If ... then ... else ...

(4. přednáška)

FF UK
Department of Logic

Jonathan L. Verner

Podmíněné příkazy

FF UK
Department of Logic

- v předchozích přednáškách byly programy v podstatě lineární
- "lineární" algoritmy jsou schopné vyřešit relativně velkou třídu problémů
- někdy to ale není ono

Řešení kvadratické rovnice

FF UK
Department of Logic

```
from math import sqrt

def solutions(a,b,c):
  diskr = b**2 - 4*a*c
  x1 = (-b + sqrt(diskr))/(2.0*a)
  x2 = (-b - sqrt(diskr))/(2.0*a)
  return (x1, x2)

def solvequad():
  print("Zadejte koeficienty a,b,c")
  a,b,c=int(input("a:")),int(input("b:")),int(input("c:"))
  print("Koreny jsou:", solutions(a,b,c))
```

Řešení kvadratické rovnice

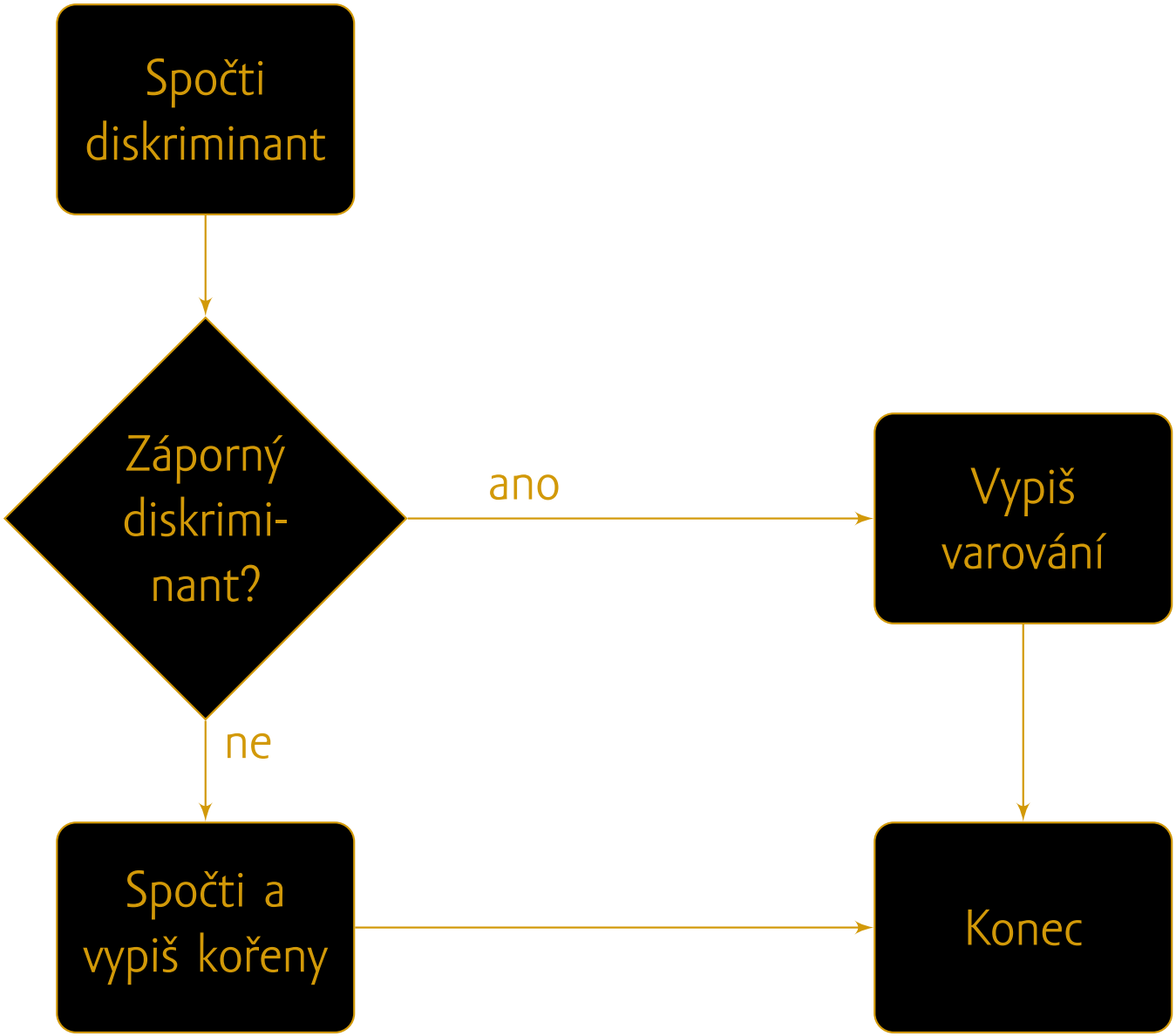
FF UK
Department of Logic

```
>>> solvequad()
Zadejte koeficienty a,b,c
a:10
b:10
c:10
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    solvequad()
  File "/tmp/quad.py", line 12, in solvequad
    print("Koreny jsou:", solutions(a,b,c))
  File "/tmp/quad.py", line 5, in solutions
    x1 = (-b + sqrt(diskr))/(2.0*a)
ValueError: math domain error
```

Kvadraticka rovnice

flowchart

FF UK
Department of Logic



Řešení kvadratické rovnice

2. verze

FF UK
Department of Logic

```
from math import sqrt

def solutions(a,b,c):
  diskr = b**2 - 4*a*c
  if diskr >= 0:
    x1 = (-b + sqrt(diskr))/(2.0*a)
    x2 = (-b - sqrt(diskr))/(2.0*a)
    return (x1, x2)
  else:
    return "Rovnice nema (realne) reseni"

def solvequad():
  print("Zadejte koeficienty a,b,c")
  a,b,c=int(input("a:")),int(input("b:")),int(input("c:"))
  print("Koreny jsou:", solutions(a,b,c))
```

solvequad

2. verze

FF UK
Department of Logic

```
>>> solvequad()
Zadejte koeficienty a,b,c
a:10
b:10
c:10
Koreny jsou: Rovnice nema (realne) reseni
```

If...else statement

FF UK
Department of Logic

```
if <podminka>:
  blokA
else:
  blokB
```

- `<podminka>` musí být výraz, který má hodnotu typu Boolean
- pokud ma `<podminka>` hodnotu `True`, provede se `blokA`
- jinak se provede `blokB`
- `else: ...` nemusí být přítomno

Booleovské výrazy

nerovnosti mezi čísly

FF UK
Department of Logic

Matematický zápis	Zápis v Pythonu
$<$	`<`
$>$	`>`
$\leq$	`<=`
$\geq$	`>=`
$=$	`==`
$\neq$	`!=`

Booleovské výrazy

not, and, or

FF UK
Department of Logic

```
>>> x = 5
>>> x > 2 or x < 6
True
>>> x > 2 and x > 6
False
>>> not x < 3
True
```

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Quizz

FF UK
Department of Logic

Napište program, který se uživatele zeptá na booleovský výraz obsahující čtyři proměnné a,b,c,d a vypíše tabulku obdobnou té předchozí

Hint: Použijte funkci `eval` a cyklama projdete vsechny moznosti.

Quizz

FF UK
Department of Logic

```
>>> tabulka()
Zadej bool vyraz: (a or b) and (c or d)
 A    |   B    |   C    |   D    |(a or b)and(c or d)
-------------------------------------------------------
True   | True   | True   | True   |           True
True   | True   | True   | False  |           True
True   | True   | False  | True   |           True
True   | True   | False  | False  |           False
True   | False  | True   | True   |           True
True   | False  | True   | False  |           True
True   | False  | False  | True   |           True
True   | False  | False  | False  |           False
False  | True   | True   | True   |           True
False  | True   | True   | False  |           True
False  | True   | False  | True   |           True
False  | True   | False  | False  |           False
False  | False  | True   | True   |           False
False  | False  | True   | False  |           False
False  | False  | False  | True   |           False
False  | False  | False  | False  |           False
```

solvequad

3. verze

FF UK
Department of Logic

```
>>> solvequad()
Zadejte koeficienty
a:1
b:2
c:1
Koreny jsou: (-1.0, -1.0)
```

- to může být trochu matoucí
- chtěli bychom, aby pro dvounásobné kořeny vypsal hlášku

```
spočti diskr
pokud:
 diskr < 0   vypiš chybovou hlášku o imaginárních kořenech
 diskr == 0  vypiš hlášku o dvojitém kořeni
 diskr > 0   vypiš oba kořeny
```

solvequad

3. verze - vnořené `if ... else`

FF UK
Department of Logic

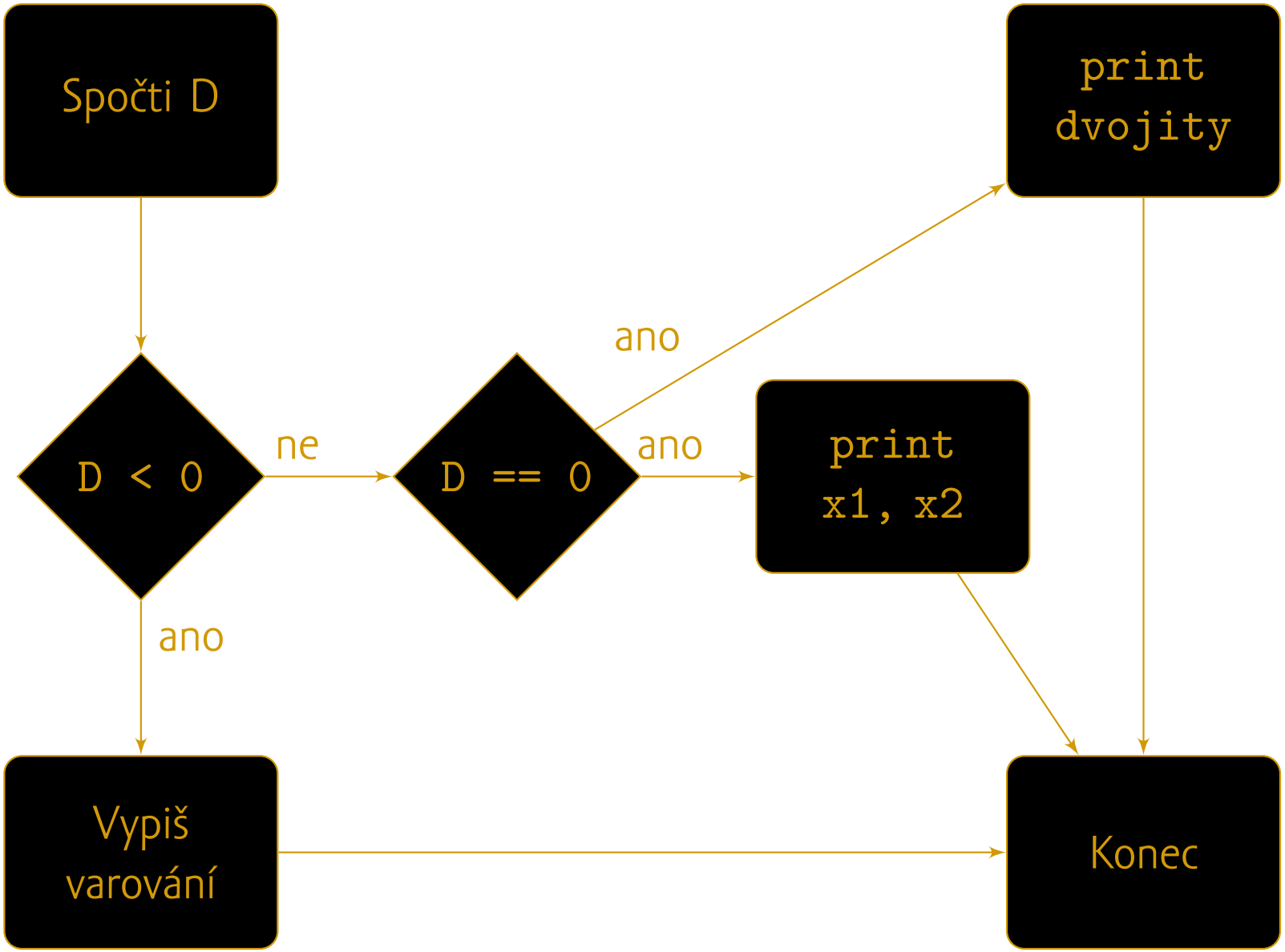
```
from math import sqrt

def solutions(a,b,c):
  D = b**2 - 4*a*c
  if D >= 0:
    x1 = (-b + sqrt(D))/(2.0*a)
    x2 = (-b - sqrt(D))/(2.0*a)
    if D == 0:
      return x1
    else:
      return x1, x2
  else:
    return "Rovnice nema (realne) reseni"
```

solvequad

FF UK
Department of Logic

3. verze - flowchart



solvequad

3. verze - `elif`

FF UK
Department of Logic

```
from math import sqrt

def solutions(a,b,c):
  D = b**2 - 4*a*c
  if D > 0:
    x1 = (-b + sqrt(D))/(2.0*a)
    x2 = (-b - sqrt(D))/(2.0*a)
  elif D == 0:
      return (-b + sqrt(D))/(2.0*a)
  else:
    return "Rovnice nema (realne) reseni"
```

If ... elif ... else

FF UK
Department of Logic

```
if <podminka1>:
  blok1
elif <podmina2>:
  blok2
elif <podminka3>:
  blok3
...
else:
  blokE
```

- provede se vždy první blok, jehož podmínka je splněna
- provede se vždy nejvýše jeden blok!!

Výjimky

FF UK
Department of Logic

- jedno z typických použití 'if'-ů: vyhnout se chybám
- když je 'if'-ů moc, hlavní algoritmus se může ztratit
- jeden ze způsobů jak to řešit jsou tzv. výjimky

Dělej A a pokud dojde k chybě, proveď B

```
def solutions(a,b,c):
  try:
    D = b**2 - 4*a*c
    x1 = (-b + sqrt(diskr))/(2.0*a)
    x2 = (-b - sqrt(diskr))/(2.0*a)
    return (x1, x2)
  except ValueError:
    return "Rovnice nema (realne) reseni"
```

try ... except

FF UK
Department of Logic

```
try:
  <blokA>
except <TypChyby1>[ as <identifikator1>]:
  <blok1>
except <TypChyby2>[ as <identifikator2>]:
  <blok2>
...
except:
  <blokE>
```

- python provede `<blokA>`
- pokud dojde k chybě, vyhledá řádku, která odpovídá typu chyby, ke které došlo a vykoná odpovídající blok
- pokud je volitelně přítomný název proměnné (`<identifikatorn>`), pak se do této proměnné vloží informace o chybě

```
def solutions(a,b,c):
  try:
    D = b**2 - 4*a*c
    x1 = (-b + sqrt(diskr))/(2.0*a)
    x2 = (-b - sqrt(diskr))/(2.0*a)
    return (x1, x2)
  except ValueError as chyba:
    msg = unicode(chyba)
    if not msg == 'math domain error':
      print("Chyba: Neco se pokazilo.")
  except TypeError:
    print("Chyba: Nektery koeficient nebyl ciselny")
  except:
    print("Chyba: Neco se pokazilo.")
  return "Rovnice nema (realne) reseni"
```

Maximum ze tří čísel

FF UK
Department of Logic

Napiště program, který vypíše největší ze tří čísel

```
x1,x2,x3=eval(input("Zadej tri cisla oddelena carkou: "))

...

print "Maximum je ", max
```

Maximum ze tří čísel

FF UK
Department of Logic

Procházení všech možností

- je třeba provést jedno z následujících přiřazení:

```
max = x1
max = x2
max = x3
```

- zdá se, že stačí akorát vložit odpovídající 'if'-y:

```
if x1 >= x2 >= x3:
  max = x1
if x2 >= x1 >= x3:
  max = x2
if x3 >= x2 >= x1:
  max = x3
if x1 >= x2 and x1 >= x3:
  max = x1
elif x2 >= x1 and x2 >= x3:
```

Co by se stalo, kdybychom chtěli počítat maximum ze 4?

- složitost podmínek by narostla
- pokud bychom chtěli zjistit maximum z n-čísel, potřebovali bychom n-podmínek, každá by měla $n - 1$ členů, t.j. $n(n - 1)$ porovnání!
- podmínky neberou v potaz, co už víme z předchozích

FF UK
Department of Logic

Pokud už vím, že

```
x1 >= x2
```

pak mi stačí testovat jen

```
x1 >= x3
```

Pokud naopak není pravda, že

```
x1 >= x2
```

pak mi stačí testovat

```
x2 >= x3
```

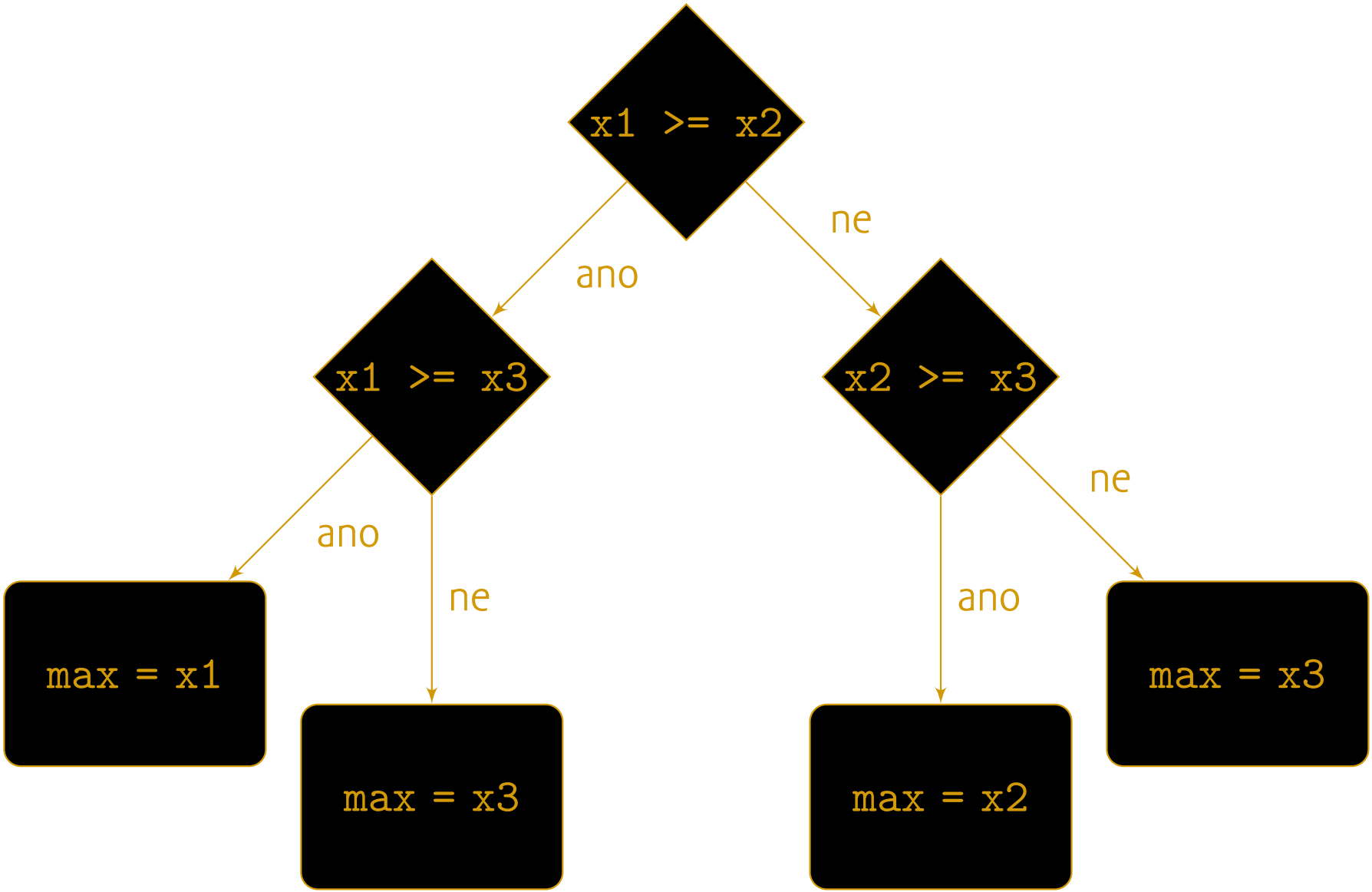
Rozhodovací strom

```
if x1 >= x2:
    if x1 >= x3:
        max = x1
    else:
        max = x3
else:
    if x2 >= x3:
        max = x2
    else:
        max = x3
```

maxofthree

2. verze - flowchart

FF UK
Department of Logic



maxofthree

2. verze - doesn't scale

- výpočetně efektivní: minimum nutných porovnání
- neefektivní zápis !!!

maxofthree

3. verze

FF UK
Department of Logic

- jak by to řešil člověk?
- pro 3 čísla nejspíše metodou kouknu a vidím
- pokud ale čísel bude 1000?
- sekvenciální průchod

maxofthree

3. verze - sekvenciální průchod

FF UK
Department of Logic

```
nastav max na prvni clen posloupnosti
prejdi k dalsimu clenu
je-li aktualni clen vetsi nez max:
  nastav max na aktualni clen
prejdi k dalsimu clenu
```

```
max = x1
if x2 > max:
  max = x2
if x3 > max:
  max = x3
```

maxofthree

verze 3.1 - obecné řešení

FF UK
Department of Logic

```
# program: maxofn.py
#    Najde maximum z posloupnosti cisel
# autor: Student Logiky

def maximum():
    n = int(input('Delka posloupnosti:'))

    max = float(input('Zadej prvni clen:'))

    for i in range(n-1):
        x = float(input('Zadej dalsi clen:'))
        if x > max:
            max = x

    print("Maximum je", max)
```

maxofthree

4. verze - ”podvodné řešení“

FF UK
Department of Logic

```
def maximum():
  x1, x2, x3 = eval(input("Zadej tri cisla (a,b,c):"))
  print("Nejvetsi cislo je", max(x1,x2,x3))
```

Poučení

FF UK
Department of Logic

- většinou je více cest, jak dojít k řešení
- ne vždy je první nápad nejlepší
- často (ne vždy) je dobré se zamyslet nad obecným případem
- ”do not reinvent the wheel”