

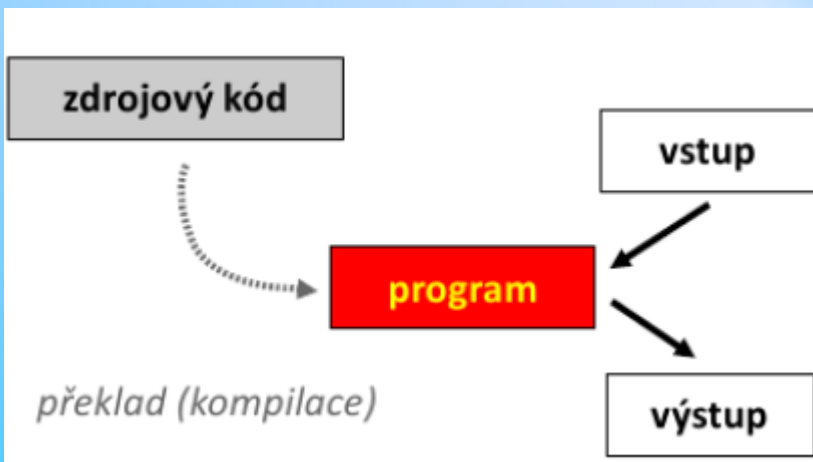
Programovanie v PYTHONE

Veronika Dillingerová

Programy vs Skripty

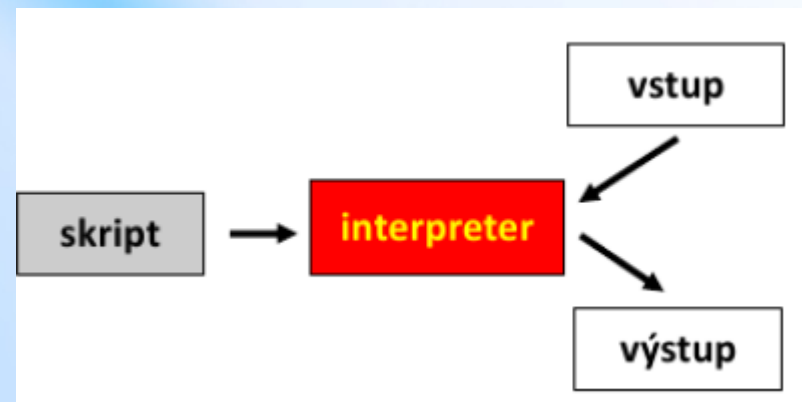
PROGRAM

- súbor strojových inštrukcií spracovávaných priamo procesorom.
- vzniká prekladom zdrojového kódu programovacieho jazyka.
- **Prekladané jazyky:**
 - C/C++
 - Fortran



SKRIPT

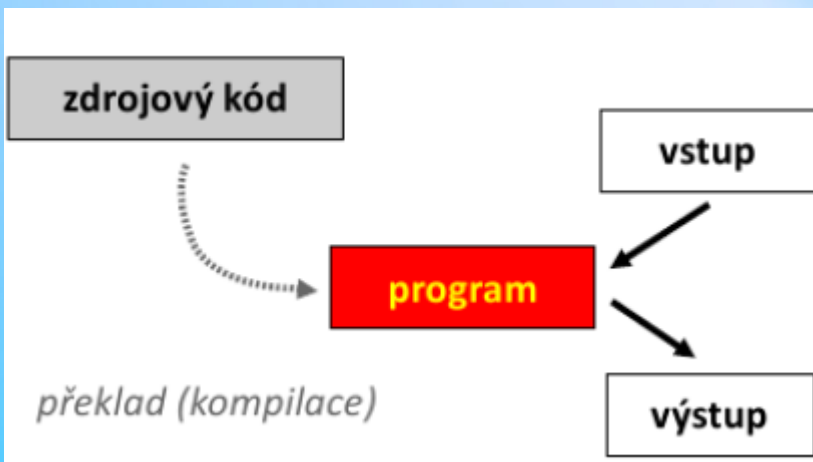
- textový súbor obsahujúci príkazy a riadiace sekvencie, ktoré sú vykonávané interpretrom použitého skriptovacieho jazyka.
- **Skriptovací jazyky:**
 - bash
 - gnuplot
 - awk
 - JavaScript
 - PHP



Programy vs Skripty

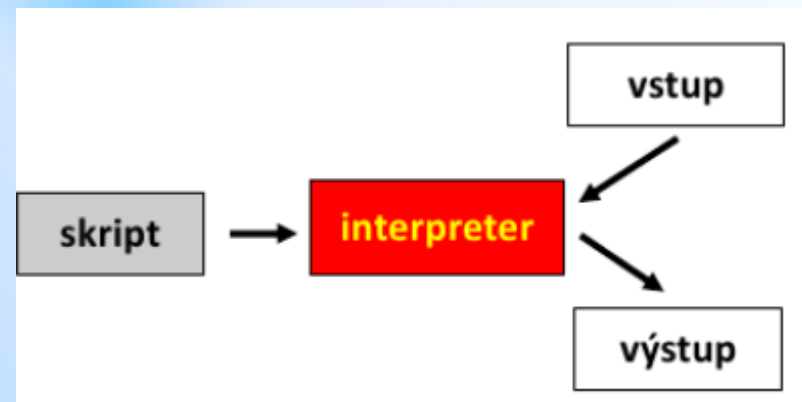
PROGRAM

- ľahká optimalizácia
- rýchle vykonávanie
- nutnosť rekompilácie
- nedá sa vytvárať samospustiteľný kód

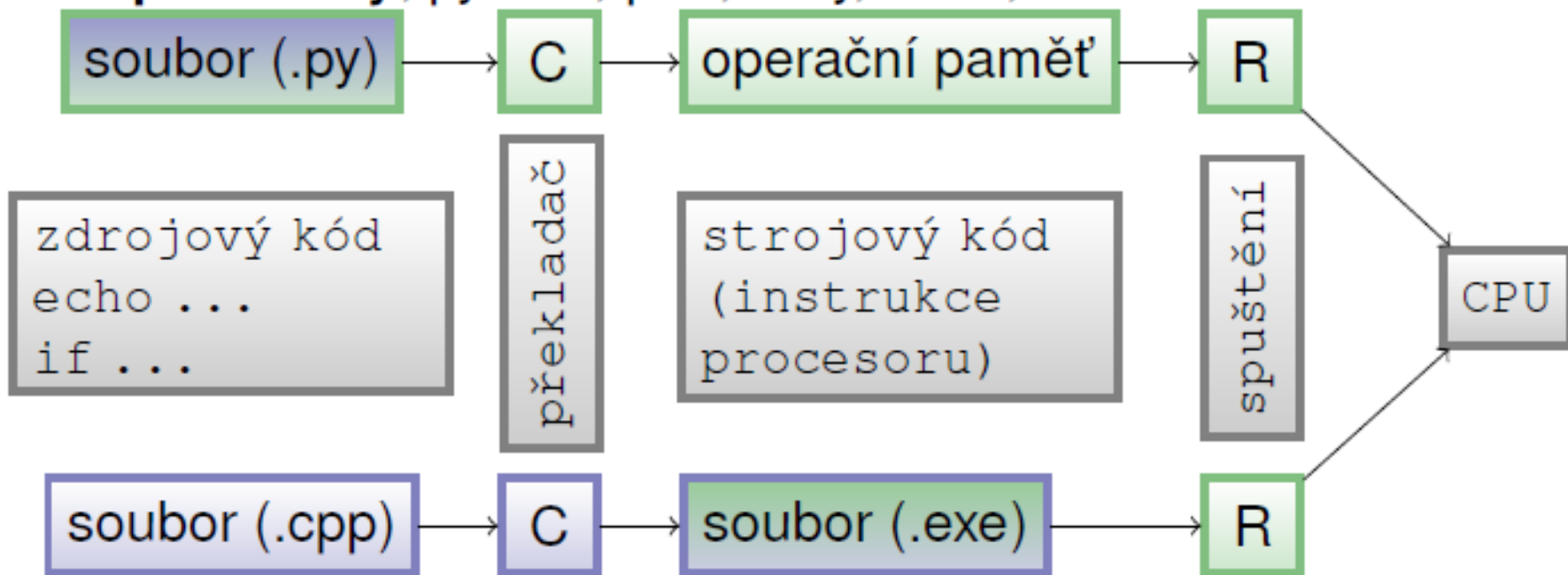


SKRIPT

- nevyžaduje rekompiláciu
- vytváranie samospustiteľného kódu
- zlá optimalizovateľnosť
- pomalšie vykonávanie



interpretovaný; python, perl, ruby, bash, ...



kompilovaný; C, C++, C#, fortran, ...

(zelená barva označuje kroky, které běží u uživatele)

Kompilované programovacie jazyky

- (napr. Pascal, C, C++)
- pred spustením sú najprv kompletne preložené kompilátorom
- výsledkom je väčšia rýchlosť, ale tiež väčšia náročnosť na správne zapísaný kód

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    printf("Tohle je program v jazyce C! \n");
    return(0);
}
```

Interpretované programovacie jazyky

- (napr. BASIC, Perl, Python, shell)
- interpretované jazyky, ktoré sa iba interpretujú (z toho dôvodu sú pomalšie)
- interpretované jazyky, ktoré sa prekladajú, ale iba do medzikódu, nie do strojového kódu počítača (napr. Java, Python)

Interpretované jazyky

- ktoré sa po spustení za behu programu prekladajú do strojového kódu počítača
- (napr. Java, pokiaľ sa použije systém JIT)

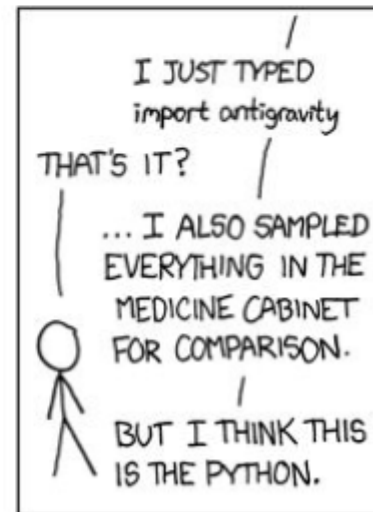
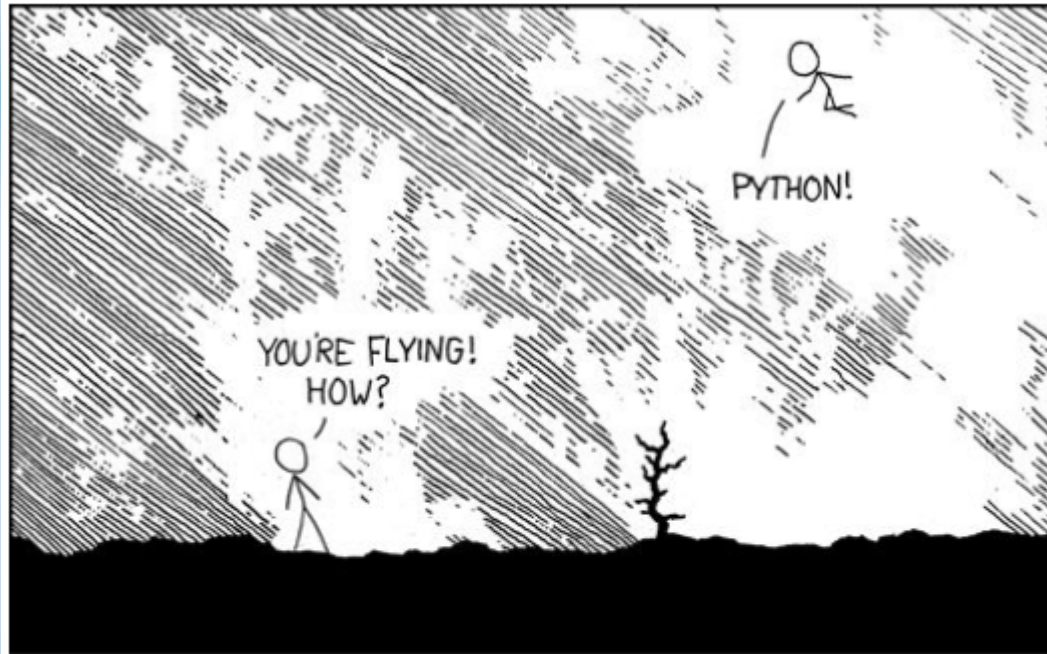
Skriptovacie jazyky

- špecializované k riešeniu určitých problémov
- Awk, Perl, html, Php, VBA, Bash

```
#!/bin/bash
```

```
echo 'Toto je skript v interpretu Bash!'
```

PYTHON



Volně dostupné výukové učebnice Pythonu

- <http://www.root.cz/knihy/oddeleni/vyvoj-a-knihypro-vyvojare/>
- **Think Python** - Allen Downey (en)
- **Učebnice jazyka Python** – Jan Švec

Vlastnosti jazyka

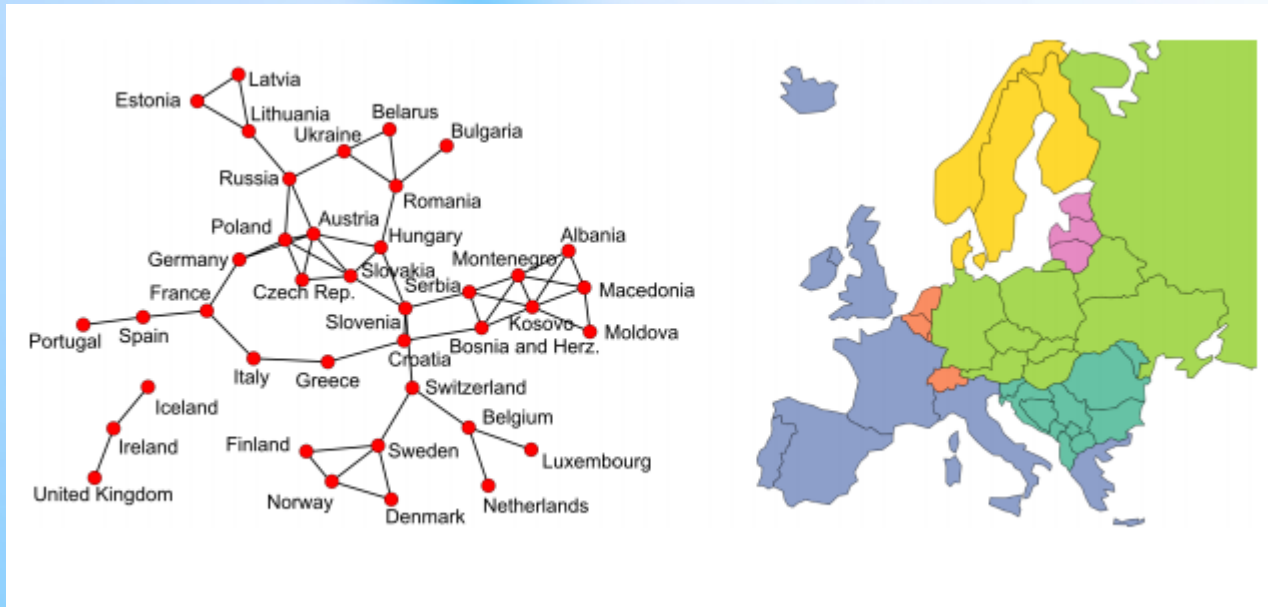
- **vysoko-úrovňový** – veľká miera abstrakcie
- **interpretovaný** programovací jazyk ("skriptovací jazyk")
- dynamicky typovaný programovací jazyk
- voľne a ľahko dostupný na všetkých platformách
- široká nabídka knihooven

Využití Pythonu

- **skriptovanie**
- **vedecké výpočty (chemoinformatika, bioinformatika, ...)**
- webové aplikácie
- administrácia
- grafika
- audio
- networking
- hry
- aplikácie pre chytré telefóny
- ...

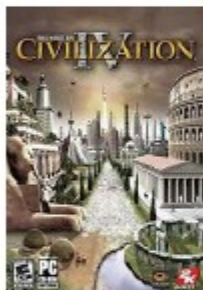
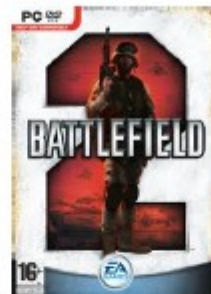
Analýza dát

Python, NumPy, SciPy, Pandas, matplotlib,
Kartograph, networkx, . . .



Firmy používající PYTHON

Google



Dropbox

SurveyMonkey



Pinterest



mozilla
FOUNDATION

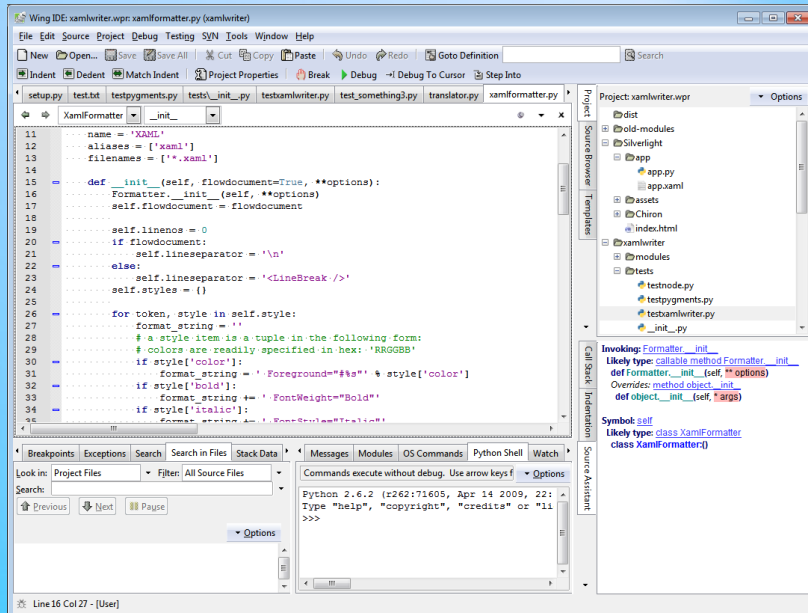


You Tube

Pracovné prostredie

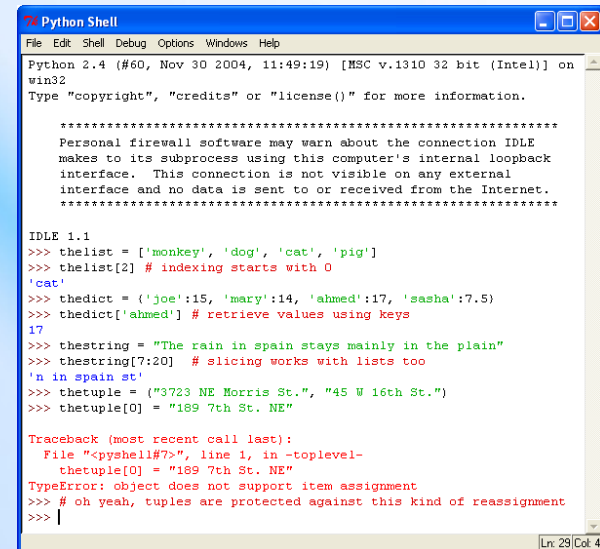
IDE

- IDE – vývojové prostredie
- IDLE, Wing, PyStudio, Eclipse ...



Shell

- Prostredie, kde sa vykonávajú príkazy priamo po napísaní
- Príkazový riadok



Hello world!

```
print ("Hello World! ")
```

1. Zadajte príkaz priamo do shellu
2. Vytvorte súbor MyFirstSkript.py a spustite ho pomocou F5 v IDLE (Python GUI)



Premenná (variable)

- má svoj názov (identifikátor, identifier): *a*
- odkazuje na miesto v pamäti, kde je uložená tzv. hodnota (value) premennej, prepojenie premennej a jej hodnoty robíme pomocou priradenia (assign) (=)

`a = 1`

- Python je jazyk s dynamickou typovou kontrolou (do premennej môžete uložiť čokoľvek, ale hodnoty si pamätajú svoj typ)

`a = 10 + "a"`

- názov premennej sa môže skladať z malých (a veľkých) písmen (bez diakritiky), čísel a podtržítka (`_`), nesmie začínať číslom, podtržítkom začínajú špeciálne premenné
- názov premennej sa nesmie zhodovať so žiadnym z 34 kľúčových slov jazyka Python:

and, as, assert, break, class, continue, def, del, elif, else, expect, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield, True, False, None

Základné dátové typy

- **boolean** (booleovský typ) nadobúda buď hodnoty *True* alebo *False*
- Čísla môžu byť celé (**integer**; 1 a 2), reálne (**float**; 1.1 a 1.2) alebo komplexné (**complex**; $3+1j$).
- Reťazce (**string**) sú postupnosti Unicode znakov. Tuto podobu môže mať napríklad html dokument.
- Bajty (**byte**) a pole bajtov, napríklad súbor s obrázkom vo formáte jpeg.
- Zoznamy (**list**) sú usporiadané postupnosti hodnôt.
- N-tice (**tuple**) sú usporiadané, nemenné postupnosti hodnôt.
- Slovníky (**dictionary**) sú neusporiadané kolekcie dvojíc kľúč-hodnota.

Matematické operátory

- + súčet (napr. $10 + 3$)
- rozdiel (napr. $10 - 3$)
- * násobenie (napr. $10 * 3$)
- / delenie (napr. $10/3$)
- // celočíselné delenie (napr. $10 // 3 = ??$)
- % modulo (zbytok po celočíselnom delení;
napr. $10 \% 3 = ??$)
- ** umocňovanie (napr. $10 ** 3$)
- () zátvorky
- abs() funkcia, ktorá vracia absolútnu hodnotu
- round(x,2) zaokrúhli x na 2 desatinné miesta

Špeciálne typy priradenia: += -= *= /= %= // =

a += 1

a = a + 1

Úloha

- Vyriešte rovnicu

$$y = \frac{2x}{\sqrt{x^2 + 1}}$$

- Pre $x_1 = 11$ a $x_2 = 66$
- Výsledok zaokrúhlite na 4 desatinné miesta

Logické operátory

and	a
or	alebo
not	negácia
==	je rovno
!=, <>	nie je rovno
>	je väčšie
<	je menšie
>=	je väčšie alebo rovno
<=	je menšie alebo rovno
is, is not	je, nie je (napr.: <i>1 is int()</i> alebo <i>a is not None</i>)
in, not in 'a'	je v, nie je v (napr.: 'pes' not in 'Harapes' alebo in 'test')

poradie operácií: matematické operácie; < > <= >=; == !=;
<>; is, is not; in, not in; not; and; or

Char (znak)

- ľubovoľný znak na klavesnici aj mimo ňu
- Python nemá špeciálny typ pre znak, používa preňho string, ale má preňho vstavané funkcie
- funkcia `chr()` vracia znak pre zadanú ASCII hodnotu
- funkcia `ord()` vracia ASCII hodnotu pre zadaný znak

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Reťazec

- je postupnosť znakov
- Python rozpoznáva reťazce ohraničené úvodzovkami " a apostrofmi '

```
reťazec = "ja som reťazec"
```

- Reťazce sa dajú
 - spojovať (reťaziť) pomocou operátoru +
"ahoj "+ "užívateľi "
 - opakovať operátorom *
"ahoj " * 10
- pristupovať ku konkrétnemu znaku pomocou indexu
reťazec[0] alebo reťazec [1:4]
- na reťazce môžeme volať vstavané funkcie
"ja som reťazec ".find("ja")
- môžeme zisťovať dĺžku reťazca
len("test")

Prístup k hodnotám

- môžeme pristupovať k akémukoľvek znaku pomocou jeho indexu `string[x]`, kde kladné čísla od nuly určujú index zľava a záporné čísla určujú index zprava

`"Danny"[0]` `"Danny"[1]` `"Danny"[-1]`

- cez dvojbodku môžeme nadefinovať rozsah
- `string[x:y]`, kde tieto výrazy si odpovedajú:
 - `string[:]` == `string`
 - `string[x:]` == `string[x:len(string)]`
 - `string[:y]` == `string[0:y]`
- čo bude výsledkom?

`"Danny"[1:4]`

`"Danny"[2:2]`

`"Danny"[2:]`

`"Danny"[-2:]`

`"Danny"[:2]`

`"Danny"[:-2]`

Vstavane funkcie

- Hľadanie

- **count**

- `string1.count(string2)`
`"Danny".count("ny")`
 - vracia počet výskytu `string2` v `String1`

- **find**

- `string1.find(string2)`
`"Danny".find("n")`
 - vracia index prvého výskytu `string2` v `string1`

- nahradzovanie a rozdeľovanie

- **replace**

- `string.replace(old, new)`
`"Danny".replace("an", "e")`
 - nahradí `old` za `new` v reťazci `string`

- **split**

- `string.split(sep)`
`"1 2 3".split(" ")`
 - Vracia list reťazcov, ktoré vzniknú rozdelením `string` podľa `sep`

Zmena veľkosti

upper

- `string.upper()`
`"danny".upper()`
- zväčší všetky písmena

lower

- `string.lower()`
`"DANNY".lower()`
- zmenší všetky písmena

title

- `string.title()`
`"danny je pes".title()`
- zväčší prvé písmena slov

capitalize

- `string.capitalize()`
`"danny je pes".capitalize()`
- zväčší prvé písmeno reťazca

Vstup a výstup

Vstup:

- `raw_input()` v Pythonu 2.x nebo `input` v Pythonu 3
- argumenty programu (cez modul `sys`):

```
import sys
```

```
    if len(sys.argv) > 1:
```

```
        print sys.argv[1]
```

- zo súboru

Výstup:

- `print()`
- `sys.stdout` a `sys.stderr`
- do súboru

Vstup zo súboru

- otvorenie súboru `open(name, mode)`
- módy: `r` , `r+` , `w` , `t` , `b`
- čítanie dát v súbore `read`
- zápis dát do súboru `write`
- uzatvorenie súboru `close`

```
myfile = open("test.txt", "r")
```

```
data = myfile.read()
```

```
myfile.close()
```

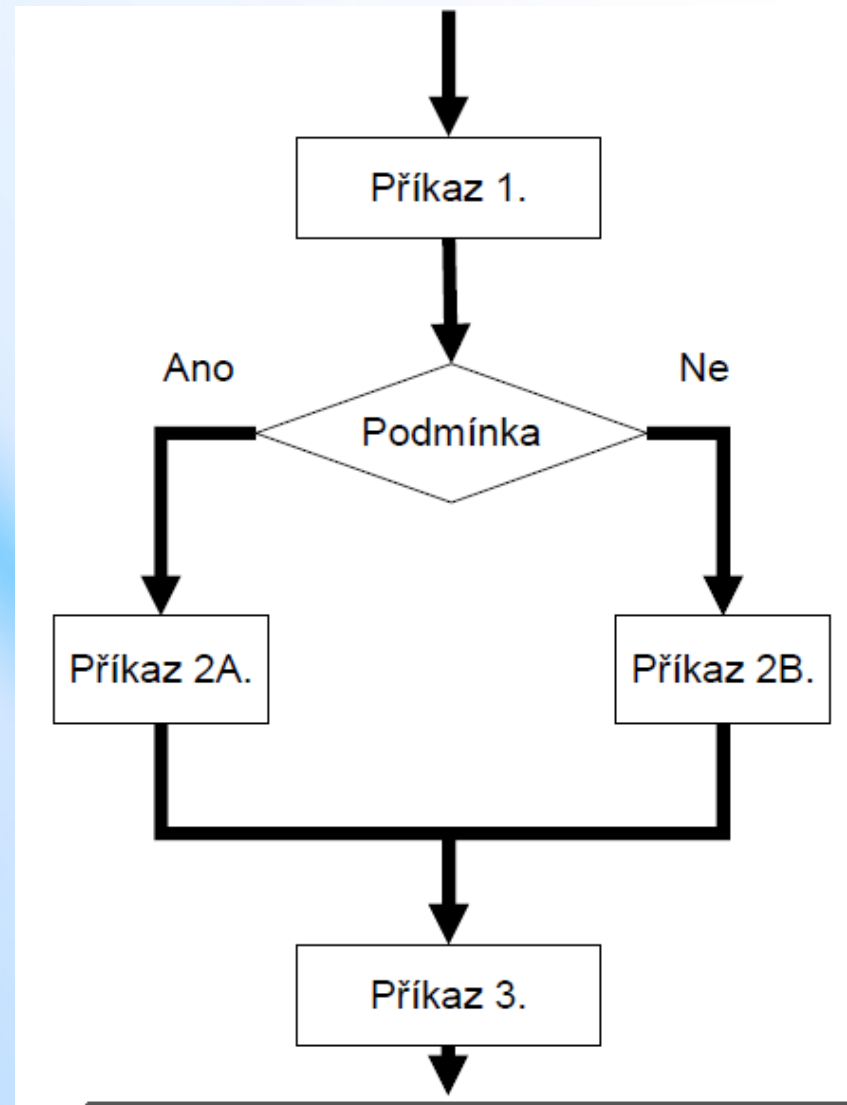
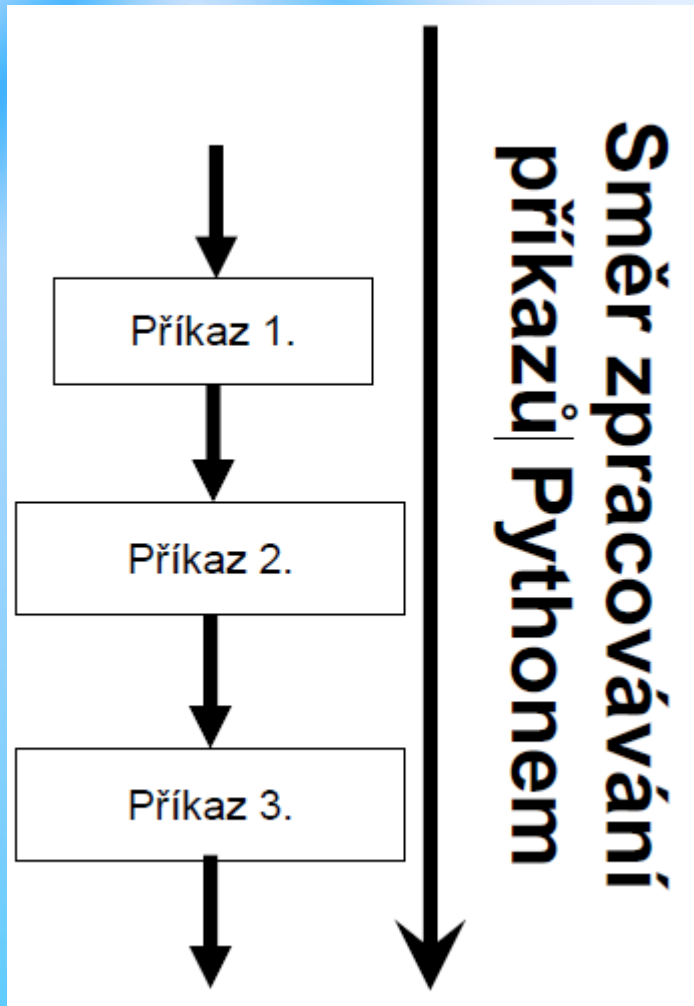
Úloha

- Stiahnite si súbor
- Uložte do priečinku v ktorom pracujete (tam kde si ukládáte skripty)

V skripte:

- Otvorte súbor "Kocur v botach.txt"
- Spočítajte výskyt slova "kocour" v texte
- Vypíšte počet

Vetvenie programu



Podmienka

```
if 'podmienka':  
    print("Podmienka splnena.")
```

```
if 9 > 5:  
    print("Deväť je väčšie ako päť.")
```

```
age = 20  
is_man = True  
if age >= 18 and is_man:  
    print("Plnoletý muž.")
```

Podmienka

Podmienka if .. else

```
if 'podmienka':  
    print("Podmienka splnená.")  
else:  
    print("Podmienka nesplnená.")
```

```
if 9 < 5:  
    print("Deväť je menšie ako  
    päť.")  
else:  
    print("Deväť je väčšie ako  
    päť.")
```

Podmienka if .. elif .. else

```
if 'podmienka' and 'podmínka2':  
    print("Podmienka splnená.")  
elif 'podmienka2':  
    print("Aspoň podmienka2  
    splnená.")  
else:  
    print("Podmienky  
    nesplnené.")
```


Úloha

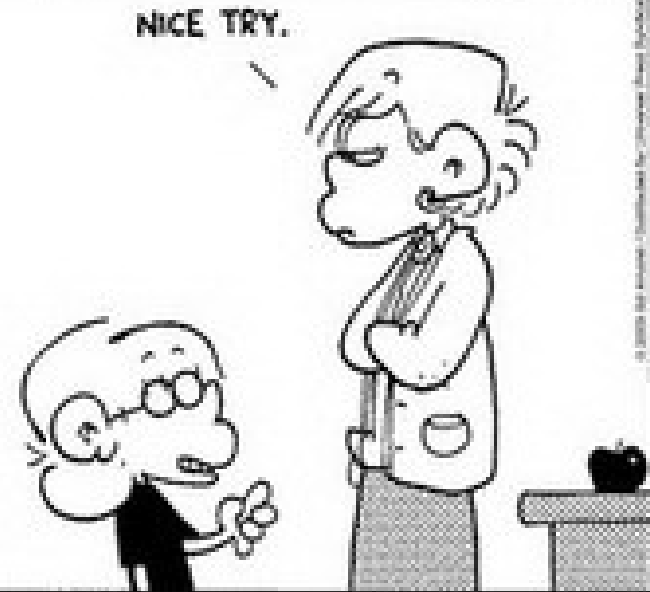
- Napíšte skript, ktorý načíta číslo zo štandardného vstupu a zistí, či je číslo deliteľné 3 alebo 5 alebo obomi.

Cykly

```
#include <stdio.h>
int main(void)
{
    int count;

    for (count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.");

    return 0;
}
```

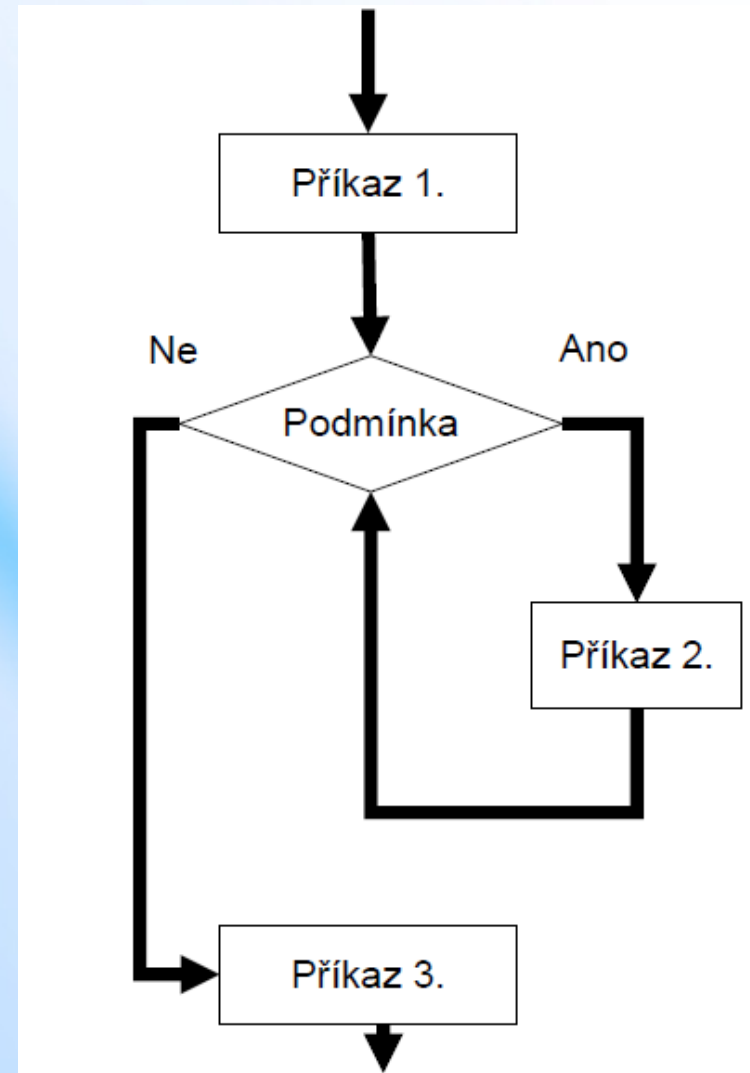


Podobne ako podmienky sa riadia logickým výrazom, ktorý rozhoduje o spustení príslušného bloku, tieto bloky môžu bežať i niekoľkokrát po sebe.

Cyklus while

```
while 'podmienka':  
    blok  
    blok
```

```
i = 0  
while i < 10:  
    print(i)  
    i += 1
```



Úloha

- Napíšte cyklus, ktorý vypíše prvých 10 čísel fibonacciho postupnosti
- **Fibonacciho postupnosť** je postupnosť čísiel, v ktorej každý ďalší člen je súčtom dvoch predchádzajúcich.

$$F_n = F_{n-2} + F_{n-1}$$

(1, 1, 2, 3, 5, 8, 13, 21...)

Cyklus for

- cyklus s predom známým počtom opakovaní

```
for i in ...:
```

```
    blok
```

```
    blok
```

```
data = [[i*j for i in range(10)] for j in range(10)]
```

```
for row in data:
```

```
    print(sum(row))
```

```
for i in range(10):
```

```
    print(i)
```

Prerušenie cyklu

- **break** vynúti ukončenie cyklu
- **continue** vynúti ukončenie vykonávania bloku a spustí ďalšiu smyčku

```
for i in range(100):  
    if i%3 == 0:  
        print("<3")  
        continue  
    if i >= 10:  
        break  
    print(i)
```

Úloha

- Napíšte skript, ktorý vypíše štvorec zložený z písmen x, dĺžku zadá užívateľ
- Napíšte skript tak, aby bol štvorec prázdny

```
x x x x
```

```
x x x x
```

```
x x x x
```

```
x x x x
```

```
x x x x
```

```
x     x
```

```
x     x
```

```
x x x x
```

- Aby funkcia print písala do riadku, je treba pridať na koniec čiarku ,

```
print " X",
```

Typy polí

- Zoznamy (List)

```
list = ["abc", 1, 3.14]
```

- N-tice (Tuples)

```
tuple = {"abc", 1, 3.14}
```

- Množiny (Set)

```
set = {"abc", 1, 3.14}
```

- Slovníky (Dictionary)

```
dict = {"string" : "abc", "integer" : 1, "float" : 3.14}
```


List (zoznam)

- patria do kolekcií, podobne ako N-tice a slovník
- vytvárame pomocou hranatých zátvoriek []
- ["a", "b", "c", "d"]
- každý prvok má svoj automatický index, ktorý odpovedá poradiu

Práce se zoznamy

vytvorenie

- `zoznam1 = [1, 1, 2, 3, 5, 8, 13]`
- `zoznam2 = list(zoznam1)`
- `zoznam3 = zoznam1[:]`
- `zoznam4 = zoznam1` # nejedna se o nový list, iba odkaz na starý
- `zoznam5 = range(2,20,2)` # [2, 4, 6, 8, 10, 12, 14, 16, 18]

pridávame prvky

- `zoznam1.append(21)` # [1, 1, 2, 3, 5, 8, 13, 21]
- `zoznam2.insert(2, 90)` # [1, 1, 90, 2, 3, 5, 8, 13]
- `zoznam3.extend([21, 34])` # [1, 1, 90, 2, 3, 5, 8, 13, 21, 34]
- `zoznam3.append([21, 34])` # [1, 1, 2, 3, 5, 8, 13, [21, 34]]

Práca so zoznamami

mazanie

- `zoznam1.remove(1)` # [1, 2, 3, 5, 8, 13, 21]
- `last = zoznam1.pop()` # last = 21; [1, 1, 2, 3, 5, 8, 13]

výber pomocou []: rovnako ako u reťazcov

- `zoznam1[1]`
- `zoznam1[1:5]`
- `zoznam1[-2:]`

prehodenie smeru

- `zoznam1.reverse()` # [13, 8, 5, 3, 2, 1, 1]

Vyhľadávanie

- `zoznam1.index(5)` # 4
- `zoznam1.count(1)` # 2

zoradenie

- `zoznam = [1, 4, 3, 6, 2, 5]`
- `zoznam.sort()` # [1, 2, 3, 4, 5, 6]
- `zoznam.sort(reverse=True)` # [6, 5, ...]

Práca so zoznamami

počítanie

- `zoznam1 = [1, 1, 2, 3, 5, 8, 13]`
- `len(zoznam1)` # 7
- `sum(zoznam1)` # 33
- `min(zoznam1)` # 1
- `max(zoznam1)` # 13

prechádzanie

- `for item in [1, 2, 3]:`
 `print(item)`
- `for i in range(0,len(zoznam1)):`
 `print(zoznam1[i])`
- `for item in (zoznam1):`
 `print (item)`

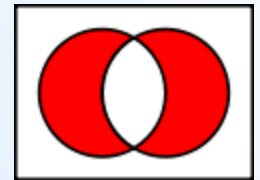
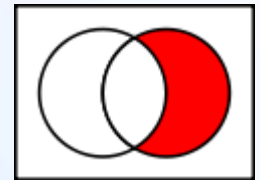
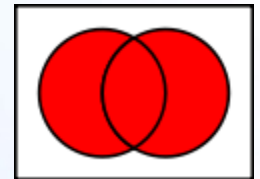
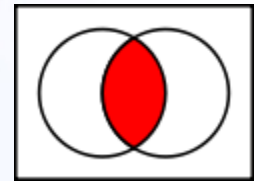
Tuple (N-tice)

- vytvárame pomocou jednoduchých zátvoriek ()
- môžeme s nimi pracovať podobne ako so zoznamami, iba ich nemôžeme meniť, tzn. že funkcia `append` a ďalšie nie sú dostupné
- môžeme jednoducho prevádzať na list pomocou `list((1,2))` a podobne späť `tuple([1,2])`

Množiny

- Neusporiadaná množina jedinečných hodnôt
- Vytvorenie: $A=\{1,2,3\}$; $B=\{3,4,5\}$

- Prienik $A \cap B \# \{3\}$
- Zjednotenie $A \cup B \# \{1, 2, 3, 4, 5\}$
- Rozdiel $A - B \# \{1, 2\}$
- Symetricky rozdiel $A \Delta B \# \{1, 2, 4, 5\}$



Množiny

- Základné operácie s množinami
 - `len(s)` # vráti počet prvkov
 - `x in s` # testuje či je x v množine
 - `s.issubset(t)` # testuje či každý prvok množiny s je v t
 - `s.union(t)` # nová množina obsahujúca prvky s aj t
 - `s.intersection(t)` # množina s prvkami v oboch
 - `s.difference(t)` # to isté ako s-t
 - `s.symmetric_difference(t)` # v s alebo t ale nie v oboch

Dictionary (slovník)

- vytvárame pomocou zložených zátvoriek {}
- {1: one, 2: two}
- prvok v slovníku sa skladá z kľúča a jeho hodnoty
- 1 a 2 sú kľúče, ich hodnoty sú one, resp. two
- nefungujú zde indexy, na hodnoty sa dotazujeme pomocou kľúča
- každý kľúč je unikátny, žiadny slovník nemôže obsahovať dva rovnaké kľúče

Operácie so slovníkmi

vytvorenie

- `dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}`

čítanie

- `print(dict['Name'])` # Zara

úprava hodnôt

- `dict['Age'] = 8` # úprava stávajúcej hodnoty
- `dict['School'] = "DPS School,"` # pridanie novej

zmazanie hodnôt

- `del dict['Name']`
- `dict.clear()` # zmaže všetky položky
- `del dict` # zmaže celý slovník

prechádzanie hodnôt

- `for key in dict:`
 `print(key)`
 `print(dict[key])`

Funkcie

- opakované spúšťanie rovnakého (veľmi podobného) kódu na rôznych miestach algoritmu
- ľahšie uvažovanie o problému, deľba práce

```
def fahr_to_kelvin(temp):  
    return ((temp - 32) * (5/9)) + 273.15
```

The diagram illustrates the components of a Python function definition. It shows the code: `def fahr_to_kelvin(temp):` followed by an indented `return ((temp - 32) * (5/9)) + 273.15`. Labels with arrows point to specific parts: 'def keyword' points to 'def', 'name' points to 'fahr_to_kelvin', 'parameter' points to '(temp)', 'return statement' points to 'return', and 'return value' points to the entire expression '((temp - 32) * (5/9)) + 273.15'.

Funkcie

- funkcia je **časť programu (je podprogram, angl. subroutine)**, ktorá sa dá volať opakovane z rôznych častí programu
- každá funkcia má svoj **identifikátor** (podobne ako premenná), pomocou ktorého ju môžeme volať
`moja_funkcia()`
- požiadavky na funkciu môžeme špecifikovať pomocou **parametrov**
`moja_funkcia(parametr1, parametr2)`
- výsledok funkcie môže byť predaný ako **návratová hodnota**
`navratova_hodnota = moja_funkcia()`

Definice funkce

- pomocou klúčového slova **def** a bloku

```
def moja_funkcia(parameter):  
    pass
```

- návratovú hodnotu definujeme klúčovým slovom **return**,
- POZOR: po jej zavolaní sa už v prevádzaní funkcie nepokračuje

```
def moja_funkcia(parameter):  
    print()  
    return True  
    print() # nedosiahnutelný kód
```

- pomocí return môžeme navracať hodnotu, hodnotu premennej alebo tiež nič (None)

```
def moja_funkcia(parameter):  
    return
```

Funkcie - príklady

```
def sum_two_numbers(a, b):  
    return a + b
```

```
>> sum_two_numbers(27, 20)
```

```
def dec_to_bin (n):
```

```
    output = []
```

```
    if n == 0:
```

```
        return [0]
```

```
    else:
```

```
        while n > 0:
```

```
            if n % 2 == 0:
```

```
                output.append (0)
```

```
            else:
```

```
                output.append (1)
```

```
                n = n // 2
```

```
    return output
```

Rekurzia

- definovanie funkcie pomocou seba samej
- funkcia, ktorá v svojom tele volá samu seba
- volanie funkcie je vždy podmienené, aby bolo zaručené, že výpočet skončí

```
def sequence(n):  
    if n > 1:  
        sequence(n-1)  
    print(n)
```

Úloha

- Napíšte funkciu, ktorá vráti faktoriál zo zadaného čísla

```
def faktorial(a):
```

```
    ...
```

```
    return faktorial
```

Úloha

- Napíšte funkciu, ktorá zistí či zadaný reťazec je palyndrom
- Palyndrom je ľubovoľná postupnosť symbolov, ktorá je rovnaká (má rovnaký význam) pri čítaní z oboch strán (zľava doprava i sprava doľava).
- *oko, kajak, jelenovi pivo nelej, 191 ...*
- Bez diakritiky
- Zjednodušenie:
 - Zadávanie reťazca bez medzier
 - Všetky písmená malé/veľké