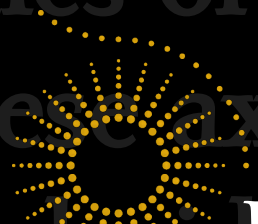
Funkce & Cykly
(6. přednáška)



FF UK
Department of Logic

Jonathan L. Verner

while cykly

FF UK
Department of Logic

```
while <condition>:
  BLOKA
BLOKB
```

- pokud je splněna podmínka `<condition>`, provede se `BLOKA`
- předchozí krok se opakuje dokud je splněna podmínka `<condition>`
- pokud python narazí na příkaz `break`, ukončí pokračuje vykonáváním `BLOKB`
- "zajišťují Turingovskou úplnost"

while cykly příklad

FF UK
Department of Logic

```
C = 0
M = 0
P = -1
S = 0

while C >= 0:
    S += C
    if C > M:
        M = C
    P += 1
    C = input('Cislo (zaporne ukonci):')

print("P:", P)
print("M:", M)
print("A:", S/P)
```

Funkce

FF UK
Department of Logic

```
def quadsolve(a,b,c):
    D=b**2-4*a*c
    if D >= 0:
        x1 = -1*b+sqrt(D)
        x2 = -1*b-sqrt(D)
        return x1, x2
```

```
def <jmeno>(<p_1>[=h_1],...,<p_n>[=h_n]):
    <telo_funkce>
```

- `jmeno` je identifikátor
- `p_1`, . . ., `p_n` jsou identifikátory (“závislé proměnné”)
- `h_1`, . . ., `h_n` jsou výrazy specifikující “defaultní” hodnoty proměnných, pokud je uživatel nezadá
- `telo_funkce` je Pythonovský kód specifikující jak spočítat výsledek
- `return` je příkaz, který lze použít v `telo_funkce` k označení konce výpočtu a specifikaci výsledné hodnoty

Funkce

scope

FF UK
Department of Logic

- definice z vnějšku se dědí (volné proměnné)
- definice uvnitř funkce zůstávají pouze uvnitř funkce (vázané proměnné)

```
def test():
    a = 10
    x = 20

    def mod(a, b):
        a = a + b + x
        print(a)

    mod(a, 10)

    print(a)
```

Funkce

mutable x immutable

FF UK
Department of Logic

```
def test2():
    a = [1,2,3]

    def mod1(a):
        a.append(4)
        print(a)

    def mod2(a):
        a = a + [4]
        print(a)

    mod1(a)
    mod2(a)

    print(a)
```