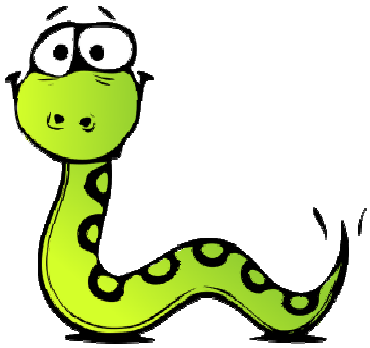
Programování v jazyce Python pro střední školy

Metodický list pro učitele

Lekce 2 – Proměnné



Andrej Blaho

Ľubomír Salanci

Václav Šimandl

Cíle lekce

- Naučit se, co je to proměnná a jak funguje přiřazovací příkaz
- Naučit se používat proměnné ve výrazech a vyhodnocovat takové výrazy

Dovednosti

- Znázornění obsahu proměnných na papír

Osvojená syntaktická pravidla

- Zápis proměnných a přiřazovacího příkazu
- Nesprávné zápisy proměnných a čísel

Průběh výuky

Cílem první úlohy je připomenout si zapisování výrazů v jazyce Python z minulé lekce:

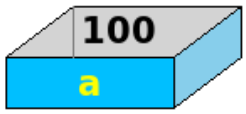
1. Spusť Python a nech jej vypočítat, čemu se rovná výraz `(123 + 456) * 789`

Proměnné mohou být náročným konceptem. Proto je cílem tohoto pracovního listu, aby se žáci postupně seznámili s jednoduchými proměnnými. Zatím do proměnných přiřazují jen čísla a ty potom používají v elementárních úlohách.

2. V matematice je zvykem označovat hodnoty písmeny, například délka strany čtverce `a = 100`. To samé můžeš udělat i v Pythonu. Zkus napsat:

   `>>> a = 100` a potvrď klávesou *Enter*

   Jestli se nic nevypsalo (ani žádná chyba), je to správně. Python si vytvořil proměnnou s názvem `a` a přitom si zapamatoval, že má hodnotu `100`. Toto můžeme znázornit pomocí krabičky vpravo:



3. Zkus nyní napsat jen:

   `>>> a` a potvrď klávesou *Enter*

   Uvidíš, jakou hodnotu si Python pamatuje v proměnné `a`.

Pokud si někteří žáci nebudou jisti principem fungování proměnných, můžeme jim pomoci následujícím vysvětlením: Proměnnou si můžeme představit jako krabičku, do níž lze vložit určitou hodnotu. My jsme pomocí zápisu `>>> a = 100` zajistili, aby se vytvořila proměnná (krabička) s názvem `a` a vložila se do ní hodnota `100`.

I další úloha slouží ke sbírání prvních zkušeností s proměnnými. Žáci by měli zjistit, že:

- je možné používat více proměnných,
- proměnné mohou mít delší názvy,
- do proměnné lze přiřadit hodnota výrazu.

4. Vyzkoušej vytvořit a nastavit i jiné proměnné:

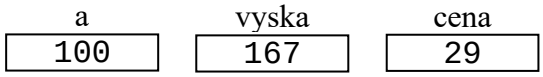
```
>>> vyska = 167
>>> cena = 22 + 7
```

Znázornit je můžeme následovně:

5. Zkontroluj, zda proměnné s názvy `vyska`, `cena` mají správné hodnoty.

Proměnná funguje podobně jako paměť kalkulačky (tlačítko `M`) – do ní si lze uložit jednu hodnotu a tu později použít v dalších výpočtech. V Pythonu si můžeš vytvořit libovolný počet takovýchto „pamětí“.

V tomto okamžiku bude v programu více proměnných. Je důležité, aby si žáci o proměnných začali vytvářet nějakou představu. Proto proměnné znázorňujeme jako krabičky, které mají svoje označení a které obsahují přiřazenou hodnotu. Bylo by dobré, aby se žáci naučili znázorňovat proměnné také sami, například do sešitu. Je vhodné, abychom se žáky diskutovali a v případě potřeby kreslili krabičky i na tabuli – stačí takto jednoduše:



Je vhodné si zvyknout i na chybová hlášení, která žáci uvidí, když použijí neexistující proměnnou, případně se zmýlí při psaní názvu proměnné (překlep):

6. Zkus napsat:

```
>>> vek
```

a potvrď klávesou *Enter*

Jestliže proměnná `vek` neexistuje, vypíše se několik řádků s chybovým hlášením – pro odhalení chyby je důležitý poslední řádek:

```
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    vek
NameError: name 'vek' is not defined
```

... proměnná `vek` neexistuje

Následuje úloha, ve které se používají proměnné ve výrazech:

7. Proměnné můžeš použít i v matematických zápisech a Python namísto názvu proměnné dosadí její hodnotu. Urči výsledek následujících příkazů:

```
>>> 190 - vyska
>>> 3 * cena + 10
>>> cena + vyska
```

Když není žákům jasné, jak se výraz vyhodnocuje, případně proč počítač zobrazuje daný výsledek, doporučujeme výraz napsat na tabuli a jeho vyhodnocení odkrokovat (například: „Počítač se podívá do proměnné `vyska`, dosadí její hodnotu do výrazu `190 - vyska`. Bude počítat `190 - 167`. Na obrazovce uvidíme výsledek `23`“).

Následují úlohy, ve kterých se mění obsah proměnných:

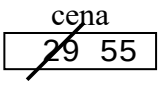
8. Proměnným můžeme změnit jejich obsah – vyzkoušej:

```
>>> cena = 5 * 11
```

Momentální stav paměti bychom mohli zakreslit takto – všimni si, že se změnila proměnná `cena`:

9. Změň hodnotu proměnné `vyska` tak, aby v ní byla tvoje výška v centimetrech. Přesvědč se, že se tak stalo.

Pokud je to potřeba, doporučujeme změny znázorňovat na tabuli, například:



Následující úloha je důležitá, neboť je na ní zřejmé, že proměnné si nepamatují vztahy, ale hodnoty (tj. proměnná `obsah` si zapamatuje `10000`, nikoliv vzorec `a * a`):

10. Zkus i takovéto příkazy – co vykonají?

```
>>> obsah = a * a
>>> obsah
>>> a = 1
>>> obsah
```

Znázorni obsah proměnných pomocí krabiček.

Někteří žáci se mohou mylně domnívat, že po změně proměnné `a` se automaticky přepočítá také hodnota proměnné `obsah`. Tedy že po přiřazení `a = 1` bude v proměnné `obsah` hodnota `1`. Takovouto zkušenost mohou mít z tabulkového procesoru.

V diskuzi se žáky shrneme dosavadní zkušenosti a sestavíme následující všeobecné schéma, jak funguje příkaz přiřazení (toto nemá smysl dělat na začátku lekce).

Přiřazovací příkaz je takový zápis, ve kterém se před znakem = nachází nějaký *nazev* proměnné a za znakem = je *hodnota*, kterou je třeba do této proměnné uložit:

```
nazev = hodnota
```

Když je *hodnotou* aritmetický výraz, tak se nejprve vyhodnotí a až potom přiřadí (nastaví) do proměnné.

Následují úlohy na procvičení, ve kterých je potřeba sestavovat jednoduché posloupnosti přiřazení a ověřovat, zda proměnné obsahují očekávanou hodnotu. Žáci úlohy stále řeší v interaktivním režimu.

11. Přiřaď do proměnné `zmrzlina` cenu jedné zmrzliny (například 25 korun). Do proměnné `pocet` přiraď počet kamarádů, kterým chceš koupit po jedné zmrzlině. Za použití proměnných sestav přiřazovací příkaz, pomocí kterého se do třetí proměnné `zaplatit` přiřadí suma, kterou zaplatíš. Přesvědč se, že to počítač dobře vypočítal.

Předpokládaný postup řešení:

```
>>> zmrzlina = 25
>>> pocet = 7
>>> zaplatit = zmrzlina * pocet
>>> zaplatit
175
```

Při zápisu složitějších příkazů v interaktivním režimu může snadno dojít k překlepu, kvůli němuž není příkaz vykonán. Abychom nemuseli psát celý příkaz znovu, umístíme kurzor myši do řádku s chybným zápisem a stiskneme klávesu *Enter*. Tím se příkaz z daného řádku zkopíruje na aktuální řádek a je možné jej upravit a následně nechat vykonat. Je však třeba si uvědomit, že upravený příkaz nenahrazuje ten původní, ale že jde o nově zadaný příkaz jako by byl zapsán ručně. Uvedený postup je technickou fintou, a proto jej začátečníkům neprozrazujeme.

12. Přiřaď do proměnných `delka`, `sirka` a `hloubka` rozměry školního bazénu v centimetrech (například s hodnotami `delka = 2500`, `sirka = 1000`, `hloubka = 180`). Sestav přiřazovací příkaz:

    a) kterým se přiřadí do proměnné `litry`, kolik litrů vody je třeba na napuštění celého bazénu,
    b) kterým se do proměnné `objem` přiřadí, kolik je to kubických metrů vody.

Očekávané řešení:

```
>>> delka = 2500
>>> sirka = 1000
>>> hloubka = 180
>>> litry = delka * sirka * hloubka / 1000
>>> litry
450000.0
>>> objem = litry / 1000
>>> objem
450.0
```

13. Vytvoř příkazy odpovídající zadání:

- do proměnné `x` přiřaď nějakou hodnotu
- zobraz hodnotu následujícího výrazu: k hodnotě proměnné `x` připočítej `1`, výsledek vynásob `2`, opět k výsledku připočítej `1` a vynásob `2` a do třetice opět k výsledku připočítej `1` a vynásob `2`.

Například pro `x` rovno `5`, bys měl(a) dostat výsledek `54`.

Řešení:

```
>>> x = 5
>>> (((x + 1) * 2 + 1) * 2 + 1) * 2
54
```

14. V matematice se počítá faktoriál nějakého čísla *n* jako součin čísel od `1` do *n*. Například faktoriál čísla `4` spočítáme jako součin čísel `1 * 2 * 3 * 4`. Do proměnné `faktorial10` přiřaď hodnotu faktoriálu čísla `10` (součin čísel od `1` do `10`). Hodnotu proměnné `faktorial10` poté zobraz.

Řešení:

```
>>> faktorial10 = 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10
>>> faktorial10
3628800
```

Cílem následujících úloh je, aby se žáci seznámili s některými pravidly pro pojmenování proměnných:

15. Všimni si názvů proměnných v následujících příkazech a znázorni proměnné pomocí krabiček. Poté příkazy vyzkoušej:

```
>>> strana_ctverce = 150
>>> obvod_ctverce = 4 * strana_ctverce
>>> obsah_ctverce = strana_ctverce * strana_ctverce
```

strana_ctverce	obvod_ctverce	obsah_ctverce
150	600	22500

Je vhodné si zvyknout používat delší názvy proměnných, neboť „samodokumentují“ programy – když zvolíme dobrý název, později lépe rozumíme účelu, ke kterému proměnná slouží.

Proměnným můžeš dát téměř libovolný název sestavený z písmen, číslic a podtržítek. Nesmí však začínat číslicí, nemohou obsahovat mezeru ani jiné speciální znaky (tečka, plus, mínus atd.).

Víme, že názvy proměnných se nesmí shodovat s vyhrazenými slovy (`for`, `def`, ...). Domníváme se, že momentálně nemá smysl tímto žáky zatěžovat, protože žádné vyhrazené slovo neznají. Doporučujeme jim to prozradit, až když taková situace nastane (například kdyby použili `def = 123`).

16. V matematice značíme obsah kruhu `S` a počítáme jej podle vzorce πr^2. Obvod kruhu značíme `O` a počítáme jej podle vzorce $2\pi r$. Zkus (podobně jako v úloze 15) nazvat proměnné pro poloměr, obsah i obvod kruhu vhodnými delšími názvy a přiřaď do nich správné výrazy. Vytvoř si i proměnnou `pi` s hodnotou `3.14`.

Možné řešení:

```
>>> pi = 3.14
>>> polomer = 5
>>> obvod_kruhu = 2 * pi * polomer
>>> obsah_kruhu = pi * polomer * polomer
>>> obvod_kruhu
31.400000000000002
>>> obsah_kruhu
78.5
```

Cílem matematických úloh v tomto předmětu není zkoušet znalost vzorců, proto žákům v úlohách napovídáme. Očekáváme však, že vzorec dokážou zapsat v jazyce Python.

V jazyce Python se pro zápis desetinných čísel nepoužívá desetinná čárka, ale tečka. Pokud chceme do proměnné `pi` přiřadit hodnotu 3,14, zapíšeme `pi = 3.14`. Kdybychom použili příkaz `pi = 3,14`, jeho vykonání neskončí chybou, ale vytvoří se tzv. n-tice obsahující čísla `3` a `14`, která se přiřadí do proměnné `pi`:

```
>>> pi = 3,14
>>> pi
(3, 14)
```

S proměnnou `pi` (obsahující nesprávně n-tici) by bylo možné dokonce dále pracovat, například ji vynásobit celým číslem, aniž by vykonání příkazu skončilo chybou. V našem případě by to mohlo vypadat následovně:

```
>>> pi = 3,14
>>> polomer = 5
>>> obvod_kruhu = 2 * pi * polomer
>>> obvod_kruhu
(3, 14, 3, 14, 3, 14, 3, 14, 3, 14, 3, 14, 3, 14, 3, 14, 3,
14, 3, 14)
```

Python v tomto případě zdesetinásobil počet prvků dané n-tice a jako prvky použil hodnoty doposud v n-tici obsažené, tj. čísla `3` a `14`.

Následky použití čárky v zápisu desetinného čísla žákům nevysvětlujeme, neboť by problematice prozatím neporozuměli. Pokud však některý žák omylem čárku použije, pomůžeme mu nalézt chybu a vysvětlíme mu, že Python kvůli zápisu čárky namísto tečky neporozuměl správně jeho záměru.

V poslední úloze je potřeba diskutovat o tom, proč jsou názvy proměnných správně, nesprávně, resp. nevhodně zvolené. Případně je možné nechat žáky navrhnout vhodný název:

17. Diskutuj se svým spolužákem, které z následujících výrazů mohou nebo nemohou být názvy proměnných. Poté své domněnky ověř – zkus vytvořit proměnné odpovídajících názvů a přiřadit do nich nějaké hodnoty:

```
kuk
Ahoj!
1.A
prvni_trida
cerno-bile
OK
o0o0o0o
asdf
věk
počet osob
trida(3)
```

Pomůcky k diskuzi:

`kuk`	... v pořádku
`Ahoj!`	... nesprávný název, obsahuje vykřičník – vhodný název by byl `Ahoj`
`1.A`	... nesprávný název, začíná číslicí a obsahuje tečku vhodný název by byl: `_1_A`, `A1`, nebo `trida_1A` apod.
`prvni_trida`	... v pořádku
`cerno-bile`	... nesprávný název (Python to pochopí jako rozdíl dvou proměnných) – vhodný název by byl `cerno_bile`
`OK`	... v pořádku
`o0o0o0o`	... v pořádku, ale je špatně čitelný (nedoporučujeme kombinovat `o`, `0`, `1`, `l`, apod.)
`asdf`	... v pořádku, ale nepoznáme význam proměnné
`věk`	... v pořádku, ale diakritika se nedoporučuje

`počet osob` ... nesprávný název, obsahuje mezeru – vhodný název by byl `pocet_osob`

`trida(3)` ... nesprávný název, obsahuje závorky – vhodný název by byl `trida_3`

Programátoři v jazyce Python mají dohodu, že názvy proměnných obsahují jen malá písmena. Velká písmena se používají při definovaní typů. Pro potřeby žáků je to jedno, ale je-li to možné, je vhodné je vést k takovéto konvenci.