Opakování - Co už umíme

- Používat Python jako kalkulačku: `((12 + 34)*213)**(-2)`
- Přiřadit do proměnné: `a = 12 + 34`
- Porovnávat: `12 > a is False`
- Indexovat řetězce: `"Ahoj"[0:3:2] == "Ao"`
- Spojovat řetězce: `"Ah" + "oj" == "Ahoj"`
- Některé užitečné funkce:
  - `print()` - výpis
  - `type()` - typ
  - `len()` - délka
  - `str()`, `int()`, `float()` - konverze

Základní tipy k Jupyter Notebooku

Edit mode (v nějaké buňce bliká kurzor)

- Jak se do něj dostat: klikněte do buňky/zmáčkněte enter
- šipky posouvají kurzor
- "tab" (↹) doplní načaté slovo nebo nastaví odsazení (indent)
- "ctr + /" comment/uncomment

Command mode (v žádné buňce kurzor nebliká)

- Jak se do něj dostat: zmáčkněte esc
- šipky (↑↓) vybírají další buňky
- "a" přidá prázdnou buňku nad
- "b" přidá prázdnou buňku pod
- "dd" smaže aktuální buňku

Vykonání instrukcí v buňce (funguje v obou módech):

- ctrl + enter (vykoná)
- alt + enter (vykoná a vloží prázdnou buňku pod tu aktuální)
- shift + enter (vykoná a vybere buňku pod tou aktuální. Není-li co vybrat, přidá prázdnou)

Základy programování v jazyce Python

Lekce 3 - seznamy, cykly

Motivace - Seznamy můžeme měnit

Řetězec - indexovatelný, ale neměnný:

```
"Pzthon"[1] = "y"
```

```
TypeError
```

Seznam (list):

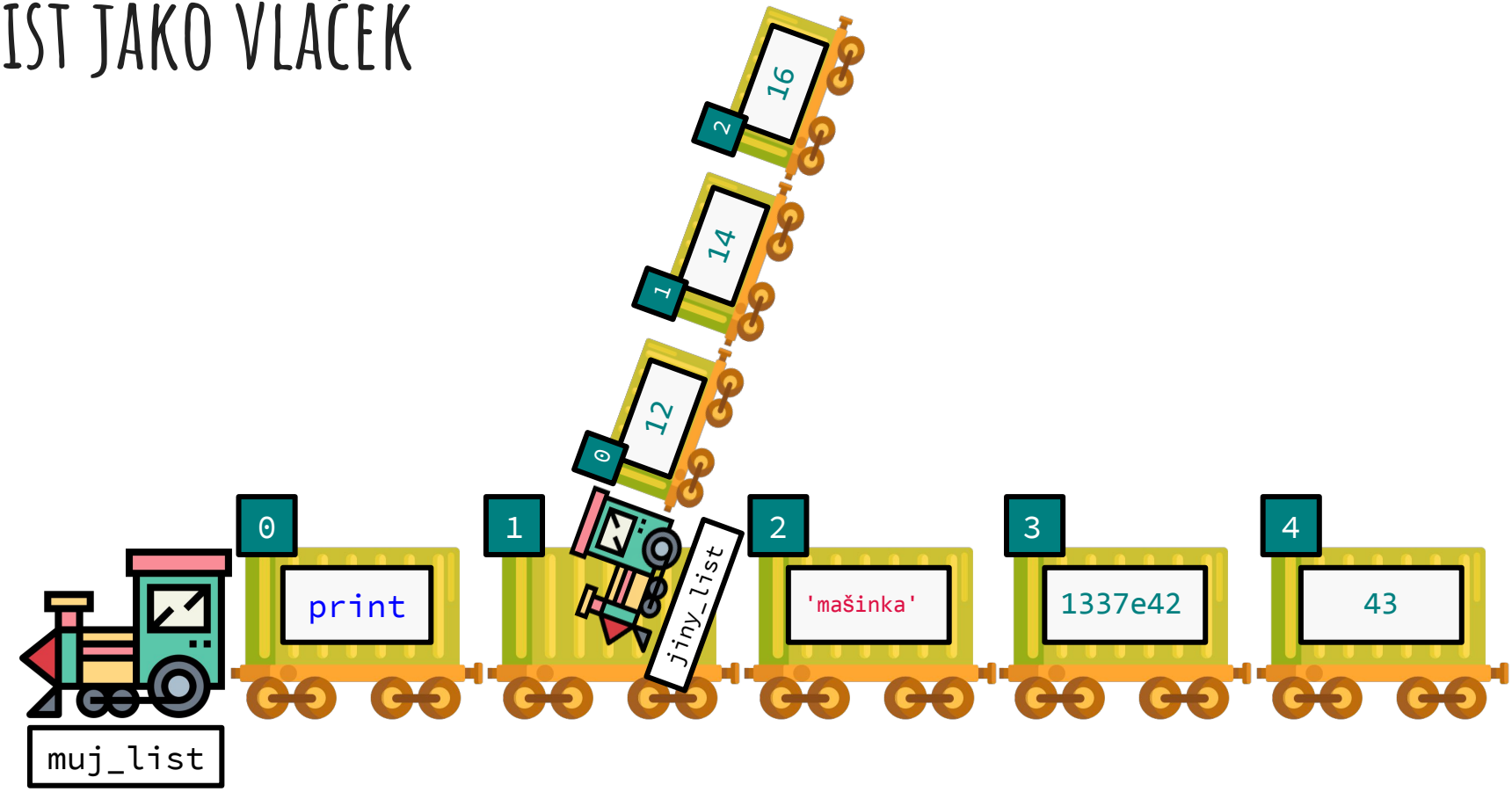
```
["P", "z", "t", "h", "o", "n"][1] = "y"
```

```
["P", "y", "t", "h", "o", "n"]
```

Do seznamu můžete dát cokoliv

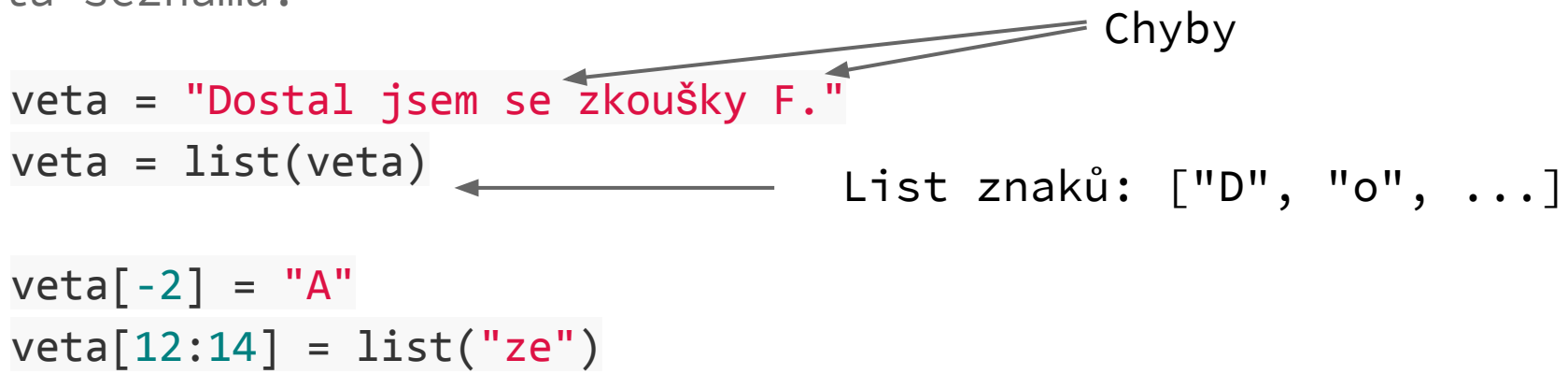
```
tisk = print # přiřadil jsem funkci print proměnné tisk (zkuste jak to funguje)
cele_cislo = 43
cislo = 1337e42
text = 'mašinka'
jiny_list = [12,14,16]

muj_list = [ tisk, jiny_list, text, cislo, cele_cislo]

zkuste si a vysvětlete:
 -  print(muj_list[0])
 -  muj_list[0](muj_list)
 -  muj_list[1][2]
```

List jako vláček



Indexování s přiřazením

Všechno, co platí pro indexování řetězců, platí pro indexování seznamů plus můžeme přirazovat na indexovaná místa seznamu:



Všimněte si:

- veta[0] je **string** "D"
- veta[0:1] je **list** obsahujici "D", t.j. ["D"]

Cvičení

Změňte list a přeložte následující větu do češtiny:

```
v = ["Smädný", "korčuliar", "zízal", "na", "cencúľ"]
```

Přitom použijte následující proměnné:

```
a = ["Žíznivý", "bruslař"]
b = "zíral"
c = "rampouch"
```

Objekty mají svoje nástroje - Metody

Metoda je funkce svázaná s objektem.

Syntax (bez mezer):

```
objekt . metoda ( argumenty )
```

V lidské řeči znamená:

> *Z (=tečka .) objektu '**objekt**' zavolej (=závorky ()) metodu '**metoda**' s argumenty '**argumenty**'.*

Metody mohou ale nemusí změnit objekt a mohou vracet hodnotu (jinou než **None**).

Co ještě seznamy umí - metody třídy list

- append
- count
- extend (podobný `+`)
- index
- insert
- pop
- remove
- reverse
- sort

Příklad:

```
a = [ 3 , 2 ]
a.append(1)
print(a)

a.sort()
print(a)
```

Nezapomeňte na závorky!

Tip - pomoc bez Googlu

Vyzkoušejte v notebooku: `?list.reverse`

Vrácení výsledku vs. změna stavu objektu

```
a = [1, 3, 2, 1, 1]
vysledek_sort = a.sort()
print(vysledek_sort) #None ..... sort() nic nevrací
print(a) # vypíše [1, 1, 1, 2, 3]  zato změnil stav seznamu "a"
vysledek_count = a.count(1) ..... count() spočítá výskyty
print(vysledek_count) #3 ..... a výsledek vrátí
```

Cvičení

```
a = ['b','c','e']
b = ['d','a']
```

1. Spojte seznamy pomocí operátoru `+`
2. Seřaďte výsledek pomocí metody `sort`

Můžete ještě zkusit spojit pomocí:

- metody seznamu `extend`
- indexování, t.j: `a[?:?] = b`

Hledání v Seznamu

Podobně jako u řetězců i u seznamů lze použít operátor **in**:

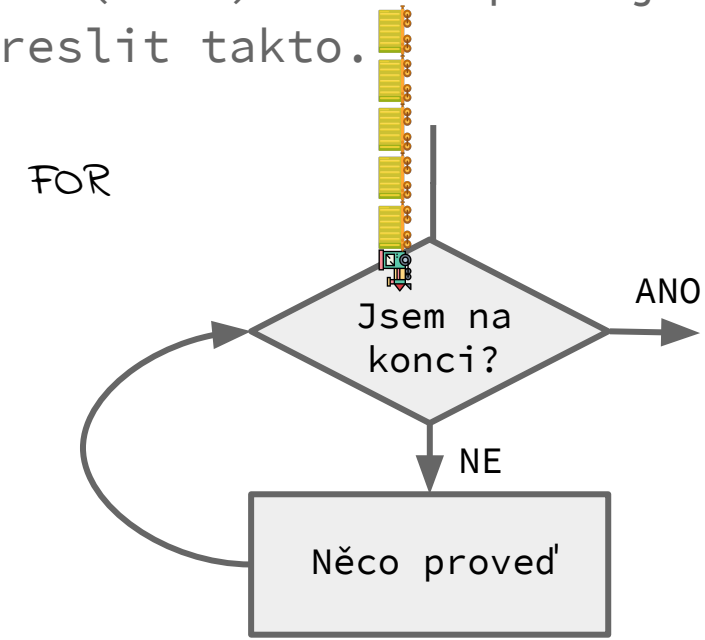
```
les = ["lípa", "topol", "bříza", "metasekvoje", "borovice"]
print("topol" in les) #True
print("sekvoje" in les) #False
```

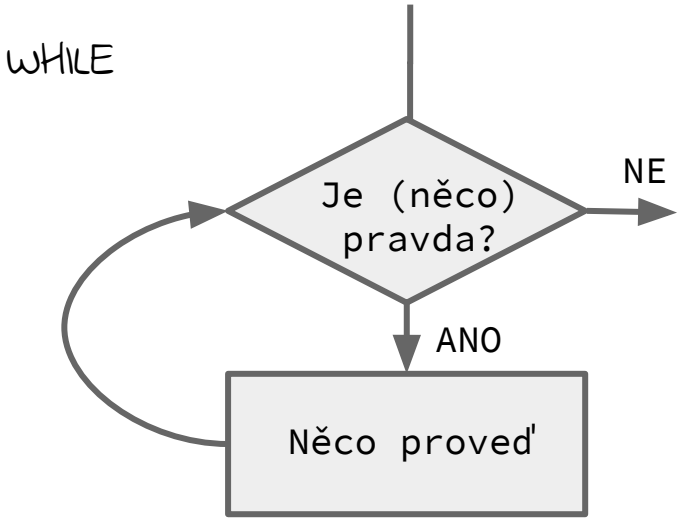
Pro zjištění kde se něco nachází je metoda **index**:

```
les.index("topol") #1
```

For a While cyklus

For a while lze volně přeložit jako pro (každý prvek) a dokud (něco). Pokud použijeme vývojový diagram můžeme je zakreslit takto.





Cyklus while

while překládáme jako **dokud** (je něco pravda, dělej tohle)

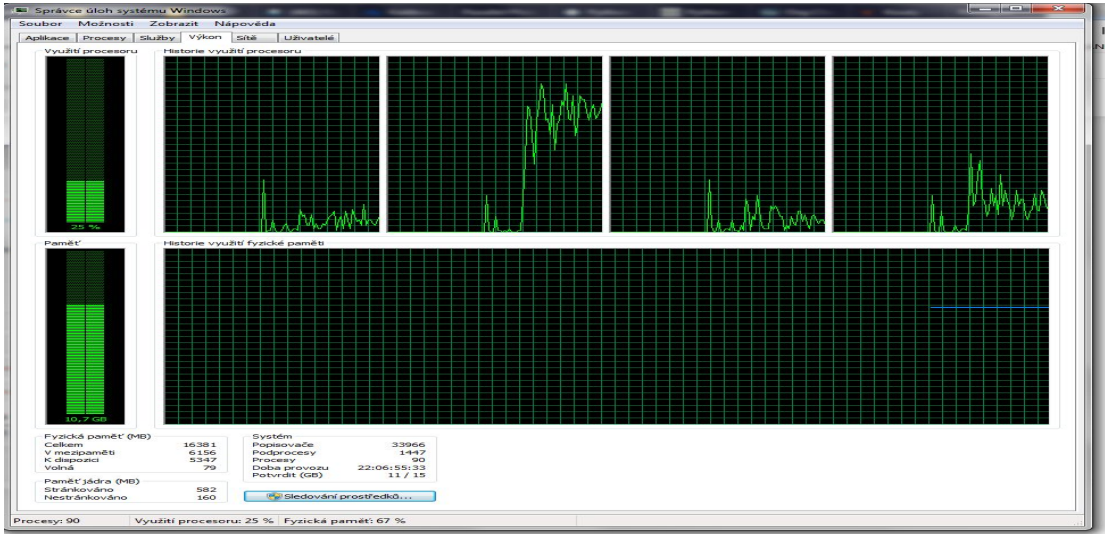
```
while not finished:
    run_further()
```

Nikdy nekončící while cyklus

```
i = 1
while i < 100:
    b = i+1
```

podmínka je splněna vždy

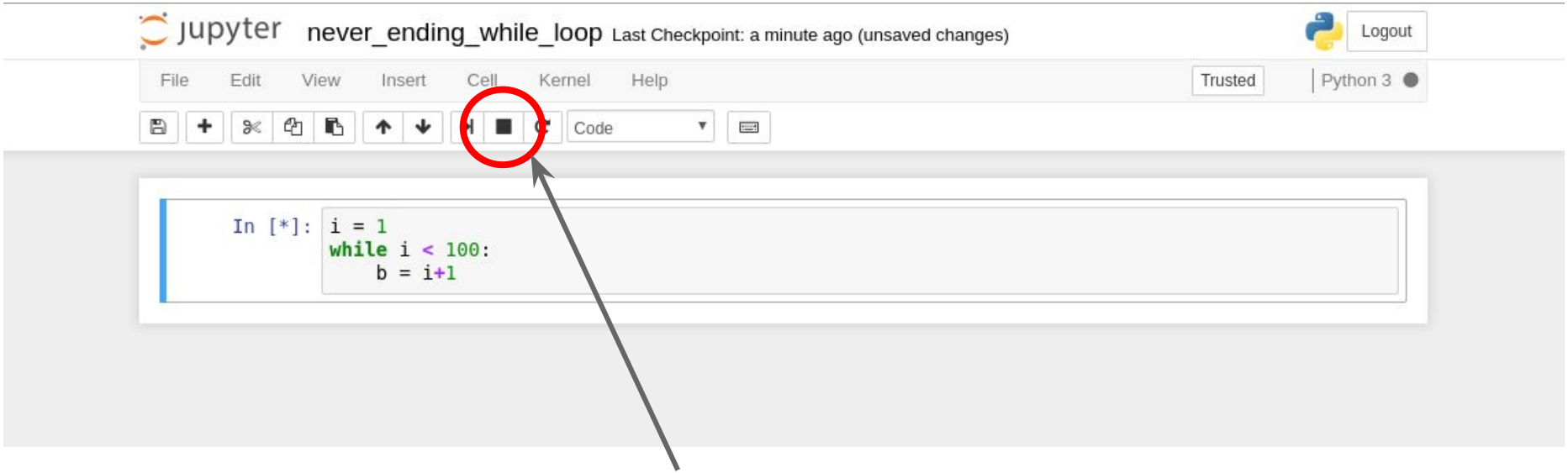
i se vůbec nemění!



výkon procesoru stoupá a drží se proklatě vysoko

Nikdy nekončící while cyklus

Nikdy nekončící while cyklus - první pomoc



Chytíte-li se do nekonečné smyčky, klikněte na černý čtvereček.

Cyklus while - syntax

```
while podminka:
    vykonej_neco
kod_po_cyklu
```

Indentace →

Mimo cyklu bez indentace →

"indentace" by správně byla "odsazení". Cizí výraz používáme proto, abyste si rozuměli s počítačem, až Vám nahlásí chybu.

Cyklus while v praxi

Tentokráte se vracíme z Pradědu do Kout, vzhledem k značnému množství zkonzumované čokolády jsme dokonalé koule a valíme se dolů z kopce se zrychlením. Za jak dlouho dorazíme do Kout? Jaká bude naše rychlost?

```
vzdalenost = 11000 # m
rychlost = 0 # m/s
zrychleni = 0.15 * 9.81# m/s**2
pozice = 0 # m
cas = 0 # s
```

Cyklus while v praxi

Tentokráte se vracíme z Pradědu do Kout, vzhledem k značnému množství zkonzumované čokolády jsme dokonalé koule a valíme se dolů z kopce se zrychlením. Za jak dlouho dorazíme do Kout? Jaká bude naše rychlost?

```
vzdalenost = 11000 # m
rychlost = 0 # m/s
zrychleni = 0.15 * 9.81# m/s**2
pozice = 0 # m
cas = 0 # s
```

```
while pozice < vzdalenost:
    pozice = pozice + rychlost
    rychlost += zrychleni
    cas += 1
else:
    print('Do kout dorazíme za ',cas, 's')
    print('Naše rychlost bude',
          rychlost,'m/s')
```

Cyklus while v praxi

Vzdálenost z Prahy do Olomouce je přibližně 250 km.

V 6.00 vyjel z Prahy do Olomouce rychlík rychlostí 85 km/h.

Ve stejném okamžiku mu vyjel naproti z Olomouce osobní vlak rychlostí 65 km/h.

Za jak dlouho se **přibližně** vlaky setkají?

Przykład polski

```
jmena = ['Zbigniew','Kazimierz',
         'Wojciech']

while a:
    print(a.pop(-1))

print(a)
```

- co dělá metoda .pop(-1) ?
- je prázdný list pravda nebo lež ?
- co se stane, když místo "-1" dáme jiný index?
- co se stane, když zavoláme .pop() bez argumentů?

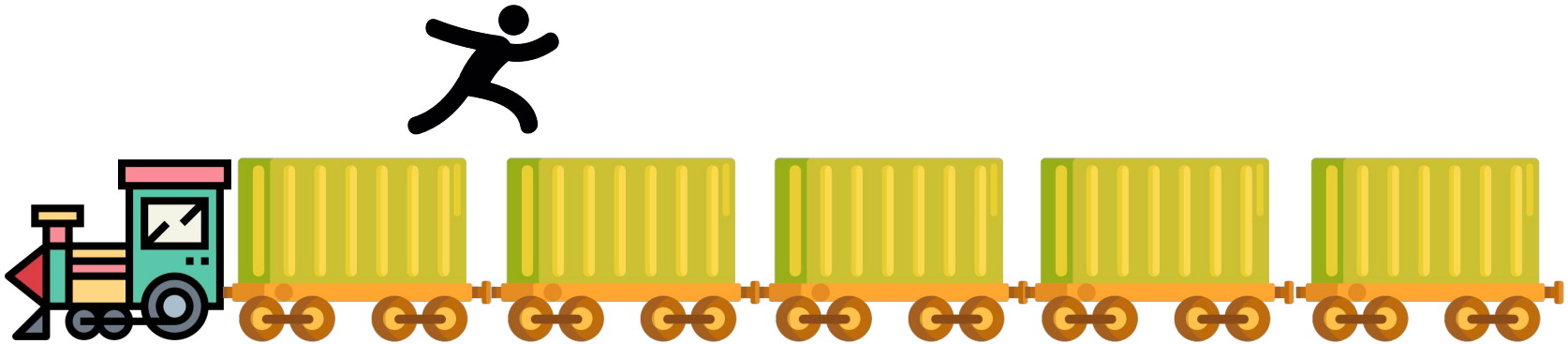
Abulafiova Extáze

Abraham Abulafia propagoval laciný způsob, jak dosáhnout extáze: kombinujte písmena, z nichž je složeno jméno Boží, dokud to na Vás nepřijde. Zkuste, zda extáze může dosáhnout i Váš počítač. Napište cyklus while tak, aby přestal, když (jmeno_bozi == extaticka_kombinace):

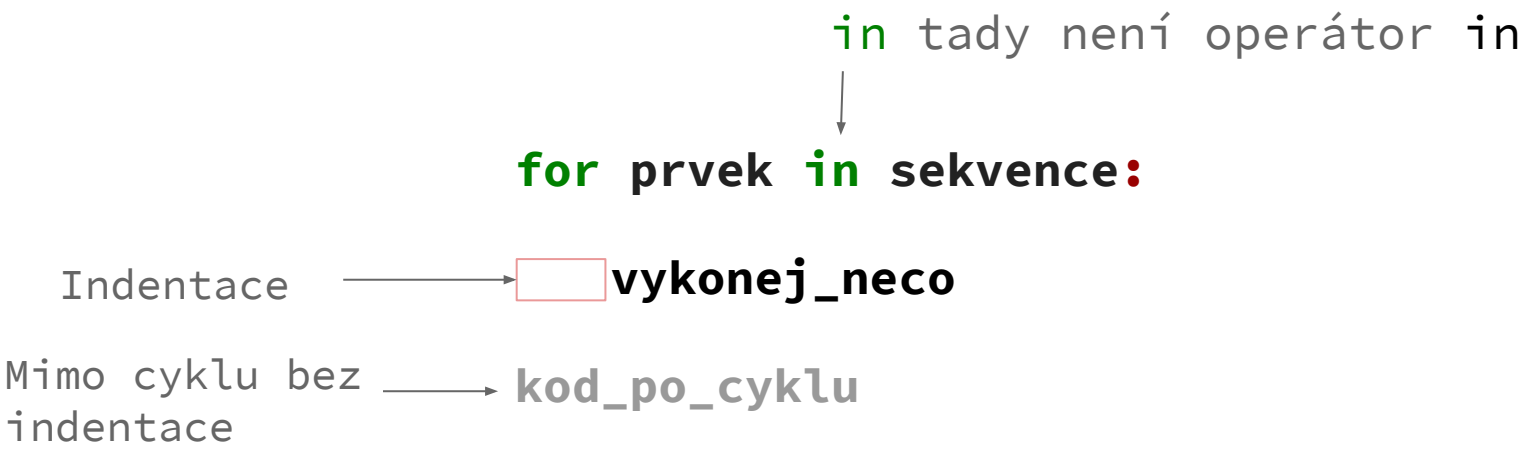
```
import random
jmeno_bozi = ['j', 'a', 'h', 'v', 'e']
extaticka_kombinace = ['v', 'e', 'j', 'h', 'a']
random.shuffle(jmeno_bozi)
print(jmeno_bozi)
```

Cyklus for

for lze volně přeložit jako **pro** (každý prvek něco udělej)



Cyklus for - syntax



"indentace" by správně byla "odsazení". Cizí výraz používáme proto, abyste si rozuměli s počítačem, až Vám nahlásí chybu.

Cyklus for - jenom opakování kódu

```
a = [ '1', '2', '3' ]
k = 0
print(a[k])
k = k + 1
print(a[k])
k = k + 1
print(a[k])
```

```
a = [ '1', '2', '3' ]
for i in a:
    print(i)
```

Možnosti iterace

```
les = ["lípa", "topol", "bříza", "metasekvoje", "borovice"]

for stromek in les:
    print(stromek)

cislo = 1
for stromek in les:
    print("Strom číslo " + str(cislo) + " je " + stromek)
    cislo +=  1
```

Možnosti Iterace - Funkce range

```
cislo = 1
for stromek in les:
    print("Strom Číslo " + str(cislo) + " je " + stromek)
    cislo = cislo + 1
```

Trochu kostrbaté

```
for i in range(len(les)):
    print("Strom Číslo " + str(i+1) + " je " + les[i])
```

Měsíce - zadání ve studijních materiálech

Zdroj xkcd