Python

Anino Belan

eUčebnica pre septimu 8 ročného alebo pre 3. ročník 4 ročného gymnázia

Anino Belan

Python

eUčebnica pre septimu osemročného alebo 3. ročník štvorročného gymnázia

2013

Autorské práva

ISBN: 978-80-89646-35-7

Úvod

Kedysi v dávnych časoch, keď domáce počítače mali osembitové procesory, keď neexistoval Linux, Mac OS ani Windows a keď grafika počítača zvládla maximálne osem farieb, sa s počítačmi robili prevažne dve veci. Hrali sa na nich hry a programovalo sa. Obe činnosti robili ľudia sami od seba a dobrovoľne. Kto mal prístup k počítaču, užíval si jednak to, že sa mohol stať mocným Pacmanom a žrať strašidlá, jednak to, že sa mohol stať mocným programátorom a niekoľkými šikovnými príkazmi mohol prinútiť počítač, aby robil presne to, čo chce. A keď niekto prišiel s nápadom, že by sa informatika mala učiť v škole, tak sa s týchto dvoch vecí začalo prirodzene učiť práve to programovanie.

Lenže doba pokročila. Počítače majú namiesto osembitových procesorov šesťdesiatštyribitové, medzi operačnými systémami si môžete vyberať a farbami sa nešetrí. Okrem toho sa počítače využívajú na mnohé šikovné veci. Môžete v nich písať typograficky pekne upravené texty, vytvárať tabuľky so zabudovanými výpočtami, počúvať či vytvárať hudbu a video, robiť 3D modely a animácie, skladovať dáta, môžete sa s ich pomocou pripájať do siete a komunikovať s celým svetom.

Tieto zmeny sa samozrejme odrazili aj na vyučovaní informatiky. Na informatike sa učí množstvo vecí, ktoré by sa mohli učiť skôr na slovenčine, hudobnej a výtvarnej výchove a matematike a jediný dôvod, prečo tieto veci pripadli pod predmet informatika je, že sa pri nich používa počítač. Z tohto dôvodu sa na informatike programuje oveľa menej, ako kedysi.

Táto knižka sa vás napriek tomu bude pokúšať naučiť programovať. Dôvody sú viaceré.

Prvý je ten, že vás programovanie môže uživiť. Je to remeslo, ktoré je zaujímavé a okrem toho ešte aj celkom slušne platené. Programátorov je málo, takže nie je problém nájsť si v odbore zamestnanie. Keď sa vydáte touto cestou, môže sa vám stať životným povolaním.

Toto sa samozrejme netýka každého. Aj pre ľudí, ktorí po skončení tohto kurzu už nenapíšu ani riadok kódu ale má zmysel, že ho absolvujú. A to z týchto dôvodov:

- Keď človek aspoň trošku programuje, tak má lepšiu predstavu o tom, čo môže od programov, ktoré pre neho píše niekto iný, čakať. A viac ocení, keď hrá nejakú skvelú hru alebo používa nejaký program, ktorý mu zjednoduší život.
- Ľudia si väčšinou myslia, že vedia veci dobre vysvetliť a väčšinou sa v tejto predstave o sebe žalostne mýlia. Programovať znamená niečo dobre vysvetliť počítaču. Dosť dobre na to, aby robil, čo od neho chceme. A táto schopnosť niečo dobre vysvetliť sa človeku zíde aj inde, než pri práci s počítačom.

- Programovanie má tú výhodu, že keď človek spraví nejakú chybu, príde na to pomerne rýchlo. V živote to tak vždy byť nemusí. Preto je programovanie zaujímavá skúsenosť, vďaka ktorej sa má človek možnosť naučiť vyrovnávať sa s vlastnými chybami a učiť sa ich opravovať.

Programovať sa dá rôznymi spôsobmi a v rôznych jazykoch. Napriek tomu, že na Slovensku sa v školách väčšinou učí Pascal alebo niektorý z jeho klonov (Delphi, Lazarus), rozhodli sme sa zvoliť iný jazyk – Python. Dôvody sú dva.

Prvý je ten, že jazyk Python je pohodlný. Robí sa v ňom príjemne a rýchlo. Na veci, ktoré by ste v Pascale potrebovali štyri riadky, vám bude stačiť jeden, to čo napíšete, bude pravdepodobne čitateľné aj pre niekoho iného, ako pre vás a namiesto toho, aby ste sa museli sústrediť na množstvo formalít, môžete sa sústrediť na to, čo chcete naprogramovať.

Druhý je ten, že časy, kedy bol Pascal najvhodnejší jazyk na vyučovanie, už dávno pominuli. Keď sa pozriete, ktoré jazyky sa učia v začiatočníckych kurzoch na univerzitách na západe, tak sú to buď jazyky, ktoré sú syntaxou odvodené od jazyka C (teda C, C++ a Java), niektorý funkcionálny jazyk (Haskell), alebo práve Python. Medzi univerzity, ktoré začínajú s Pythonom patrí napríklad Cambridge alebo MIT. Python je totiž jazyk, ktorý samotnou svojou syntaxou núti ľudí, aby písali programy čitateľne. Návyky, ktoré takto získate, oceníte, keď budete programovať v iných jazykoch. K jazyku Python okrem toho existuje mnoho veľmi pekne napísaných knižníc, čo z neho robí rovnako univerzálny jazyk, ako je C++

alebo Java.

V každom prípade sa teším, že ste po tejto knižke siahli a dúfam, že sa vám bude páčiť a bude pre vás užitočná. Želám vám veľa trpezlivosti, pevné nervy, psychickú odolnosť a príjemnú zábavu.

1. lekcia

Python alebo „Nebezpečné hady“

Python začal vytvárať Holanďan Guido van Rossum v roku 1989 a dodnes je jeho hlavným vývojárom. Názov dal jazyku podľa anglickej humoristickej skupiny Lietajúci cirkus Montyho Pythona. Dnes sa používajú dve jeho verzie – verzia 2 a verzia 3. Tento kurz bude zameraný na verziu 3.

Inštalácia je jednoduchá. Pod Linuxom budete mať Python pravdepodobne rovno nainštalovaný. Ak náhodou nie, stačí použiť váš obľúbený balíčkovací systém a nechať automaticky nainštalovať balíček s názvom `python3`. Ak používate Windows alebo Mac OS X, stačí zájsť na stránku http://www.python.org/getit a stiahnuť si odtiaľ patričný balíček. Dajte si pozor, aby ste stiahli verziu, ktorej číslo začína na 3. Keďže je Python zverejnený pod OpenSource licenciou, nájdete tam aj jeho zdrojové kódy (jadro Pythonu je napísané v Cčku).

Python je teda šťastne nainštalovaný. Čo s ním. Prvá vec, ktorú môžete spraviť, je, že spustíte priamo program `python3`. Keď tak učiníte, ukáže sa vám na obrazovke niečo takéto[1]:

```
Python 3.2.3 (default, Jun  8 2012, 05:36:09)
[GCC 4.7.0 20120507 (Red Hat 4.7.0-5)] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Tie zobáky na konci znamenajú, že Python počúva. Môžete mu zadať príkaz a on ho okamžite vykoná. Môže to vyzerať napríklad takto:

```
>>> print(2+3)
5
>>> print(1+1+1+0*9)
3
>>> print("Dobrý deň")
Dobrý deň
>>> print(9/4)
2.25
>>> print(2**5)
32
>>> kr = 5
>>> print(kr)
5

>>> kr = "Ahoj!"
>>> print(kr)
Ahoj!
>>> quit()
```

Ako si môžete všimnúť, Python vie fungovať celkom dobre ako kalkulačka. Funkcia `print` slúži na to, keď potrebujete niečo vypísať. Matematiku Python zvláda celkom dobre, takže keď mu dáte vypísať hodnotu `1+1+1+0*9`, vie, že násobenie má prioritu pred sčítaním, takže najprv vypočíta to `0*9` (hviezdička znamená „krát“) a potom k tomu pripočíta tie

jedničky. Dobre mu ide aj delenie a umocňovanie (`2**5` znamená 2^5).

Zaujímavý je príkaz `kr = 5`. Znamená totiž „do krabice niekde v pamäti, ktorá sa nazýva `kr` vlož hodnotu 5“. Takéto krabice v pamäti sú veľmi užitočné a nazývajú sa premenné. Meno premennej si môžete vymyslieť aké chcete.[2] Všimnite si, že po vykonaní tohto príkazu Python nič nevypisoval a iba sa objavili ďalšie zobáky. Až keď sme použili funkciu `print`, obsah sme vypísali. Keď potom vykonáme príkaz `kr = "Ahoj!"`, v premennej `kr` sa ocitne nový obsah – reťazec „Ahoj!“ – a predošlá hodnota 5 bude navždy zabudnutá. Keď znovu vypíšeme obsah premennej `kr`, dočkáme sa iného výsledku, ako v predošlom prípade. Funkcia `quit` Python ukončí. Je dôležité, aby funkcia mala za sebou zátvorky. Podľa nich ju vie Python odlíšiť od premennej. Preto má funkcia `quit` za sebou zátvorky, aj keď v nich nemá žiadne parametre.

Toto je jeden spôsob, ako Python používať. Nie je to úplne zlý spôsob – podobne sa používa príkazový riadok. Problém je, že keď chceme tieto všetky veci znovu zopakovať, tak musíme príkazy odznovu napísať.

Nie je ale nič jednoduchšie, ako si zvoliť obľúbený textový editor (napríklad kate, vi, notepad++, TextWrangler ale pokojne aj notepad), všetky príkazy do neho napísať, súbor uložiť s koncovkou `.py` (teda napríklad `01-01-pokus.py` – to prvé číslo je číslo lekcie, druhé je číslo programu v danej lekcii). Súbor teda bude obsahovať nasledujúci text:

```
print(2+3)
print(1+1+1+0*9)
print("Dobrý deň")
print(9/4)
print(2**5)
kr = 5
print(kr)
kr = "Ahoj!"
print(kr)
quit()
```

Teraz už len treba povedať Pythonu, aby ten súbor spustil. Ak pracujete pod linuxom, situácia je jednoduchá. Stačí sa v príkazovom riadku dostať do adresára, v ktorom máte súbor uložený a zadať príkaz

```
python3 01-01-pokus.py
```

Program vám vypíše

```
5
3
Dobrý deň
2.25
32
5
Ahoj!
```

Pod windowsami je situácia trochu zložitejšia. Aby sa súbor s vašimi príkazmi spustil, stačí naň dvojkliknúť. A skutočne. Otvorí sa okno, do neho sa vypíšu výsledky, program skončí a okno sa zavrie. Problém je v tom, že toto všetko trvá len

zlomok sekundy a vy poriadne nestihnete uvidieť, čo v tom okne vlastne je. Aby ste tomu zamedzili, pred príkaz `quit()` vložte ďalší príkaz

```
input("Stlačte ENTER!")
```

Tento príkaz vypíše `Stlačte ENTER!` a čaká na to, kedy mu používateľ niečo zadá. A aj keď mu nezadáte nič, tak to nevadí. Pre nás je dôležité to čakanie. Kým nestlačíte `ENTER`, okno nezmizne a vy sa môžete kochať tým, čo pekné vám Python vypísal.

Úloha 1: Spustite Python a zadajte mu uvedené príkazy. Vymyslite si nejaké vlastné a tie mu zadajte tiež.

Úloha 2: Čo vypíše Python, keď mu zadáte príkaz `print(10 * "Ahoj ")`

Úloha 3: Vo svojom obľúbenom textovom editore napíšte uvedené príkazy, uložte to pod nejakým rozumným menom s koncovkou `.py` a súbor spustite.

Úloha 4: Čo to spraví, keď do nejakého súboru napíšete príkazy

```
a = "(auto)"
b = "(autobus)"
zapcha = 3*a + 2*b + 4*a + 3*b + a
print(zapcha)
```

a potom to spustíte?

Úloha 5: Nechajte Python vypočítať súčet nepárnych čísel od 1 do 19 a výsledok vypíšte.

Úloha 6: Nechajte Python vypočítať 20! (dvadsať faktoriál) a výsledok vypíšte.

Úloha 7: (Bonusová pre machrov) Spravte zápchu, ktorá nebude vypísaná, ako v úlohe 4, ale vykreslením autíčka nasledujúcim spôsobom:

```
 _/ L\__,
'-o---o-'
```

Ak do reťazca pridáte `"\n"`, znamená to „prejdi na nový riadok“.

---

1 To, čo sa vám ukáže, samozrejme závisí od použitého operačného systému. V prípade linuxu budete program pravdepodobne spúšťať v príkazovom riadku, v prípade windows sa po spustení programu objaví terminál. Aj text, ktorý sa objaví, môže byť iný. Dôležité je, že vám Python povie, že sa spustil, uvedie použitú verziu a na konci sa objavia tie tri zobáky.
2 Dávajte si ale pozor, aby nezačínalo na číslicu a obsahovalo iba písmená, číslice a znak `_`.

2. lekcia

Podmienky a výnimky alebo „Mocné odsadenie“

Prvá lekcia, ktorú ste mali tú česť prednedávnom dokončiť, bola celkom jednoduchá. Pythonu ste voľačo prikázali a on to urobil. Na prvý pohľad sa môže zdať, že kurz splnil svoje poslanie. Počítač vás poslúcha a robí presne to, čo ste mu povedali. Na druhú stranu, skončiť kurz Pythonu prvou lekciou, to predsa len vyvoláva istý pocit nenaplnenia a neúplnosti. Nebojte sa, Python ešte nepovedal svoje posledné slovo.

Na tých programoch, ktoré sme robili minule, je totiž jedna nesympatická vec – zakaždým spravia to isté. Vôbec sa používateľa nespýtajú, čo by chcel a či by to pri ďalšom behu programu náhodou nechcel nejako inak. To sa dá ale ľahko napraviť. Pozrite si nasledujúci program:

```
print("Ako sa voláš?")
meno = input()
vystup = "Ahoj " + meno + ", ja som Python."
print(vystup)
```

Všetko to napíšte do súboru s koncovkou `.py` a spustite. Keď

program spustíte, najprv sa vás slušne opýta, ako sa voláte. Potom príde zaujímavá časť. To, čo je v tomto programe nové, je funkcia `input`. Tá čaká, kým niečo napíšete a stlačíte `Enter`. Text, ktorý napíšete, vráti funkcia ako svoj výsledok, s ktorým potom môžete robiť, čo uznáte za vhodné. My sme si ho uložili do premennej `meno`.

V treťom riadku vytvoríme premennú `vystup`, do ktorej uložíme za sebou zreťazené texty `"Ahoj "`, meno, ktoré sme načítali zo vstupu a `", ja som Python"`. V štvrtom riadku tento výsledok vypíšeme.

Úloha 1: Pochopte a vyskúšajte si to.

Keď už vieme získať od užívateľa nejakú spätnú väzbu, môžeme skúsiť naprogramovať niečo praktickejšie. Napríklad hrací automat.

```
print("Ja som hrací automat.")
print("Stav desať eur.")
heslo = input("Zadaj heslo: ")
if heslo == "krokodil":
→       print("Vyhral si.")
→       print("Môžeš si zobrať tých desať eur naspäť.")
else:
→       print("Prehral si.")
→       print("Desať eur ti prepadlo.")
print("Isto si chceš zahrať ešte raz.")
```

Prvý zaujímavý detail sa vyskytuje v treťom riadku. Funkcia `input` tam totiž má parameter `"Zadaj heslo: "`. To spôsobí, že funkcia najprv napíše `Zadaj heslo:` a až potom čaká na vstup od užívateľa.

Tie dôležité veci sa ale dejú až na riadkoch 4 až 9. Heslo, ktoré užívateľ zadal, máme uložené v premennej `heslo`. Ak užívateľ náhodou trafil správne heslo „krokodil“[2], tak mu chceme napísať, že vyhral a môže si vklad zobrať naspäť. Inak mu chceme napísať, že prehral. Náš program teda potrebujeme rozdeliť na dve nezávislé vetvy, pričom jedna sa vykoná vtedy, keď bude v premennej `heslo` „krokodil“ a druhá sa vykoná vtedy, keď tam bude niečo iné.

Presne na to slúži príkaz `if` (po anglicky „ak“). Konkrétne v našom prípade vyzerá takto:

```
if heslo == "krokodil":
```

Za príkazom `if` nasleduje podmienka. Naša podmienka `heslo == "krokodil"` je pradivá iba vtedy, ak je v premennej `heslo` je uložený text „krokodil“. Za podmienkou treba dať dvojbodku, aby bolo jasné, že už podmienka skončila. Tie znaky „rovná sa“ v tej podmienke musia byť dve, pretože jedno „rovná sa“ už používame, keď chceme niečo vložiť do nejakej premennej.

A teraz sa dostávame k veci, podľa ktorej sa nazýva celá táto kapitola, teda k mocnému odsadeniu. Keď sa pozriete na riadky 5 a 6, teda tie, ktoré sa majú vykonať v prípade, že používateľ ako heslo skutočne zadal `"krokodil"`, tak sú oproti riadku s podmienkou odsadené o jeden tabulátor. (Tabulátor je naznačený šípkou. Tie šípky do svojho programu nepíšte, dajte tam tabulátory!) Python z toho

pochopí, že to, čo je odsadené, sa má vykonať len vtedy, ak je podmienka splnená. Keď sa dostane k ďalším neodsadeným riadkom, tak vie, že ten podmienený úsek už skončil a zas má robiť všetko.

Po skončení podmienenej časti nasleduje príkaz

```
else:
```

Tento príkaz sa nikdy nevyskytuje samostatne a vždy sa musí spájať s nejakým predošlým `if`. (Naopak to neplatí – `if` samostatne môže byť a príkaz `else` po ňom nasledovať nemusí.) „Else" po anglicky znamená „inak". Riadky, ktoré po `else` nasledujú (a sú odsadené o tabulátor), sa teda vykonajú iba vtedy, keď podmienka v predošlom príkaze `if` nie je splnená. Ak teda používateľ heslo neuhádne, vykonajú sa riadky 8 a 9 a program mu oznámi, že prišiel o peniaze.

Riadok 10 nie je odsadený. Vykoná sa teda bez ohľadu na to, aké heslo užívateľ zadal a vyzve hráča, aby si zahral ešte raz.

Úloha 2: Pochopte, vyskúšajte si to a neprepadnite hráčskej závislosti. Aj tak nevyhráte.

Úloha 3: Pridajte pred riadok 4 príkaz `print(heslo == "krokodil")` Čo bude tento príkaz vypisovať?

Čo robiť, ak chcete napísať nejakú komplikovanejšiu vec a napríklad do jednej vetvy programu vložiť ďalší príkaz `if`? Je to jednoduché. Stačí pridať ďalší tabulátor ako v

nasledujúcom programe:

```
odpoved = input("Chceš mi povedať číslo? ")
if odpoved == "ano":
→       cislo = input("Tak povedz: ")
→       if cislo != "42":
→       →       print("Aha...")
→       else:
→       →       print("Óóóóó :)")
```

Program už nebudeme komentovať tak podrobne, ako predošlý. Na začiatku sa opýtame užívateľa, či sa s nami vôbec hodlá baviť a ďalej sa bude niečo robiť iba ak odpovie `"ano"`. Keďže celá ďalšia časť programu od riadku 3 po riadok 7 je podmienená, všetko je odsadené o tabulátor. V tejto časti sa používateľa program opýta, aké číslo mu to chcel povedať a ak to bude 42[2], patrične to ocení. Riadky 5 a 7 sú podmienené vďaka príkazom `if` a `else` na riadkoch 4 a 6. Keďže sú ale riadky 4 a 6 už o jeden tabulátor odsadené, riadky 5 a 7 musia byť odsadené až o dva.

Všimnite si na tomto programe niekoľko ďalších zaujímavostí. Prvá je dvojica znakov `!=` v riadku 4. Táto dvojica znamená „nerovná sa“. Ak teda užívateľ zadá iné číslo, ako 42, program to veľmi neocení.

Ďalšiu zaujímavú vec uvidíte, keď sa na program pozriete z väčšej diaľky. To odsadzovanie je dobré nie len pre Python, ale aj pre vás. Vďaka nemu je rovno vidieť, ktoré časti kódu patria ku ktorým podmienkam a program sa preto dobre číta. Vidíte napríklad hneď, že `else` na riadku 6 patrí k `if` na riadku 4 a nie k `if` na riadku 2. K podobnému odsadzovaniu