Algoritmy a programování - Úvod

Algoritmy a programování

- *Přednášející*:
  - Jan Kybic, `http://cmp.felk.cvut.cz/~kybic`, `kybic@fel.cvut.cz`
- *Stránky předmětu:*
  - `https://cw.fel.cvut.cz/wiki/courses/b3b33alp/start`

Organizace předmětu

- *Přednášky*:
  - Účast je nepovinná, ale doporučená
- *Cvičení*:
  - samostatná práce pod vedením cvičícího
  - automatické ověření správnosti řešení
  - 50% váhy zkoušky
  - nerozumíte-li, ptejte se
  - *pokročilí* - chtějte další úlohy
- *Domácí práce*:
  - úlohy ze cvičení
  - další úlohy dle potřeby
- *Literatura:*
  - Sedgewick et al.: Introduction to programming in Python
  - Wentworth et al.: Learning with Python 3
  - Zelle et al.: Python Programming: An Introduction to Computer Science
  - Kubias (překl.): Učíme se programovat v jazyce Python 3
- *Konzultace* - po přechozí domluvě

Proč se učit programování

- číst, psát, počítat, *programovat*
- počítače jsou všude
- počítač je nástroj, *program* je nástroj
- programování učí logicky a strukturovaně myslet

Co se máme naučit

Řešení problémů

- formulace problému
- analýza možných řešení a návrh algoritmu
- implementace v programovacím jazyce
- testování, ověření funkčnost
- optimalizace
- oprava chyb, implementace nových požadavků, údržba
- dokumentace

Algoritmus a program

- Co je to *algoritmus?*
- přesný, detailní a úplný postup (obvykle řešení problému)
- Co je to *program?*
- zápis algoritmu v konkrétním programovacím jazyce

Python

- Programovací jazyk *Python* `http://www.python.org`
  - autor Guido van Rossum, 1989

Figure 1: Guido van Rossum

Proč Python?

- jazyk vysoké úrovně, pro všeobecné použití
- dobře čitelný, multiparadigmatický
- mnoho knihoven
- velmi populární (5. místo)
- *dynamický, interpretovaný (byte-code)*
- *s automatickou alokací paměti*

Budeme používat Python 3.
Toto není kurz jazyka Python. Detaily najdete v dokumentaci.

Jak Python spustíme?

Napíšeme do příkazové řádky `python3`:

```
!python3

Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more i
>>>
```

(nebo spustíme IDE prostředí jako `idle`, nebo Jupyter notebook...
Možností je mnoho)

Python jako kalkulačka

```
>python3
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more i
>>> 3+8
11
>>> 11*(5+3)
88
>>> 128./16.
8.0
>>> 2**16
65536
```

Python jako kalkulačka (2)

totéž, hezky vysázeno

```
3+8
```

```
11
```

```
11*(5+3)
```

```
88
```

```
128./16.
```

```
8.0
```

```
2**16
```

```
65536
```

Výrazy

- **Výrazy** *(expressions)* obsahují
  - Celá čísla: `3`, `8`, ...
  - Reálná čísla: `128.`, `11.5`,...
  - Operátory: `+`, `-`, `/`, `*`, ...
  - Oddělovače: `(`, `)`

Co se děje v zákulisí *(REPL)*

- Spustili jsme program `python3`, *interpret* Pythonu
  - tisk výzvy *(prompt)* >>>
  - přečtení uživatelského vstupu *(read)*
  - vyhodnocení výrazu *(evaluate)*
  - tisk výsledku *(print)*
- Opakované vykonávání (smyčka, *loop*)
- **REPL** *(read-eval-print-loop)*

Program jako transformace (filtr)

Figure 2: Transformace vstupu na výstup

Toky dat *(data flow)*

Proměnné a přiřazení

Hodnotu výrazu lze uložit pro pozdější použití

identifikátor = *výraz*

Příklad:

```
a=3
b=3+a
```

Jaká je hodnota proměnné *b*?

```
b
```

```
6
```

Příklad: Proměnné

```
boys=15
girls=17
total=boys+girls
difference=girls-boys
ratio=boys/total
total
```

```
32
```

```
difference
```

```
2
```

```
ratio
```

```
0.46875
```

Učte se anglicky. Počítače mluví anglicky, programy jsou anglicky, informace jsou anglicky.
Z důvodů přenositelnosti je bezpečnější se omezit na znaky anglické abecedy.

Hodnoty proměnných lze měnit

```
a=10
a=a-2
a=a*2
```

Jaká je hodnota *a*?

```
a
```

```
16
```

Není-li pro to dobrý důvod, hodnoty proměnných neměňte.

Proč používat proměnné

- **DRY** = *Do not repeat yourself.*
- Šetřme si práci, neopakujme se
- Zlepšení
  - **Srozumitelnosti** - smysluplná jména proměnných
  - **Údržby** - jedna změna jen na jednom místě
  - **Efektivity** - využijeme předchozích výpočtů

Začínáme programovat

První program - *Hello world*

```
# Vypíše pozdravení
print("Hello world")
```

- vytvoříme v textovém editoru
- uložíme do souboru `hello_world.py`
- spustíme (z příkazové řádky, opakovaně)

```
>python3 hello_world.py
```

```
!python3 hello_world.py
```

```
Hello world
```

První řádka je komentář - každý program má být komentován.

Editory, integrovaná prostředí (IDE)

- **Editory:** Emacs, gEdit, joe, ...
- spouštění
  - z příkazové řádky
  - z integrovaného prostředí
  - z Jupyter notebooku
- **Integrované prostředí:** IDLE, PyCharm, Eclipse, ...

Na konkrétním editoru, případně IDE, zase tolik nezáleží.

Editace a interpretace programu

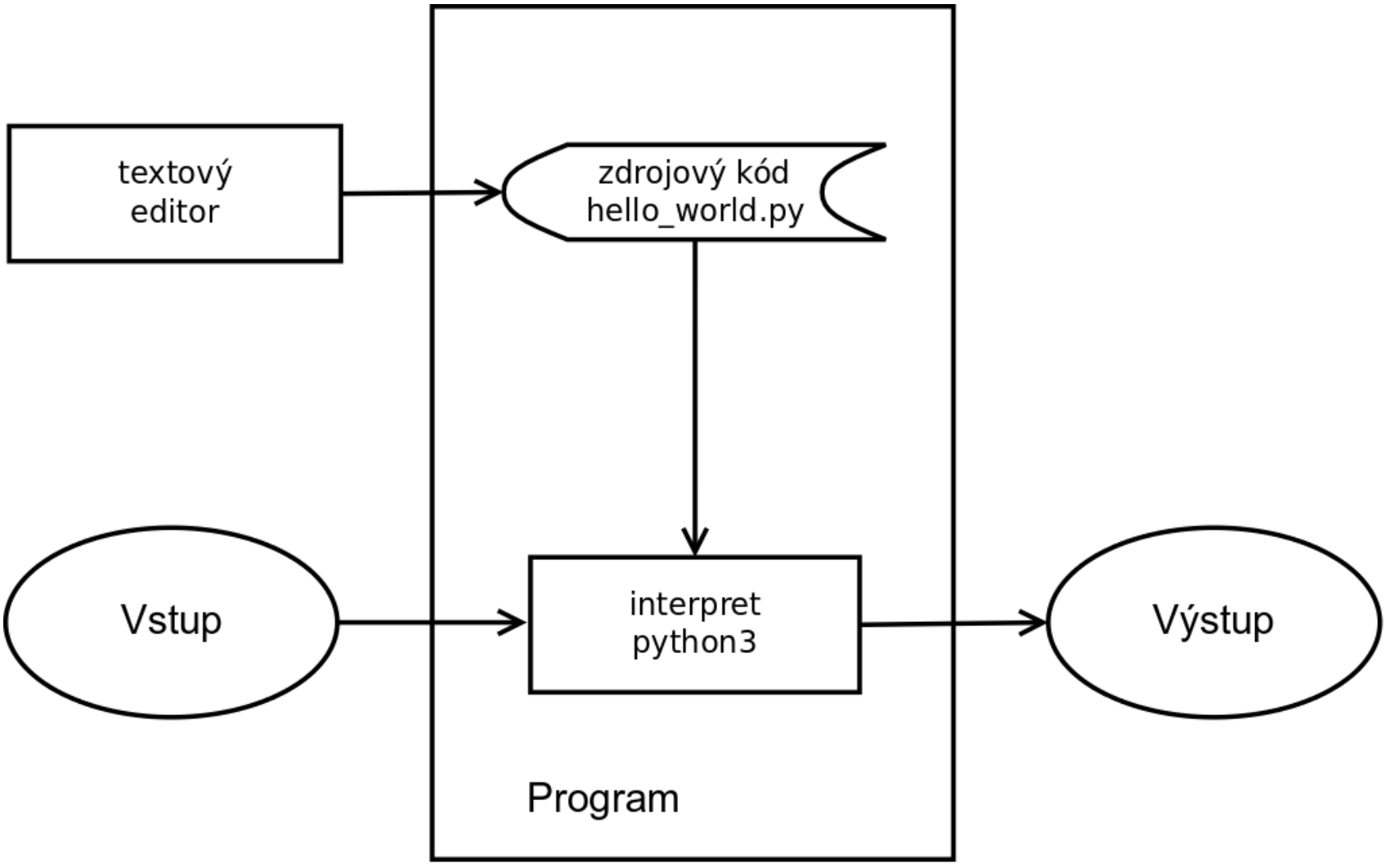

Figure 3: Editace a interpretace programu

Příklad: Převod stupňů Fahrenheita na stupně Celsia

Kolik $^\circ C$ je $75^\circ F$?

```
f=75
c=(f-32)*5./9.
print(c)
```

```
23.88888888888889
```

Trochu hezčí výpis *(pro pokročilé)*:

```
print(f,"stupňů Fahrenheita je",c,"stupňů Celsia.")
```

```
75 stupňů Fahrenheita je 23.88888888888889 stupňů Celsia.
```

```
print("%f stupňů Fahrenheita je %f stupňů Celsia." % (f,c))
```

```
75.000000 stupňů Fahrenheita je 23.888889 stupňů Celsia.
```

- *funkce* `print` vytiskne své argumenty
- argumentem funkce `print` může být číslo nebo *řetězec*
- `%f` do řetězce doplní reálná čísla z dalších argumentů
- Omezíme počet desetinných míst *(pro pokročilé)*:

```
print("%0.1f stupňů Fahrenheita je %0.1f stupňů Celsia." %
```

Program = automatizace

Co když chceme převést hodnot více? Kolik $^\circ C$ je $30^\circ F$?

- Šetřme si práci, neopakujme se
- **DRY** = *Do not repeat yourself*
- Vytvoříme program, který budeme moci opakovaně spouštět

Program = soubor

Do souboru `convert1.py` uložíme jednotlivé příkazy:

```
# Program pro převod stupňů Fahrenheita na stupně Celsia
f=75
c=(f-32)*5./9.
print("%0.1f stupňů Fahrenheita je %0.1f stupňů Celsia." %
```

a spustíme z příkazové řádky (i opakovaně)

```
>python3 convert1.py
```

```
!python3 convert1.py
```

```
75.0 stupňů Fahrenheita je 23.9 stupňů Celsia.
```

Čtení parametru z příkazové řádky

Náš program počítá pořád to samé... není flexibilní.

Vylepšená verze (convert2.py):

```
# Program convert2.py pro převod stupňů Fahrenheita na stup
import sys

f=int(sys.argv[1]) # první argument
c=(f-32)*5./9.
print("%0.1f stupňů Fahrenheita je %0.1f stupňů Celsia." %
```

Argument (stupně Fahrenheita) zadáme pří spouštění:

```
!python3 convert2.py 60
```

```
60.0 stupňů Fahrenheita je 15.6 stupňů Celsia.
```

```
!python3 convert2.py 90
```

```
90.0 stupňů Fahrenheita je 32.2 stupňů Celsia.
```

```
!python3 convert2.py -20
```

```
-20.0 stupňů Fahrenheita je -28.9 stupňů Celsia.
```

Základní části textu programů

- **Komentáře**:

  # Program convert2.py pro převod stupňů Fahrenheita na

- **Klíčová slova**: `import`

```
import keyword
print(keyword.kwlist)

['False', 'None', 'True', 'and', 'as', 'assert', 'break',
```

- **Identifikátory:** (jména proměnných a funkcí) `f`, `c`, `print`, ...
  Písmena, čísla, podtržítka, nezačíná číslem,není klíčové slovo
- **Operátory:** +,-,*,/,=,...
- **Literály:**
  - Celá čísla: 32,-20,...
  - Reálná čísla: `5.`, `9.`, `32.3`, `1.3e-6` ($1.3 \cdot 10^{-6}$)...
  - Řetězce: `"Hello world"`, `'xyz'`,`"%0.1f stupňů Fahrenheita je %0.1f stupňů Celsia."`...

Chyby

(neúplný přehled)

- **Syntaktické chyby** *(syntax errors)*- nejedná se o korektně zapsaný program v Pythonu, např:

```
c=(f-32*5./9.
```

```
  File "<ipython-input-9-6c154279ea5c>", line 1
    c=(f-32*5./9.
                ^
SyntaxError: unexpected EOF while parsing
```

- nedefinované jméno funkce nebo proměnné

```
prnt("%0.1f stupňů Fahrenheita je %0.1f stupňů Celsia." % (
```

```
---------------------------------------------------------------

NameError                                 Traceback (most r

<ipython-input-8-2e2cdc5aa55f> in <module>()
----> 1 prnt("%0.1f stupňů Fahrenheita je %0.1f stupňů Cels
```

Řídící struktury (control structures)

Porovnávání (čísel)

```
8>3
```

```
True
```

```
10<=10
```

```
True
```

```
1==0
```

```
False
```

```
2!=3
```

```
True
```

```
a=4
b=6
a<b
```

```
True
```

Operátory >,<,==,>=,<=,!=. Vracejí `True` nebo `False` *(typ `bool`)*.

Podmíněný příkaz *(if)*

```
# Program conditionals.py pro demonstraci podmíněných příkaz
import sys
n=int(sys.argv[1]) # první argument - celé číslo
if n>0:
    print(n,"je kladné číslo")
print("Konec programu.")
```

```
!python3 conditionals.py 10

10 je kladné číslo
Konec programu.

!python3 conditionals.py -1

Konec programu.
```

Bloky kódu jsou v Pythonu určené odsazením.
Bloky kódu = základ strukturovaného programování.

Větvení *(if-else)*

```
# Program conditionals2.py pro demonstraci podmíněných pří
import sys
n=int(sys.argv[1]) # první argument
if n>0:
    print(n,"je kladné číslo")
else:
    print(n,"není kladné číslo")
```

```
!python3 conditionals2.py 14

14 je kladné číslo

!python3 conditionals2.py -3

-3 není kladné číslo
```

Vnořené větvení

```
# Program conditionals3.py pro demonstraci podmíněných pří
import sys
n=int(sys.argv[1]) # první argument
if n>0:
    print(n,"je kladné číslo")
else:
  if n==0:
     print(n,"je nula")
  else:
     print(n,"je záporné číslo")
!python3 conditionals3.py 14

14 je kladné číslo

!python3 conditionals3.py -3

-3 je záporné číslo

!python3 conditionals3.py 0

0 je nula
```

Zřetězené podmínky *(if-elif-else)*

```
# Program conditionals4.py pro demonstraci podmíněných pří
import sys
n=int(sys.argv[1]) # první argument
if n>0:
    print(n,"je kladné číslo")
elif n==0:
    print(n,"je nula")
else:
    print(n,"je záporné číslo")
```

```
!python3 conditionals4.py 14
```

```
14 je kladné číslo
```

```
!python3 conditionals4.py -3
```

```
-3 je záporné číslo
```

```
!python3 conditionals4.py 0
```

```
0 je nula
```

(Příklad:) Maximum tří čísel

Vytiskněte maximum tří vstupních čísel.

```
# maximum.py - Vytiskne maximum tří zadaných čísel
import sys
a=int(sys.argv[1])
b=int(sys.argv[2])
c=int(sys.argv[3])
if a>b: # maximum je a nebo c
  if a>c: # a>b, a>c
    print(a)
  else:   # c >= a > b
    print(c)
else:     # b >= a
  if b>c: # b > c, b >= a
    print(b)
  else:   # c >= b >= a,
    print(c)
!python3 maximum.py 10 29 3
```

(Příklad:) Kontrola prázdného vstupu

```
!python3 conditionals4.py
```

```
Traceback (most recent call last):
  File "conditionals4.py", line 3, in <module>
    n=int(sys.argv[1]) # první argument
IndexError: list index out of range
```

- Zkontrolujeme počet vstupních argumentů
- `sys.argv` - seznam vstupních parametrů
- `len(sys.argv)` - počet prvků seznamu = počet parametrů + 1
- `sys.argv[0]` - nultý parametr = jméno programu (např. "conditionals4.py")
- `sys.argv[1]` - první parametr = první uživatelský argument

```
# Program conditionals5.py pro demonstraci podmíněných pří
import sys
if len(sys.argv)!=2:
  print("Zadej cele cislo")
else:
```

Předčasný návrat

```
# Program conditionals6.py pro demonstraci podmíněných pří
import sys
if len(sys.argv)<=1:
  print("Zadej jedno cele cislo")
  sys.exit() # ukončí program

# zde následuje původní program
n=int(sys.argv[1]) # první argument
if n>0:
  print(n,"je kladné číslo")
elif n==0:
  print(n,"je nula")
else:
  print(n,"je záporné číslo")
```

Funguje stejně jako `conditionals5.py`.
`sys.exit()` ukončí celý program. Jak ukončit funkce a cykly se naučíme později.

Cykly (smyčky) *(loops)*

- Smyčka slouží k opakování části (bloku) programu
  - daný počet opakování *(for)*
  - pro všechny elementy z dané sekvence *(for)*
  - dokud platí podmínka *(while)*

```
for i in range(10):
    print("Budu se pilně učit.")
```

```
Budu se pilně učit.
Budu se pilně učit.
Budu se pilně učit.
Budu se pilně učit.
Budu se pilně učit.
Budu se pilně učit.
Budu se pilně učit.
Budu se pilně učit.
Budu se pilně učit.
Budu se pilně učit.
```

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

for cyklus

Proměnná v příkazu

```
for <promenna> in range(n):
    <blok>
```

nabírá postupně hodnoty $0 \ldots n-1$:

```
for i in range(5):
    print("i=",i)
```

```
i= 0
i= 1
i= 2
i= 3
i= 4
```

Funkce `range` může mít i parametry `start` a `step`.
help(range)
Funkce `help` ukáže nápovědu k dané funkci či příkazu. Zkuste si `help()`.
Nastavení počáteční hodnoty cyklu:

```
for i in range(1,5):
```

Příklad: Tabulka Fahrenheit - Celsius

```
for f in range(0,110,10):
    c=(f-32)*5./9.
    print("%5.1fF = %5.1fC" % (f,c))
```

```
  0.0F = -17.8C
 10.0F = -12.2C
 20.0F =  -6.7C
 30.0F =  -1.1C
 40.0F =   4.4C
 50.0F =  10.0C
 60.0F =  15.6C
 70.0F =  21.1C
 80.0F =  26.7C
 90.0F =  32.2C
100.0F =  37.8C
```

- *Soubor `tabulka_fahrenheit.py`*
- *Napříště už uložení do souboru a spuštění zdůrazňovat nebudeme.*

Příklad: Součet čísel

Vypočítejte $\sum_{i=1}^{100} i$:

```
s=0
for i in range(1,101):
    s=s+i
print("Soucet je ",s)
```

```
Soucet je  5050
```

Kontrola:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Vnořené bloky a cykly

Blok kódu může obsahovat další (vnořené) bloky.

Příklad: násobilka (soubor `nasobilka.py`)

```
n=5
for i in range(1,n+1):
    for j in range(1,n+1):
        print("%2d * %2d = %2d" % (i,j,i*j))
```

```
 1 *  1 =  1
 1 *  2 =  2
 1 *  3 =  3
 1 *  4 =  4
 1 *  5 =  5
 2 *  1 =  2
 2 *  2 =  4
 2 *  3 =  6
 2 *  4 =  8
 2 *  5 = 10
 3 *  1 =  3
```

Smyčka *while*

Iteruje, dokud je podminka splněná

```
while <podminka>:
    <blok>
```

```
i=5
while i>0:
    print("i=",i)
    i=i-1
```

```
i= 5
i= 4
i= 3
i= 2
i= 1
```

Cyklus `while` může nahradit cyklus `for`, ale nikoliv naopak

Odbočka: Operátor celočíselného dělení a modulo

```
10/3
```

```
3.3333333333333335
```

```
10//3
```

```
3
```

a zbytek po dělení (operace *modulo*)

```
10%3
```

```
1
```

Vždy platí: `(n//k)*k+(n%k)=n`

```
(100//7)*7+(100%7)
```

```
100
```

Příklad: Kolik číslic má dané přirozené číslo?

Myšlenka: Spočítáme, kolikrát lze dělit deseti, než se dostaneme k nule.

```
c=1 # počet číslic
while n>10:
  n=n//10 # celočíselné dělení
  c=c+1
print("Počet číslic=%d" % c)
```

Celý program včetně kontroly vstupů:

```
# %load pocet_cislic.py
# Spočítej, kolik číslic má dané přirozené číslo
import sys

if len(sys.argv)!=2:
  print("Chyba: zadej jedno přirozené číslo.")
  sys.exit()

n=int(sys.argv[1])
```

Zkrácené přiřazení

Místo c=c+1 píšeme c+=1. Totéž funguje i pro další operátory.
Místo:

```
c=1 # počet číslic
while n>10:
  n=n//10 # celočíselné dělení
  c=c+1
print("Počet číslic=%d" % c)
```

napíšeme *(soubor pocet_cislic2.py)*

```
c=1 # počet číslic
while n>10:
  n//=10 # celočíselné dělení
  c+=1
print("Počet číslic=%d" % c)
```

Nekonečný cyklus

Vinou chyby program nikdy neskončí.

```
n=982
c=1 # počet číslic
while n>10:
  n//10 # celočíselné dělení
  c+=1
print("Počet číslic=%d" % c)
```

- Nekonečný cyklus může být i záměrný.
- Snažme se *dokázat*, že program skončí.

Přerušení cyklu *(break, continue)*

V těle cyklu `for` nebo `while`:

- `break` ukončí celý cyklus
- `continue` přeruší aktuální iteraci a začne následující

```
for i in range(5):
    if i==3:
        break
    print(i)
```

```
0
1
2
```

```
for i in range(5):
    if i==3:
        continue
    print(i)
```

```
0
1
2
```

Příklad: Test prvočíselnosti

prvočíslo $n > 1$ je dělitelné pouze 1 a n.
$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, \ldots$
Test dělitelnosti v Pythonu:

```
10 % 3 == 0
```

```
False
```

```
10 % 5 == 0
```

```
True
```

Úkol 1: Napište program, který zjistí, zda je zadané číslo prvočíslo. Zkusím dělit zadané číslo n čísly $p \in 2 \ldots n-1$, jestli $p \mid n$. *(soubor `prvocislo1.py`)*

```
# Test prvočiselnosti zadaného čísla
import sys
n=int(sys.argv[1]) # číslo, které testujeme
p=2
while p<n:
    if n % p == 0:
```

Optimalizace - nešlo by to rychleji?

- Hledáme algoritmus, který je **správný**.
- Hledáme algoritmus, který je **rychlý**.

Myšlenka: Stačí testovat pro $p \leq \sqrt{n}$, neboť pokud $n = ab$, pak buď $a \leq \sqrt{n}$ nebo $b \leq \sqrt{n}$.

```
# prvocislo3.py - Vypíše prvočísla menší než zadaný limit
import sys
m=int(sys.argv[1])
for n in range(2,m): # cyklus 2..m-1
  p=2 # začátek testu
  while p*p<=n:
    if n % p == 0:
        break
    p+=1
  if p*p > n: # n je prvočíslo
    print(n,end=", ")
print() # závěrečný konec řádky
```

```
!python3 prvocislo3.py 1000
```

Závěr

Co jsme se naučili

- Proč se učit programovat
- Program, algoritmus, programovací jazyk
- Python, jak ho spustit
- Jak spustit program ze souboru
- Čísla, výrazy, proměnné
- Vstupní argumenty, výstup
- Podmíněné příkazy (`if`,`else`)
- Smyčky (`for`, `while`, `break`, `continue`)
- Chyby
- Optimalizace