25. Podmínky a cykly: while, for, if v různých jazycích, pass, continue, break

Větvení programu = běh programu **různými cestami** (podmínkami)
Podmínka = vyhodnocení na základě logického výrazu bool (True/False)
Blok příkazů = co všechno se má udělat při splnění podmínky
V pythonu:

- **za podmínky dvojtečka**
- blok příkazů je odsazen **mezerou**
- za jednotlivé příkazy se **nepíše nic**

V C:

- **za podmínky závorky**
- blok příkazů je ukončen další **závorkou,**
- za jednotlivé příkazy se píše **středník**

Operátory podmínek	
bool	True/False
==	Rovnost
!=	Nerovnost
>	Větší než
<	Menší než
>=	Větší nebo rovno
<=	Menší nebo rovno
! (not)	Logická negace
|| (or)	Logický součin
&& (and)	Logický součet

if:

- **True**: provede se příkaz
- **False**: program pokračuje dále

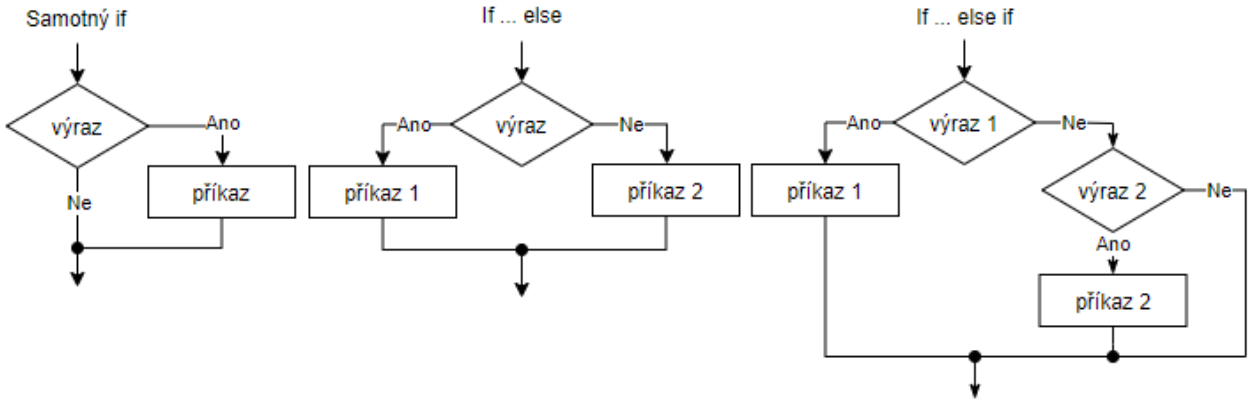
```
if (x == 1):
    print("x = 1")
else if (x >= 2):
    print("x >= 2")
else:
    print("x <= 0")
```

if … else if:

- **False**: další větvení podmínkou if
- V pythonu se může zapsat jako **elif**

if … else:

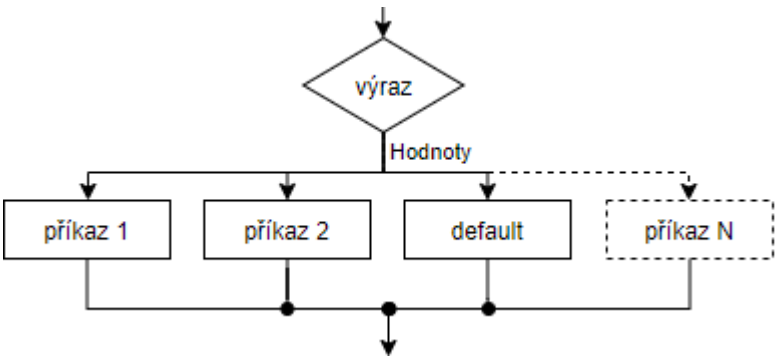
- **False**: provede se defaultní příkaz



switch:

- **Není v pythonu**, přesto se dá vytvořit podmínkami if … else if … else if … else
- Vstup musí být stejného typu jako porovnávací hodnoty (string <--> string, int <--> int)
- Kontroluje **jestli vstup == hodnota**, neumí porovnávat intervaly
- Zápis v C:

```
int x;
switch(x){
  case 0: printf("0"); break;
  case 1: printf("1"); break;
  default: printf("?"); break;
}
```



Cyklus = opakování běhu části programu
Tělo cyklu = příkazy uvnitř cyklu

while:

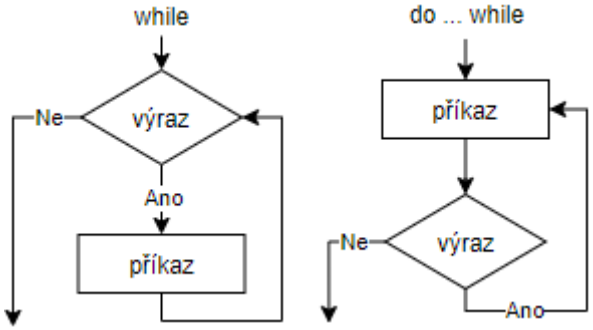
- Cyklus s **podmínkou na začátku**, **False před startem** -> cyklus nezačne
- **Nevíme kolikrát se příkaz udělá**
- **True:** provede příkaz
- **False:** ukončí cyklus

```
import random

x = 0
while(x != 5):
    x = random.randrange(0,6)
    print(x)
```

do … while:

- Cyklus s **podmínkou na konci**, **False před startem** -> příkaz se udělá jednou
- **Není v pythonu**, dal by se (neefektivně) vytvořit pomocí while … if … break
- **True:** provede příkaz
- **False:** ukončí cyklus



for:

- Cyklus s určeným počtem opakování
- Cyklus se **ukončí po vyčerpání všech prvků**

Zápis v Pythonu:

- **Funguje jinak** než v jiných jazycích, zapisuje se **for … in …**
- **Proměnná se nahrazuje** dalším číslem/písmenem posloupnosti
- Kolikrát se příkaz provede určujeme pomocí **range()** (vrací posloupnost jako array čísel)
- Pokud přiřadíme string/array, budou se příkazy opakovat dokud žádný nezbude

```
## range() vrátí [0,1,2,3,4]

for x in range(5):
    print(x)
```

```
## x se postupně nahrazuje pozdravy
pozdrav = ["ahoj", "cus", "cau"]
for x in pozdrav:
  print(x)
```

Zápis v C:

- 3 výrazy na začátku:
  - 1 -> **příkaz před prvním** provedením cyklu
  - 2 -> **podmínka před každým** provedením cyklu
  - 3 -> **příkaz po** provedení cyklu

```
for (int i = 0; i < 5; i++){
  printf("ahoj");
}
```

continue:

- Přeskočí jeden krok cyklu

break:

- Okamžitě ukončí celý cyklus

```python
## Program vypíše pouze cus a cau

pozdrav = ["ahoj", "cus", "cau"]
for x in pozdrav:
    if (x == "ahoj"):
        continue
    print(x)
```

```python
## Program nevypíše nic

pozdrav = ["ahoj", "cus", "cau"]
for x in pozdrav:
    if (x == "ahoj"):
        break
    print(x)
```

pass:

- **Pouze v pythonu** (nikde jinde)
- Nic nedělá, jiný zápis **return Null**
- Když chceme vytvořit funkci/objekt, kterému **zatím nechceme nic přiřazovat**

```python
def funkceCoNicNeumi():
    pass
```