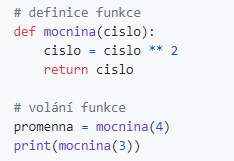
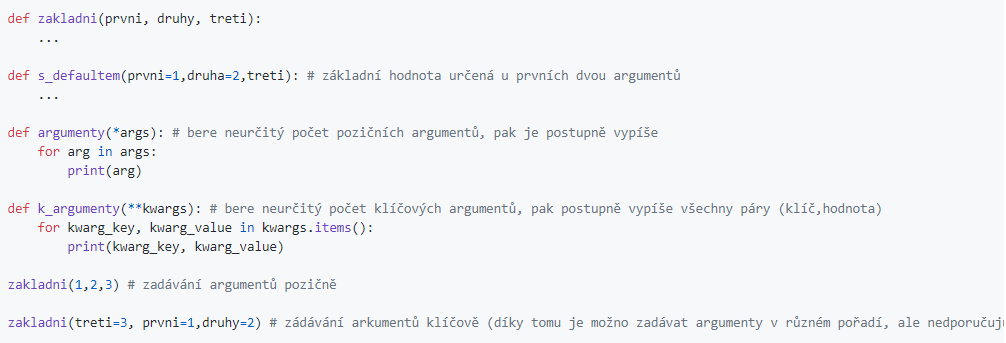
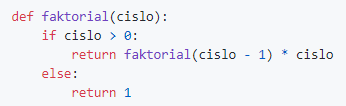
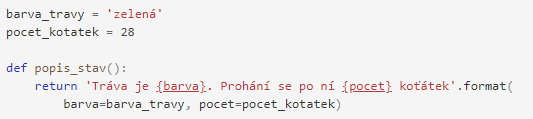
**14. PRG – Funkce, moduly**

Funkce:

* funkce se používají, pokud nechceme psát program imperativně (provede se odshora dolů), ale chceme si program rozdělit na menší úseky - funkce
* tato metoda se nazývá funkcionální styl programování
* funkce se dá v programu opakovaně volat a za pomoci vstupních argumentů i na různých místech diverzifikovat její výstup (znovupoužitelnost a přehlednost)
* funkce obvykle přijímá argumenty (data, která zpracuje) a něco vrací – výslednou hodnotu; funkce taky nemusí vracet nic (př. funkce print())
* Syntaxe:
  + def – syntax značící definici funkce
  + mocnina – název funkce
  + hodnoty v závorce (*cislo*) – Parametry funkce, které se zadávají při volání funkce; může jich být samozřejmě více
    - Povinné – když je uživatel nedosadí, tak funkce neproběhne a program vyhodí chybu (v příkladu je to *cislo*)
    - Implicitní (defaultní) – parametry, které jsou již předdefinované, ale dají se při volání funkce změnit, např. *def fce(cislo, n=1)*
    - Nepovinné – např. *def fce(\*y)*, y bude n-tice o libovolné délce, tzn. nemusí se zadat žádné parametry ale taky hodně úplně
  + return – (návratová) hodnota, kterou funkce vrátí – může se uložit do proměnné, nebo vytisknout; pokud ve funkci chybí return a například jen něco tiskneme, tak se první vytiskne hodnota None, poté to co chceme, většinou se píše na konec, pokud return napíšeme někam doprostřed tak se funkce přeruší (prakticky to funguje jako break), může být ve funkci i vícekrát, ale pak je vhodné to nějak upravit pomocí *if*
  + celou funkci lze ukončit příkazem break
* Argumenty:
  + argumenty jsou hodnoty, které zadáváme při volání funkce, poté se přiřadí parametrům
  + argumenty musíme určit v definici funkce
  + při volání můžeme argumenty zadat 2 způsoby:
    - poziční: nezadává se jméno argumentu, ale pouze se vkládá hodnota na správnou pozici
    - klíčové: zadává se název argumentu a jeho určená hodnota
  + pokud má funkce přijmout neurčitý počet argumentů nebo klíčových argumentů, tak se používá syntaxe '\*' a '\*\*', tyto znaky se napíší před název proměnné. Do proměnné se poté uloží argumenty ve formě n-tice u pozičních a ve formě slovníku u klíčových
* Rekurze:
  + rekurze označuje, když funkce volá sama sebe, to se využívá například pro výpočet faktoriálu
  + při rekurzi se musí dávat pozor, aby se funkce někdy ukončila, jinak program spadne na přetečení zásobníku
* Lokální a globální proměnné:
  + lokální proměnné vznikají ve funkci, tzn. jsou použitelné pouze v té samotné proměnné, tzn. můžeme mít v programu třeba dvě proměnné *x*, jednu ve funkci a druhou mimo ni a jedná se o dvě nezávislé proměnné
  + pokud ve funkci ale použijeme *global x*, tak se sjednotí a bude existovat již pouze jedna proměnná *x*
* Volání funkce komponentou:
  + knihovna Tkinter, komponentou je třeba tlačítko
  + ke komponentě doplníme *command*, a do něho dáme pouze název (adresu) funkce (ne závorky, ne parametry), tím se fce spustí až poté, co se na tlačítko klikne
* Lambda funkce:
  + jednoduše to je definice funkce na jediném řádku
  + oproti normální funkci jí chybí název, nedá se zavolat
  + výhodou je šetření místa, nevýhodou horší přehlednost
  + používá se pro krátké, jednorázové operace, např. v Tkinteru

Moduly:

* modul je v pythonu něco, z čeho můžeme importovat, například knihovny (moduly) random, math…, tzn. pokud máme třeba vlastní funkci, která je využitelná pro více programů tak ji nemusíme psát dvakrát, ale stačí ji z nějakého modulu (knihovny) zavolat
* z modulu lze importovat jak jednotlivé funkce, např. *from random import randint*, pokud se importuje *from random import \**, tak může nastat problém kolize modulů (pokud takhle importujeme více modulů, tak se může stát, že mají stejně pojmenované funkce; potom se používá funkce té knihovny, která se importuje poslední a z té první knihovny je tato funkce nedostupná); tak i celý modul najednou, např. *import random*, nebo *import random as rnd*, to se potom využívá *rnd.randint()*
* python má své základní moduly, které jsou součástí základní instalace pythonu, např. math, random, sqlite, tkinter…
* moduly, které nejsou v základu součástí pythonu, např. pygame, pylag (matplotlib)… lze přidat použitím příkazu pip v cmd
* uživatel si může také vytvářet své vlastní moduly, ze kterých lze poté importovat
* Příklad: vytvořený modul louka.py obsahuje funkci pro výpis věty, tento modul si importujeme do jiného souboru, např. vypis.py a využijeme funkci popis\_stav() z modulu louka k vypsání *vypis.py*

*louka.py*