Úvod do jazyka Python (2/2)

Jan Kybic

http://cmp.felk.cvut.cz/~kybic
kybic@fel.cvut.cz

2016–2020

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
```

Chyby (2)

Nedefinované jméno funkce nebo proměnné

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

---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

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


NameError: name 'prnt' is not defined
```

Chyby (3)

- Chybám se nevyhnete.
- Možných chyb je velmi mnoho...
- Chybová hlášení nám pomáhají chybu najít.
- Postupně se naučíme, jak dělat chyb méně.

Chybný vstup

```
>python3 convert2.py 20c
```

Chybný vstup

```
>python3 convert2.py 20c

Traceback (most recent call last):
  File "convert2.py", line 4, in <module>
    f=int(sys.argv[1]) # první argument
ValueError: invalid literal for int() with base 10: '20c'
```

Program skončil chybou = "spadnul" *(crashed)*

Chybný vstup (2)

- Dobrý program vstupy kontroluje.
- Dobrý program umí s chybnými daty pracovat, reaguje na chyby srozumitelnou informací uživateli.
- To se naučíte postupně.
- Jako začátečníci předpokládejte, že vstupy jsou správné.

Logická/významová chyba

(bug) - program běží, ale dává špatné výsledky

```
c=(f-32)+5./9.
print("%0.1f stupňů Fahrenheita je %0.1f stupňů Celsia." %
      (f,c))
```

```
75.0 stupňů Fahrenheita je 43.6 stupňů Celsia.
```

Odstranit tento druh chyb je nejnáročnější.

Řídící struktury
(control structures)

- Pokud chceme změnit posloupnosti vykonávání příkazů
- Podmíněné příkazy — `if`, `else`
- Smyčky — `for`, `while`
  - Přerušení a návrat — `break`, `continue`

Porovnávání (čísel)

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

Porovnávání (2)

```
8>3
```

```
True
```

```
10<=10
```

```
True
```

```
1==0
```

```
False
```

```
2!=3
```

```
True
```

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

```
True
```

Podmíněný příkaz *(if)*

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

Podmíněný příkaz *(if)*

```
# Program conditionals.py pro demonstraci podmíněných příke
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.
```

Podmíněný příkaz (*if*)

```
# Program conditionals.py pro demonstraci podmíněných příka
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.
```

Podmíněný příkaz (*if*)

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

Větvení (*if-else*)

```
# Program conditionals2.py pro demonstraci podmíněných příkazů
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 příkazů
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")
```

Vnořené větvení

```
# Program conditionals3.py pro demonstraci podmíněných příkazů
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
```

Vnořené větvení

```
# Program conditionals3.py pro demonstraci podmíněných příkazů
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
```

Vnořené větvení

```
# Program conditionals3.py pro demonstraci podmíněných příkazů
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 příkazů
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)
```

(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
```

```
29
```

(Příklad:) Maximum tří čísel (2)

Takové funkce už v Pythonu samozřejmě jsou...

```
max(10,29,3)
```

```
29
```

(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
```

(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

(Příklad:) Kontrola prázdného vstupu (2)

```
# Program conditionals5.py pro demonstraci podmíněných příkazů
import sys
if len(sys.argv)!=2:
  print("Zadej cele cislo")
else:
  n=int(sys.argv[1]) # první argument
  # zde následuje původní kód
  if n>0:
    print(n,"je kladné číslo")
  elif n==0:
    print(n,"je nula")
  else:
    print(n,"je záporné číslo")
```

(Příklad:) Kontrola prázdného vstupu (2)

```
# Program conditionals5.py pro demonstraci podmíněných příkazů
import sys
if len(sys.argv)!=2:
  print("Zadej cele cislo")
else:
  n=int(sys.argv[1]) # první argument
  # zde následuje původní kód
  if n>0:
    print(n,"je kladné číslo")
  elif n==0:
    print(n,"je nula")
  else:
    print(n,"je záporné číslo")
```

```
>python3 conditionals5.py 1
```

```
1 je kladné číslo
```

(Příklad:) Kontrola prázdného vstupu (2)

```
# Program conditionals5.py pro demonstraci podmíněných příkazů
import sys
if len(sys.argv)!=2:
  print("Zadej cele cislo")
else:
  n=int(sys.argv[1]) # první argument
  # zde následuje původní kód
  if n>0:
    print(n,"je kladné číslo")
  elif n==0:
    print(n,"je nula")
  else:
    print(n,"je záporné číslo")
```

```
>python3 conditionals5.py 1

1 je kladné číslo

>python3 conditionals5.py

Zadej cele cislo
```

Předčasný návrat

```
# Program conditionals6.py pro demonstraci podmíněných příkazů
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`)*

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.")
```

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 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*

Funkce `range` může mít i parametry `start` a `step`.

```
help(range)
```

Funkce `help` ukáže nápovědu k dané funkci či příkazu. Zkuste si `help()`.

Funkce *range* (2)

Nastavení počáteční hodnoty cyklu:

```
for i in range(1,5):
    print("i=",i)
```

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

Funkce *range* (2)

Nastavení počáteční hodnoty cyklu:

```
for i in range(1,5):
    print("i=",i)
```

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

Nastavení kroku 3, počáteční číslo 1, horní hranice 10:

```
for i in range(1,10,3):
    print("i=",i)
```

```
i= 1
i= 4
i= 7
```

Funkce *range* (3)

Můžeme počítat i sestupně (počáteční hodnota 5, krok 1, spodní hranice 0):

```
for i in range(5,0,-1):
    print("i=",i)
```

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

Příklad: Tabulka Fahrenheit - Celsius

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

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
```

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)
```

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
```

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))
```

Násobilka (2)

```
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
3 *  2 =  6
3 *  3 =  9
```

```
3 *  4 = 12
3 *  5 = 15
4 *  1 =  4
4 *  2 =  8
4 *  3 = 12
4 *  4 = 16
4 *  5 = 20
5 *  1 =  5
5 *  2 = 10
5 *  3 = 15
5 *  4 = 20
5 *  5 = 25
```

Smyčka *while*

Iteruje, dokud je podminka splněná

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

Smyčka *while*

Iteruje, dokud je podminka splněná

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

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

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
```

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
```

Odbočka: Operátor celočíselného dělení a modulo

```
10/3
```

```
3.3333333333333335
```

```
10//3
```

```
3
```

Odbočka: Operátor celočíselného dělení a modulo

```
10/3
```

```
3.3333333333333335
```

```
10//3
```

```
3
```

a zbytek po dělení (operace *modulo*)

```
10%3
```

```
1
```

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?

Příklad: Kolik číslic má dané přirozené číslo?

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

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)
```

Příklad: Kolik číslic má dané přirozené číslo? (2)

Celý program včetně kontroly vstupů (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])

if n<1:
  print("Chyba:",n,"není přirozené číslo!")
  sys.exit()

print("Bylo zadáno číslo", n)

# tady začíná vlastní výpočet
```

Příklad: Kolik číslic má dané přirozené číslo? (3)

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

Příklad: Kolik číslic má dané přirozené číslo? (3)

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

Pro přehlednost budeme kontroly v přednáškách občas vynechávat.
Ve skutečných programech je používejte.

Zkrácené přiřazení

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

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

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*)

V těle cyklu `for` nebo `while`:

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

Přerušení cyklu *(break)*

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)
```

Přerušení cyklu *(break)*

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
```

Přerušení cyklu *(continue)*

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

Přerušení cyklu *(continue)*

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

```
0
1
2
4
```

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
```

Příklad: Test prvočíselnosti (2)

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`*)*

Příklad: Test prvočíselnosti (2)

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:
        break
    p+=1
if p<n:
    print(n, "není prvočíslo, je dělitelné", p)
else:
    print(n, "je prvočíslo")
```

Příklad: Test prvočíselnosti (3)

```
>python3 prvocislo1.py 17

17 je prvočíslo
```

Příklad: Test prvočíselnosti (3)

```
>python3 prvocislo1.py 17

17 je prvočíslo

>python3 prvocislo1.py 15

15 není prvočíslo, je dělitelné 3
```

Příklad: Více prvočísel

Úkol 2: Napište program, vypíše všechna prvočísla menší než *m*. *(soubor `prvocislo2.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<n:
    if n % p == 0:
        break
    p+=1
  if p==n: # n je prvočíslo
    print(n,end=", ")
print() #  závěrečný konec řádky
```

Příklad: Více prvočísel

Úkol 2: Napište program, vypíše všechna prvočísla menší než *m*. *(soubor `prvocislo2.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<n:
    if n % p == 0:
        break
    p+=1
  if p==n: # n je prvočíslo
    print(n,end=", ")
print() #  závěrečný konec řádky
```

Poznámka: *`print(n)` místo `print(n,end=", ")` by psalo každé číslo na vlastní řádku*

Příklad: Více prvočísel (2)

```
>python3 prvocislo2.py 1000

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107,
109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229,
233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283,
293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359,
367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431,
433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491,
499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571,
577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641,
643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709,
719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787,
797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859,
863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941,
947, 953, 967, 971, 977, 983, 991, 997,
```

Optimalizace - nešlo by to rychleji?

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

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 $b \leq \sqrt{n}$.

Optimalizace (2)

```
# 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
```

Optimalizace (3)

```
>python3 prvocislo3.py 1000

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107,
109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229,
233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283,
293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359,
367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431,
433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491,
499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571,
577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641,
643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709,
719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787,
797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859,
863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941,
947, 953, 967, 971, 977, 983, 991, 997,
```

Optimalizace (4)

Změříme čas pro $m = 10000$. *(Pro přehlednost potlačíme výstup)*

```
>time python3 prvocislo2.py 10000 >/dev/null
```

```
4.11user 0.01system 0:04.13elapsed 99%CPU (0avgtext+0avgdata 584
0inputs+0outputs (0major+1649minor)pagefaults 0swaps
```

Optimalizace (4)

Změříme čas pro $m = 10000$. *(Pro přehlednost potlačíme výstup)*

```
>time python3 prvocislo2.py 10000 >/dev/null

4.11user 0.01system 0:04.13elapsed 99%CPU (0avgtext+0avgdata 584
0inputs+0outputs (0major+1649minor)pagefaults 0swaps

>time python3 prvocislo3.py 10000 >/dev/null

0.17user 0.00system 0:00.17elapsed 98%CPU (0avgtext+0avgdata 585
0inputs+0outputs (0major+1650minor)pagefaults 0swaps
```

Vylepšená verze je $24\times$ **rychlejší**!

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