

Podmíněný příkaz

Podmíněný příkaz se používá ve chvílích, kdy se potřebujeme rozhodnout na základě nějaké podmínky.

```
if podmínka:           # když podmínka platí, vykonej 1. skupinu příkazů
    příkaz
    příkaz
    ...
else:                 # když podmínka platí, vykonej 2. skupinu příkazů
    příkaz
    příkaz
    ...
```

Podmínky je možné vnořovat do již vytvořené podmínky – takový zápis nazýváme **vnořený příkaz if**.

Pro vnořený příkaz se v Pythonu používá příkaz **elif**:

```
if podmienka_1:       # když podmínka _1 platí, vykonej 1. skupinu příkazů
    příkaz
    ...
elif podmienka_2:    # když podmínka _1 neplatí, ale platí podmínka_2, ...
    příkaz
    ...
elif podmienka_3:    # když podmínka _1neplatí a ani podmínka _2 neplatí, ale platí podmínka _3, ...
    příkaz
    ...
else:                # když žádná z podmínek neplatí, ...
    příkaz
    ...
```

Pokud se v podmínkách jedná o interval určitého rozpětí, dají se podmínky zapsat pouze s **if**. Větev s příkazem **else** může chybět:

```
if body >= 90:         #když je číslo větší než 90, vykonej následující příkaz
    příkaz
if 80 <= body < 90:     #když je od 80 do 89, vykonej následující příkaz
    příkaz
if 70 <= body < 80:     #když je od 70 do 79, vykonej následující příkaz
    příkaz
if body < 70:          #když je číslo menší než 69, vykonej následující příkaz
    příkaz
```

Příkaz **pass** se používá, pokud v daném případě nechceme dělat nic.

Způsoby zápisu podmínek:

$a < b$	je menší
$a \leq b$	je menší nebo rovno
$a == b$	rovná se
$a != b$	nerovná se
$a > b$	je větší
$a \geq b$	je větší nebo rovno
$a < b \leq c$	příkazy je možné i řetězit
$a < b < c$	a je menší než b a b je menší než c

Podmínky v Pythonu mohou obsahovat logické operandy:

podmienka1 **and** *podmienka2* ... znamená, že musí platit obě podmínky

podmienka1 **or** *podmienka2* ... znamená, že musí platit alespoň jedna z podmínek

not *podmínka* ... znamená, že daná podmínka neplatí

Podmínky, které platí, mají hodnotu **True** (1) a podmínky které neplatí, mají hodnotu **False** (0).

Logika argumentů „**and**“

A	B	=
<i>False</i>	<i>False</i>	<i>False</i>
<i>True</i>	<i>False</i>	<i>False</i>
<i>False</i>	<i>True</i>	<i>False</i>
<i>True</i>	<i>True</i>	<i>True</i>

Logika argumentů „**or**“

A	B	=
<i>False</i>	<i>False</i>	<i>False</i>
<i>True</i>	<i>False</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>
<i>True</i>	<i>True</i>	<i>True</i>

Logika argumentů „**not**“

A	notA=
<i>False</i>	<i>True</i>
<i>True</i>	<i>False</i>

Logické pravidla v Pythonu platí i pro skoro všechny další typy. Python pro každý typ definuje případy, kdy je parametr False (tj. 0) a všechny ostatní případy, kdy je True.

Např.:

```
typ  False  True
int   $x == 0$    $x != 0$ 
float  $x == 0.0$   $x != 0.0$ 
str   $x == ''$    $x != ''$ 
```

Přerušení cyklu pomocí příkazu **break**

Příkaz **break** v těle cyklu způsobí ukončení cyklu, tzn., že vykonávání příkazů cyklu se v tomto momentě přeruší a pokračuje se až za prvním příkazem za cyklem. Proměnná cyklu bude mít hodnotu posledního cyklu.

Např.:

```
for přeměnná in ...:
    příkazy1
    if podmínka:
        break
    příkazy2
příkazy_za_cyklom
```

V každém přechodě cyklu se nejprve vykoná **příkaz1**, potom se zkontroluje podmínka a pokud je pravdivá, vykonávání cyklu končí a pokračuje se na **příkaz_za_cyklem**. Pokud podmínka pravdivá není, příkaz **break** se přeskočí a pokračuje se na **příkaz2**.

Podmíněný cyklus

je konstrukce cyklu, která opakuje vykonávání posloupnosti příkazů v závislosti na určité podmínce:

```
while podmínka:          # opakuj příkazy, kým platí podmínka
    příkaz
    příkaz
    ...
```

Tento nový příkaz postupně:

- Zjistí hodnotu podmínky, která je zapsaná za slovem **while**
- Pokud má tato podmínka hodnotu **False**, blok příkazů, který je v těle cyklu se přeskočí a pokračuje se na následném příkaze za celým while-cyklem.
- Pokud má podmínka hodnotu **True**, vykonají se všechny příkazy v těle cyklu.
- Znovu se otestuje podmínka za slovem **while** a to celé se opakuje

While-cykly se nejčastěji používají ve chvíli, kdy zápis pomocí **for-cyklu** je příliš komplikovaný anebo se udělat nedá. Podmínka **while-cyklu** neříká kdy má skončit, ale naopak, pokud podmínka platí, vykonávají se všechny příkazy v těle.

Princip půlení hodnot (při zjišťování druhé odmocniny):

je vhodný ve chvílích, kdy potřebujeme zjistit určitou hodnotu nacházející se ve známém rozsahu, kdy pokaždé rozdělíme počet na půl a zjišťujeme, která z půlek obsahuje dané číslo a tento proces se opakuje, až do doby, než zbude jen hledané číslo.

- zvolíme si interval, v kterém se určitě bude nacházet hledaný výsledek (hledaná odmocnina), například nech je to interval $<1, \text{číslo}>$ (pro čísla větší než 1 je i odmocnina větší než 1 a určitě je menší než samotné číslo)
- jako „x“ (první odhad naší hledané odmocniny) zvolíme střed tohoto intervalu
- zjistíme, zda je druhá mocnina tohoto „x“ větší než zadané číslo, nebo menší
- když je větší („x“ je už zbytečně veliké), tak upravíme předpokládaný interval, tak že jeho horní hranici změním na „x“
- když je ale menší, upravíme dolní hranici intervalu na „x“
- tím se nám interval zmenšil na polovic
- toto celé opakujeme, až dokud je nalezené „x“ dostatečně blízko k hledanému výsledku, tj. či se neliší od výsledku méně než zvolený rozdíl (epsilon)

Zapišeme to:

```
číslo = float(input('zadej číslo:'))
```

```
od = 0
```

```
do = číslo
```

```
x = (od + do) / 2
```

```
počet = 0
```

```
while abs(x**2 - číslo) > 0.001:
```

```
    if x**2 > číslo:
```

```
        do = x
```

```
    else:
```

```
        od = x
```

```
    x = (od + do) / 2
```

```
    počet += 1
```

```
print('druhá odmocnina', číslo, 'je', x)
```

```
print('počet přechodů while-cyklem byl', počet)
```

Nekonečný cyklus

Cyklus s podmínkou, která má stále hodnotu **True**, bude nekonečný.

Např.:

```
i = 0
```

```
while i < 10:
```

```
    i -= 1
```

Takovýto cyklus můžeme přerušit stisknutím kláves **Ctrl/C**.