas emoce. Učitel by měl vědět, že tyto projevy jsou znakem kvalitní výuky. Je na učiteli, aby si ohlídal, aby žáci nediskutovali mimo téma výuky nebo úlohy.

Jak hodnotit a známkovat žáky

Vždy hodnoťme praktické dovednosti: žák umí sestavit správný program, umí jej přečíst a vysvětlit, co program dělá, umí najít v programu chybu a opravit ji. Kromě tradičních metod (samostatná práce na naprogramování nějaké úlohy nebo test z úlohy na čtení programu) se nabízí další metody. Velmi vhodné je **pozorování** žáka či **individuální konzultování** jeho rozpracované práce. Známkování je pak snadné, protože je podobné slovnímu hodnocení.

Další možností je **analyzovat práci žáka** (nechat si soubor na konci hodiny uložit a prohlédnout si, jak žák sestavil programy, jestli používá nějaké pokročilejší příkazy jako opakování, podmínky atd.).

Nikomu tento způsob hodnocení nevnucujeme. Co bychom ovšem **neradi**, aby učitelé ze zkoušení dělali nástroj k udržení své autority, aby zkoušeli, jestli žáci rychle odpovídají na jednoduché otázky nebo jestli něco umí zpaměti – tím jistě schopnost programovat neověří.

Jako zvláštní kapitolu k metodice přidáváme **vzorové ukázky testů** či programovacích „písemek“, na nichž ukazujeme, jaké znalosti považujeme za důležité zkoušet. Učitel tyto testy vůbec použít nemusí; raději doporučíme jednu z výše uvedených metod hodnocení. Ukázky testů slouží k získání představy, jaké dovednosti z oblasti sestavení a čtení programu vidí autoři jako důležité.

Jak vést výuku

Aby žáci programování zvládli:

- Pokud program nedělá to, co žák chce, **není** dobrou cestou vše smazat a začít zapisovat nový program od začátku. Je vhodné s žákem **projít řádek po řádku, co program dělá**.
- **Nechvátejte**. Vše chce čas. V prvních několika hodinách se Vám bude zdát, že tempo je pomalé, žáci si nic nepamatují a nevědí, jak pracovat. Teprve časem žáci najednou začnou pracovat zcela samostatně, sami opravovat chyby, experimentovat.
- Situací, v nichž si nebudete vědět rady, se hlavně na začátku vyvarujete tím, že **nedovolíte žákům dělat si, co je napadne**. Pokud se žákovi „něco stalo“ s programem, často je to proto, že se nedržel pokynů úlohy, zkoušel něco jiného podle sebe. V takovém případě je vhodné nechat žáka pracovat od začátku podle pokynů.
- Nejen aktivity, v nichž žáci programují samostatně, jsou důležité. **Nepřeskakujte úlohy na čtení programu**, na diskusi, objevování. Žáci se tak učí dívat se na problémy komplexněji, z více úhlů, a lépe rozumí programovému kódu.
- Je vhodné občas **přiznat, že něco nevíte**, např. že nevíte, čím je chyba způsobena. Nic se neděje, žák aspoň příště bude více spoléhat na sebe, bude experimentovat rozumně.

Jak se připravovat na výuku

- Pokud učíte podle této učebnice poprvé, je třeba **vyhradit si dostatek času**. Počítejte s tím, že Vám **příprava vezme více času, než výuka** samotná. Toto hrozí především po několika lekcích, když už látce „v globálu“ budete rozumět, ale bez precizní přípravy bude riziko, že při výuce budete dělat chyby, kterých si ani nebudete vědomi.
- **Nestačí, když budete mít před žáky „náskok“ jednu lekci**. Potřebujete vědět, k čemu zvládnuté dovednosti směřují a jak budou používány v dalších lekcích.
- Projděte si úlohy v pracovních listech a **zkuste je nejprve vyřešit sami, bez nápovědy**. Teprve pak se podívejte do metodiky. Projdete si tak chybami, které mohou vaši žáci dělat také. „Kouknutí se“ na výsledky nikoho programovat nenaučilo.
- **Prostudujte si metodické listy**, i když jste úlohy sami úspěšně vyřešili. Kromě správných řešení jsou zde uvedeny typické chyby, které mohou žáci udělat; možné postupy, jak žákům při řešení úloh pomoci, aniž byste jim prozradili správná řešení; a rozšiřující informace k danému učivu.
- Některé úlohy se zdají na pohled snadné a člověka to **svádí k tomu**, že jejich vyzkoušení při přípravě na hodinu **přeskočí**. Tím si může zadělat na problém při samotné hodině, protože se mu program může chovat jinak, než očekával.
- Je dobré vědět, jaké **cíle** se vlastně výukou sledují. Jsou uvedeny v metodice u úloh nebo na začátku lekcí.

Zkušenému učiteli netřeba radit. Až budete učit druhým rokem, řadu problémů nebudete jako problém vůbec vnímat. Až budete učit potřetí, možná metodické listy nebudete ani potřebovat.

Naplnění revidovaných RVP

Základní učivo učebnice pokrývá očekávané výstupy zejména v informatickém tématu Algoritmizace a programování:

- Vysvětlí daný algoritmus, program; určí, zda je daný postup algoritmem
    - Už od 3. lekce žáci sestavují programy, což jsou algoritmy zapsané v jazyce Python
- Rozdělí problém na menší části, rozhodne, které je vhodné řešit algoritmicky, své rozhodnutí zdůvodní; sestaví a zapíše algoritmy pro řešení problému
    - Od 8. lekce žáci řeší úlohy rozdělováním problémů na podproblémy pomocí podprogramů
- Zobecní řešení pro širší třídu problémů; ověří správnost, najde a opraví případnou chybu v algoritmu
- Ve vztahu k charakteru a velikosti vstupu hodnotí nároky algoritmů; algoritmy podle různých hledisek porovná a vybere pro řešený problém ten nejvhodnější; vylepší algoritmus podle zvoleného hlediska
    - Žáci řeší problémy pro různé hodnoty vstupních proměnných,
    - Žáci pomocí konstrukcí větvení a cyklů zobecňují řešení problémů
    - Žáci hledají a opravují chyby ve svých programech
- V textovém programovacím jazyce sestaví přehledný program, ten otestuje a optimalizuje
    - Žáci programují v textovém programovacím jazyce Python
- Používá opakování, větvení programu se složenými podmínkami, proměnné, seznamy a objekty, podprogramy s parametry a návratovými hodnotami, externí knihovny; ve snaze o vyšší efektivitu navrhuje, řídí a hodnotí souběh procesů
    - Žáci používají příkazy přiřazení do proměnných, příkazy větvení a opakovaní
    - Žáci vytvářejí podprogramy s parametry a pracují s podprogramy z externích knihoven i s podprogramy s návratovými hodnotami
    - Žáci vytvářejí grafické objekty v grafické ploše a manipulují s nimi
    - Žáci se seznamují s programováním souběžných procesů při programování obsluhy událostí myši

Instalace Pythonu

Python je open source a je možné jej zdarma stáhnout a instalovat mimo jiné pod operačními systémy MS Windows, Linux nebo MacOS. Pro MS Windows doporučujeme instalační exe soubory stáhnout ze stránek https://www.python.org. NEinstalujte verzi **2.7**, ale verzi minimálně **3.6.0** nebo vyšší (v době přípravy této učebnice to byla verze **3.8.1**).

V tomto kurzu budeme pracovat ve standardním vývojovém prostředí **IDLE**, které je součástí instalace. Proto budeme Python spouštět prostřednictvím IDLE, jehož ikonu naleznete v menu Start nebo na Ploše s názvem např. **IDLE (Python 3.8)**. Je vhodné tuto ikonu na žákovských

počítačích vložit na Plochu nebo do panelu Rychlého spouštění, aby byla žákům snadno přístupná.

Po spuštění lze v prostředí IDLE upravit velikost písma, případně nastavit automatické ukládání vytvářených programů před jejich spuštěním.

Pokud budete od žáků vyžadovat či očekávat domácí přípravu na výuku, je potřebné jim názorně ukázat, odkud a jak Python instalovat. Vhodným řešením může být Python před žáky nainstalovat na učitelském počítači. Nezaměřujte se přitom pouze na samotný proces instalace, ale zobrazte žákům stránky https://www.python.org, ukažte, kde konkrétně najdou potřebné instalační balíčky a doporučte jim konkrétní odkaz, pomocí něhož lze Python stáhnout a následně instalovat na počítač s operačním systémem MS Windows. Tyto technické záležitosti však nedoporučujeme řešit před začátkem samotné výuky, ale až později, kdy budou žáci zkušenější a budou se v prostředí Pythonu samostatně orientovat. Vhodná doba může např. po absolvování 3. lekce, kdy si žáci budou moci doma vyzkoušet programy, které vytvořili při výuce, a případně je vylepšovat.

Alternativní vývojová prostředí

Pro Python existuje množství různých vývojových prostředí. Pokud uznáte za vhodné, nemusíte pro výuku Pythonu využívat standardní prostředí IDLE, ale můžete zvolit některé z alternativních prostředí, např.:

- PyCharm Edu - Easy and Professional Tool to Learn & Teach Programming with Python
- Wing Personal - A free Python IDE for students and hobbyists
- Thonny - Python IDE for beginners

Přehled vývojových prostředí naleznete na stránce Integrated Development Environments.

Je nutné si však uvědomit, že potom by si takovéto prostředí žáci měli nainstalovat i na svých domácích počítačích, což pro některé z nich (zejména pro začátečníky) nemusí být jednoduchá záležitost.