# Poznamky Python

## String indexing and Slicing notation

Retezce v Pythonu jsou indexovany. To znamena, ze kazdy znak v retezci ma ciselnou hodnotu (index), ktery jej reprezentuje. Pocatecni hodnota je 0 (-0=0). Kazdy dalsi znak ma hodnotu +1. Vzhledem k tomu, ze se zacina od 0 je posledni znak o 1 mensi, nez je pocet znaku. Tedy slovo ‘Python’ ma 6 znaku, ale index posledniho je 5. Rozepsano to vypada takto:

**P**=0, **y**=1, **t**=2, **h**=3, **o**= 4, **n**=5

Je mozne pocitat od zadu, pak ma index hodnotu **-**. Rozepsano to vypada takto:

**P**=-6, **y**=-5, **t**=-4, **h**=-3, **o**= -2, **n**=-1

**Priklad:**

Vytvorim promenou word, ktera obsahuje slovo Python  
*word = ‘Python’  
word[0]* # vrati znak na indexu 0, tedy ‘P’  
*word[5]* # vrati znak na indexu 5, tedy ‘n’  
*word[-1]* # vrati znak na indexu -1, tedy ‘n’  
*word[-6]* # vrati znak na indexu -6, tedy ‘P’

Navic je mozne ziskavat fragmenty zadaneho retezce “urezavanim” jeho casti. Toto je v Pythonu nazyvano “slicing notation”.

**Priklad:**

*word[0:2]* # vrati vsechny znaky na indexu 0, 1 -> ‘Py’  
*word[2:5]* # vrati vsechny znaky na indexu 2, 3 ,4 -> ‘tho’

Posledni znak zamerne neni soucasti vystupu. Diky tomu plati, ze **s [:i]+s[i:] = s**

Stringy lze take scitat:

**Priklad:**

*word[:2] + word [2:]* #vystupem je ‘Python’, prvni retezec vraci znaky na idexu 0 a 1 a druhy retezec vraci znaky na indexu 2, 3, 4, 5

Pokud budete primo zadat o znaky na indexu, ktere string neobsahuje, Python vrati chybu. *word[42]* #python vrati ‘string index out of range’. Pokud ovsem napiseme slicing notaci takto *word[4:42]*, pak Python vrati vsechny znaky od indexu 4 do konce stringu, tedy ‘on’. Pokud napiseme slicing notaci takto *word[42:]*, pak Python vrati prazdny retezec ‘’.

Retezce v Pythonu jsou nemenne, proto kdyz se pokusime priradit novy znak do retezce na jiz existujici index, Puthon vrati chybu.

**Priklad:**

*word[0] = ‘J’* # vrati chybu, index 0 je jiz obsazen

Pokud potrebujeme-li jiny string, musime si ho vytvorit.

Pomoci metody ‘len’ je mozne ziskat pocet znaku, ktere string obsahuje.

**Priklad:**

*S = 'supercalifragilisticexpialidocious'  
len(s)* # vystup je cislo 34

## If statement

Klasicka podminka. Pouziva se, kdyz je potreba rozhodnout, jak pokracovat v programu dale.

Priklad:

*#Program na hadani spravneho cisla*

*hadane = 100*

*def h(vase):*

*if hadane == vase:*

*print ('uhadnuto')*

*elif hadane < vase:*

*print ('to je moc')*

*elif hadane > vase:*

*print ('to je malo')*

Funguje to tak, ze ‘if’ statement zpracuje podminku a rozhodne, zda se spusti kod primo pod ‘if’ (podminka splnena) nebo skoci do else (podminka nesplnena). Proto uz do else nepisme zadnou dalsi podminku, je to primo vztazene k podmince v ‘if’. Muzeme pridat dalsi podminky pro pripad, ze se muze stat vice scenaru pomoci ‘elif’. Pak ma kazdy ‘elif’ svoji podmiku. Jediny nutny statement je ’if‘, ‘else’ a ‘elif’ jsou optional.

## While statement

Smyslem ‘While’ cyklu je opakovat nejakou cinnost, dokud je splnena vstupni podminka.

Priklad:

*a = 0  
b = 1*

*while b < 10: #dokud cislo ‘b’ bude mensi nez 10  
 print b #vrat hodnotu cisla ‘b’  
 a,b = b, a+b*

**Dulezita poznamka:**

Kod  
*a,b=b,a+b*není indenticky s  
*a=b,  
b=a+b* V prvnim pripade Python vytvori promennou, ve které ulozi hodnotu ‘a’, ‘b’, ‘a+b’ a nasledne vse za print zpracuje. Nepracuje tedy ciste krok za krokem, ale rozlusti vse za printem a nasledne to provede naraz.  
Vysledkem je Fibonacciho posloupnost.

## For statement

Smyslem ‘For’ cyklu je opakovat nejako cinnost na predem znamych datech.

## Reading and writing files

Otevreni, cteni a zapis z textoveho souboru (\*.txt) zajistuje zakladni Python knihovna pomoci metod ‘open‘, ‘read‘, readline, write a close.

### 4.1) Metoda Open

Po zavolani metody open si Python vytvori object, ktery reprezentuje soubor. Aby bylo mozne s timto objektem pracovat, je potreba vytvorit promennou, na kterou se odkazuje. Metoda akceptuje dva argumenty, cestu k souboru a rezim prace. Rezim prace znamena ‘jen pro cteni’, ‘zapis’, ‘append’ (pridani textu na konec souboru), a ‘r+’ (cteni a zapis soucasne). Argument rezim prace je nepovinny a pokud neni zadan je za vychozi rezim povazovan ‚r‘ (read only). V prikladech nize je uvedena promenna ‘F’.

**Dulezite upozorneni:**

Oficialni dokumentace Pythonu popisujici metodu ‘open’ uvadi jako prvni argument cesty k souboru retezec. *f = open('workfile', 'w')* Takovyto zapis se nebude snazit najit soubor na disku!

**Priklad:**

*F = open (C:\Users\Z0001932\Desktop\python\pika.txt, ‘r’)*

#do promenne ‘F’ ulozi objekt reprezentujici textovy soubor ‘pika.txt v read only rezimu

### 4.2) Metoda Read

Metoda ‘read’ nacte cely obsah souboru do promenne ‘F’ reprezentujici soubor. Metoda vraci retezec. Volitelny argument je velikost. Tato promenna je ulozena v RAM pocitace. Python nehlida, jestli nacitany soubor neni vetsi, nez dostupna pamet RAM. Proto je jistejsi se pouziti teto metody vyhnout.

**Priklad:**

f.read()

### 4.3) Metoda Readline

Metoda ‘readline’ nacte do promenne ‘F’ retezec znaku z jedne radky souboru. Na konec radky prida znak ‘\n’. Tento znak je vynechan pouze tehdy, kdy nova radka neexistuje = konec souboru. To znamena, ze tato metoda ma nejednoznacny vystup.

**Priklad**:

f.readline() -> ‘This is the first line of the file. \n’  
f.readline() -> ‘’ #takhle vypada konec souboru

### 4.4) Metoda Write

Metoda ‘Write’ zapisuje data do souboru, nema zadny vystup. Pokud se zapisuje ciselna hodnota, musi nejprve byt prevedena na retezec.

**Priklad**:

Value = (‘the answer’, 42)  
s=str(values)  
F.write(s)

### 4.5) Metoda Close

Metoda ‘Close’ slouzi k ukonceni cinnosti nad souborem a uvolneni systemovych prostredku.

**Priklad:**

f.close()