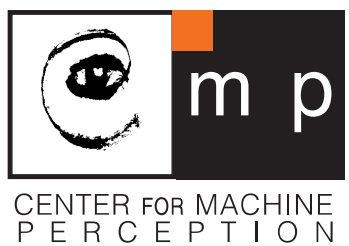
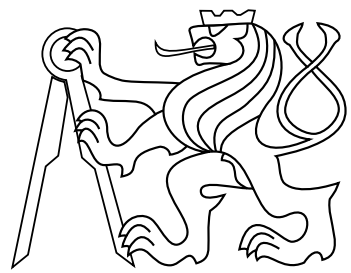
Konečný automat

Jan Kybic

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

2016–2017





Konečný automat
finite state machine

Konečný automat = výpočetní model, primitivní počítač

- Řídící jednotka s konečným počtem stavů
- Vstupní posloupnost — textový řetězec, fyzické vstupy
- Výstupní posloupnost, fyzické výstupy

Konečný automat
finite state machine

Konečný automat = výpočetní model, primitivní počítač

- Řídící jednotka s konečným počtem stavů
- Vstupní posloupnost — textový řetězec, fyzické vstupy
- Výstupní posloupnost, fyzické výstupy

Definice:

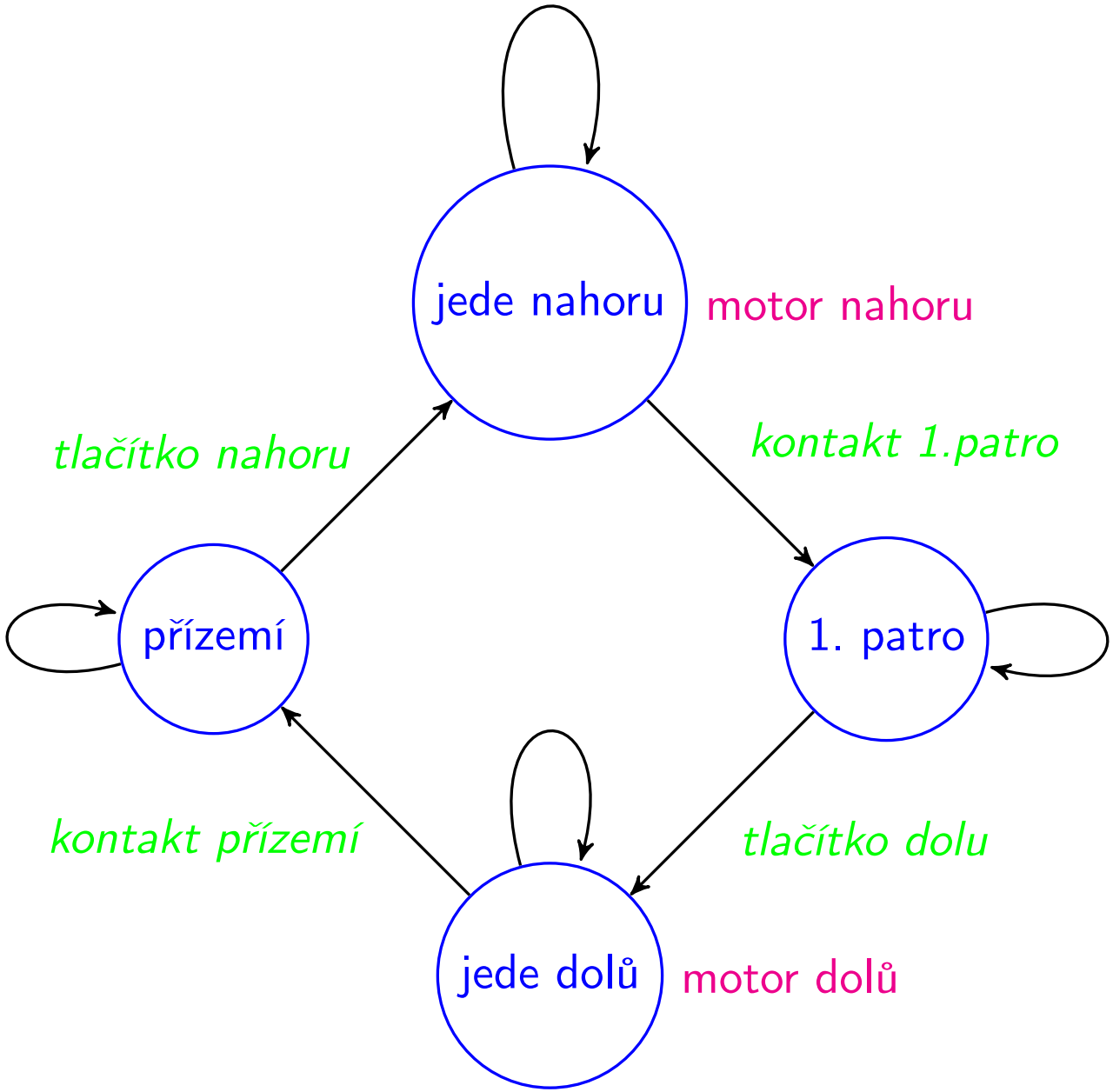
- konečná množina stavů Q
- počáteční stav $q_0 \in Q$
- konečná množina vstupních symbolů A
- přechodová funkce $f : Q \times A \to Q$

$$q_{t+1} = f(q_t, a_t)$$

  t je čas nebo index do vstupní posloupnosti
- (někdy) množina koncových stavů $F \subseteq Q$
- (někdy) množina akcí G a výstupní funkce $g : Y \to G$ nebo $g : Y \times A \to G$

Příklad: řízení výtahu

Stavový diagram



Příklad: řízení výtahu

Přechodová a výstupní tabulka

vstupy				stavy		výstupy	
Tl.d.	Tl.n.	Příz.	1.p.	q_t	q_{t+1}	$\uparrow$	$\downarrow$
?	0	?	?	přízemí	přízemí	0	0
?	1	?	?	přízemí	jede nahoru	1	0
?	?	?	0	jede nahoru	jede nahoru	1	0
?	?	?	1	jede nahoru	1. patro	0	0
0	?	?	?	1. patro	1. patro	0	0
1	?	?	?	1. patro	jede dolů	0	1
?	?	0	?	jede dolů	jede dolů	0	1
?	?	1	?	jede dolů	přízemí	0	0

Automat přijímající jazyk

Jazyk = (i nekonečná) množina řetězců

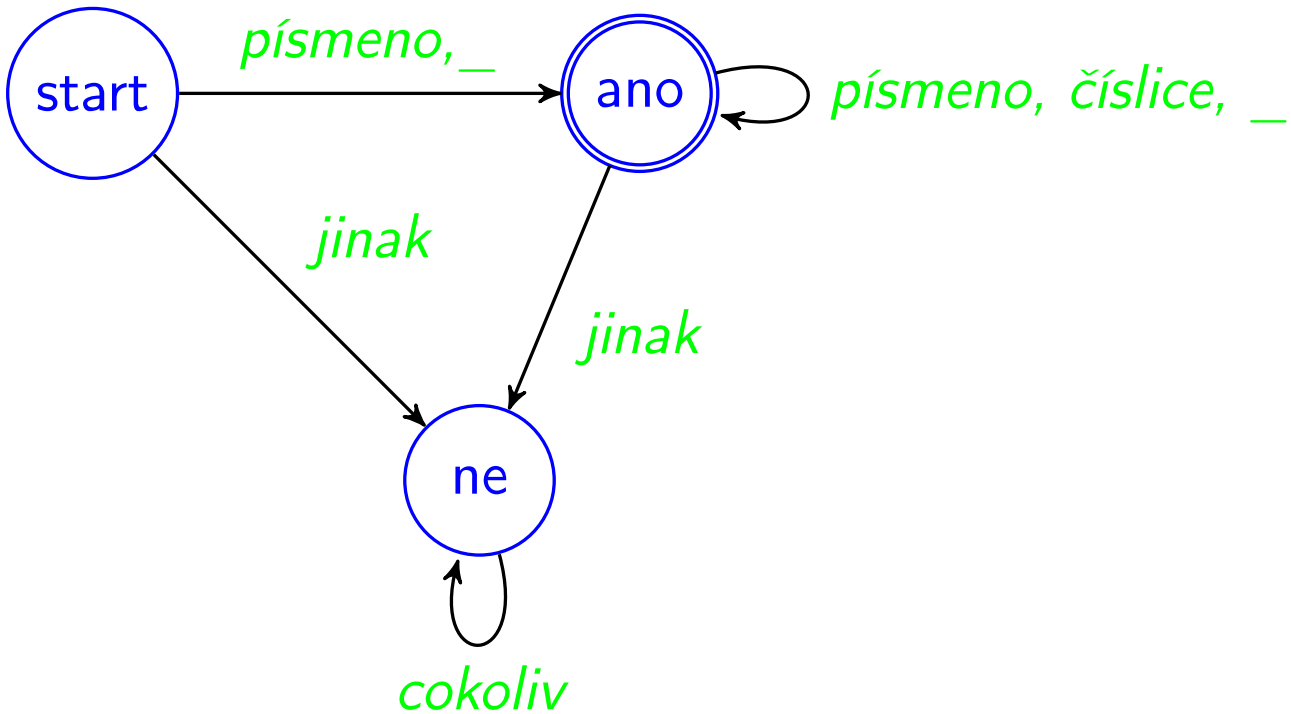
Rozhodovací konečný automat

- Vstupem je řetězec.
- Dává odpověď ano/ne, zda vstup patří do jazyka.
- V každém kroku čte automat jeden znak z řetězce.
- Vstupní abeceda A jsou všechny přípustné znaky.
- Automat přijímá řetězec $\Leftrightarrow$ na konci řetězce je automat v *přijímajícím (koncovém) stavu.*

Příklad: jméno proměnné

Totální automat

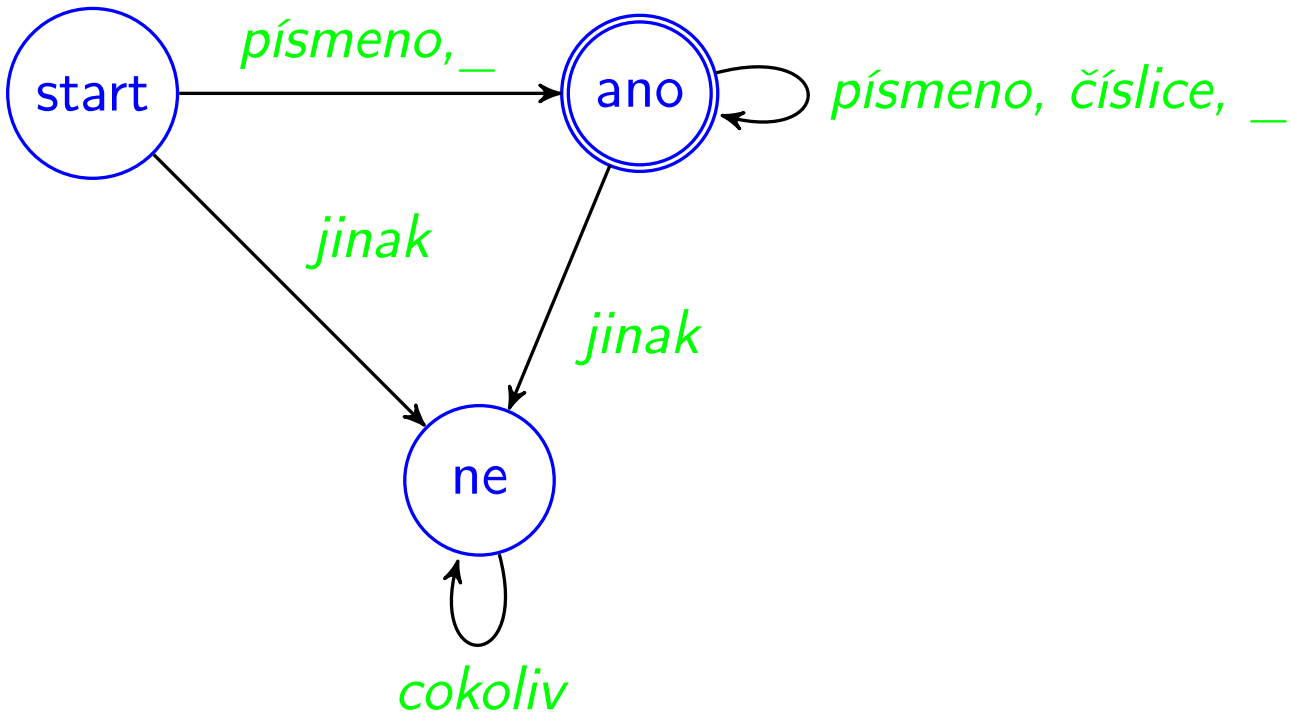
Automat přijímá platná jména proměnných v Pythonu.



Příklad: jméno proměnné

Totální automat

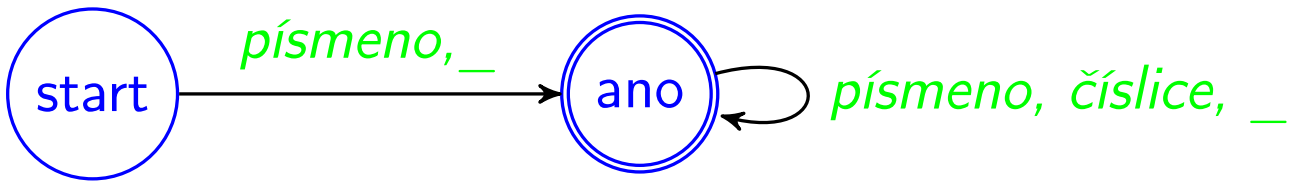
Automat přijímá platná jména proměnných v Pythonu.



- Automat je *totální* — přechodová funkce f je definována pro všechna $Q \times A$.

Příklad: jméno proměnné (2)

Částečný automat



- Částečný automat — nedefinovaný přechod znamená nepřijetí.

Jméno proměnné — implementace

```
def is_variable_name(s):
  """ je 's' platné jméno proměnné v Pythonu? """
  state="start"
  for c in s:
    if state=="start":
      if c=="_" or is_letter(c):
        state="ano"
      else:
        return False
    else: # stav "ano"
      if not (c=="_" or is_letter(c) or c.isdigit()):
        return False
  return state=="ano"

def is_letter(c):
  return c.isupper() or c.islower()
```

Soubor `automaton_examples.py`.

Jméno proměnné — příklady

```
print(is_variable_name("moje_promenna12"))
```

```
True
```

```
print(is_variable_name("moje{}_promenna"))
```

```
False
```

```
print(is_variable_name("12moje{}_promenna"))
```

```
False
```

Transformační konečný automat

- Vstupem je řetězec.
- Výstupem je řetězec.
- V každém kroku čte automat jeden znak z řetězce.
- Vstupní abeceda A jsou všechny přípustné znaky.
- Součásti každého přechodu (nebo stavu) může být výstup jednoho či více znaků.

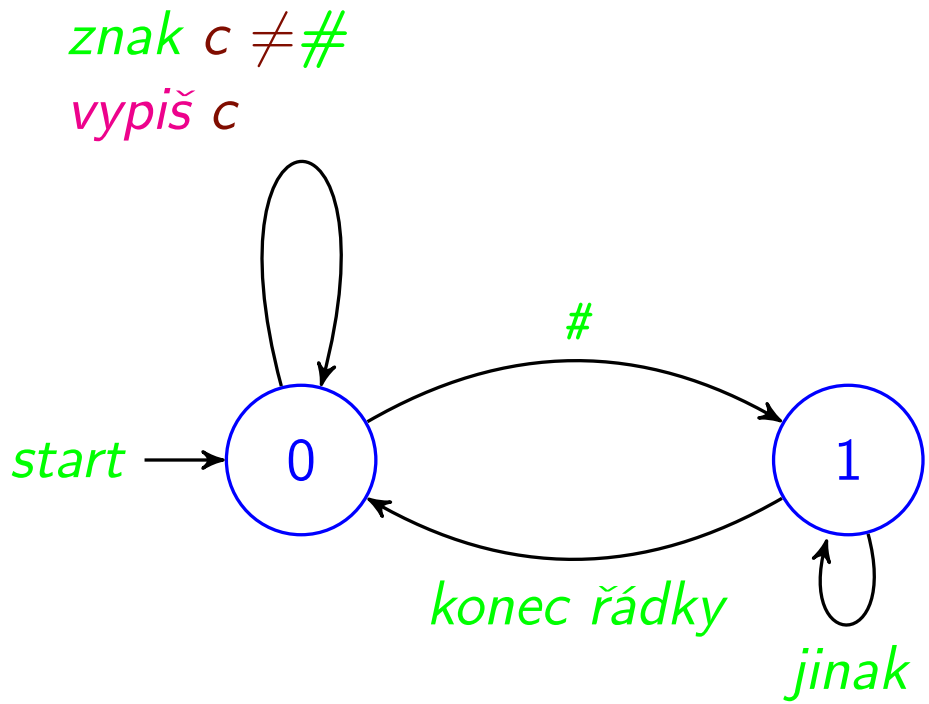
Příklad: přeskakování komentářů

Úkol:

- Vstupem je textový soubor
- Komentář je vše od znaku `#` (včetně) do konce řádky
- Vypište obsah souboru bez komentářů.

Soubor `preskoc_komentare.py`

Přeskakování komentářů

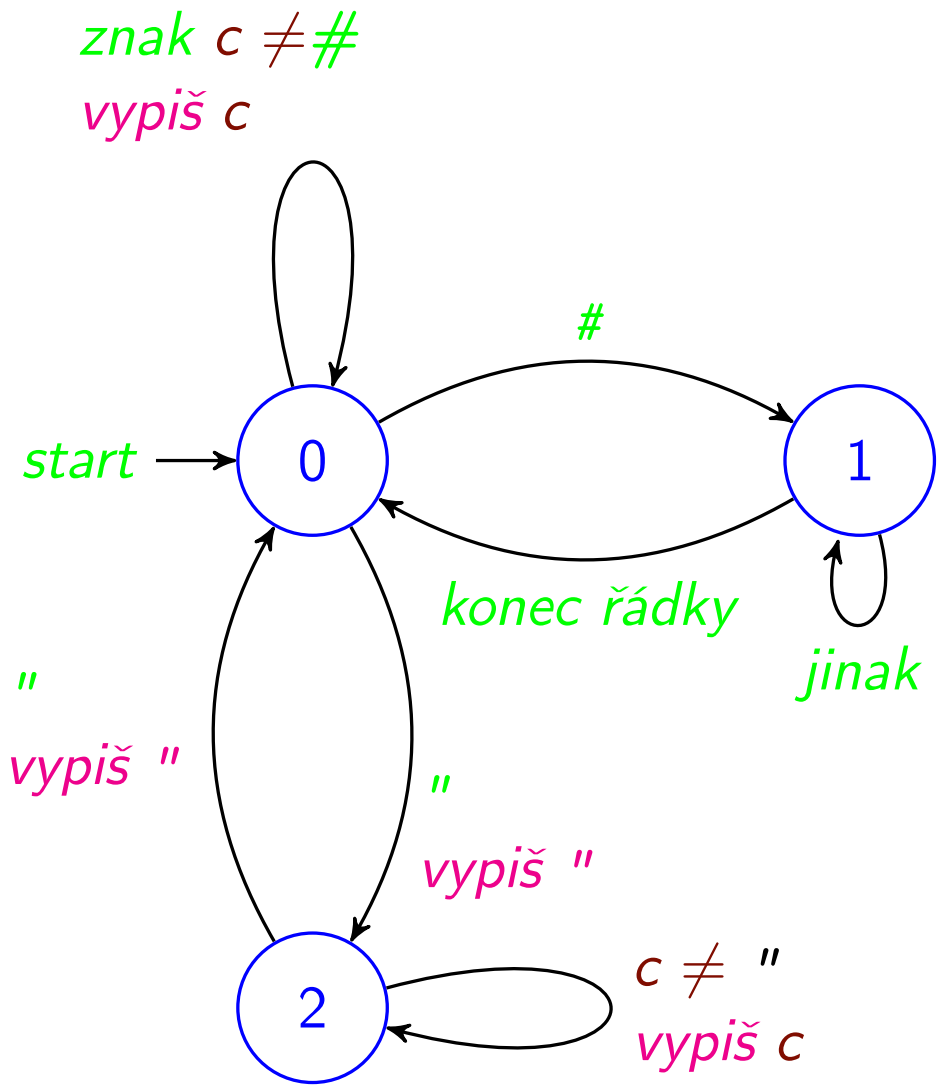


```python
# Tento program voláme s argumentem jméno souboru
# Vypíše obsah souboru bez Pythonovských komentářů

import sys

def preskoc_komentare(f):
  # vytiskne obsah souboru 'f' s vynechanymi komentari
  stav=0           # počáteční stav automatu
  while True:
    c=f.read(1)    # přečti jeden znak
    if c=="":      # konec souboru
      return
    if stav==0:    # počáteční stav
      if c=="#":   # začátek komentáře
        stav=1
      else:
        print(c,end="") # vytiskni znak
    else: # stav 1="komentar"
      if c=="\n": # konec řádky
        stav=0     # konec komentáře
        print(c,end="")
      else:        # vše ostatní ignorujeme
        pass

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f: # otevři textový soubor
    preskoc_komentare(f)  # pokud se povedlo, přeskakuj
```

```
Terminal> python3 preskoc_komentare.py preskoc_komentare.py


import sys

def preskoc_komentare(f):

  stav=0
  while True:
    c=f.read(1)
    if c=="":
      return
    if stav==0:
      if c=="
        stav=1
      else:
        print(c,end="")
    else:
      if c=="\n":
        stav=0
        print(c,end="")
      else:
        pass

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f:
    preskoc_komentare(f)
```

Přeskakování komentářů (2)

Verze, kterou nezmate # v řetězci.



```
# Verze kterou nezmate # mezi " "

import sys

def preskoc_komentare(f):
  # vytiskne obsah souboru 'f' s vynechanymi komentari
  stav=0           # počáteční stav automatu
  while True:
    c=f.read(1)    # přečti jeden znak
    if c=="":      # konec souboru
      return
    if stav==0:    # počáteční stav
      if c=="#":   # začátek komentáře
        stav=1
      else:
        if c=='"':
          stav=2     # začátek řetězce
        print(c,end="") # vytiskni znak
    elif stav==1: # 1="komentar"
      if c=="\n": # konec řádky
        stav=0     # konec komentáře
        print(c,end="")
      else:        # vše ostatní ignorujeme
        pass
    else:          # stav = řetězec
      if c=='"':
        stav=0
      print(c,end="") # vytiskni znak
```

```
Terminal> python3 preskoc_komentare2.py preskoc_komentare.py
```

```
import sys

def preskoc_komentare(f):

  stav=0
  while True:
    c=f.read(1)
    if c=="":
      return
    if stav==0:
      if c=="#":
        stav=1
      else:
        print(c,end="")
    else:
      if c=="\n":
        stav=0
        print(c,end="")
      else:
        pass

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f:
    preskoc_komentare(f)
```

```
Terminal> python3 preskoc_komentare2.py preskoc_komentare.py
```

```
import sys

def preskoc_komentare(f):

  stav=0
  while True:
    c=f.read(1)
    if c=="":
      return
    if stav==0:
      if c=="#":
        stav=1
      else:
        print(c,end="")
    else:
      if c=="\n":
        stav=0
        print(c,end="")
      else:
        pass

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f:
    preskoc_komentare(f)
```

Cvičení: doplňte řetězce oddělené ’ ’.

Lexikální analýza
(parsing)

- Vstupem je řetězec (posloupnost znaků)
- Výstupem je posloupnost symbolů *(tokens)* — číslo, operátor, identifikátor, klíčové slovo, ...
  - Symboly mohou mít atributy — řetězec, hodnota, ...
  - Reprezentujeme jako dvojice — typ symbolu + atribut
- Další operace
  - Vynechání komentářů
  - Odstranění mezer
  - Chybová hlášení

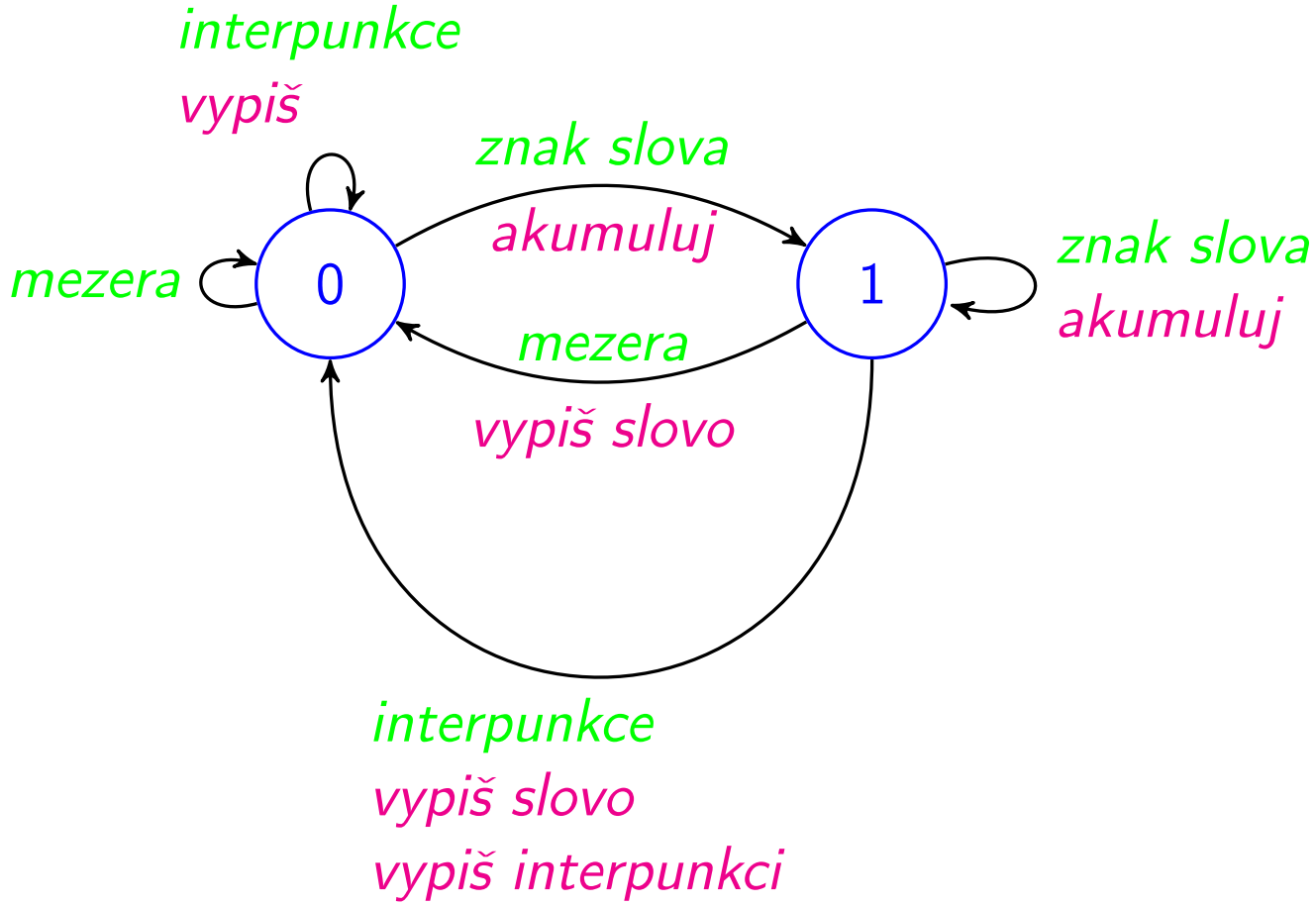
Příklad: Lexikální analýza textu

Rozdělte řetězec na posloupnost symbolů:

- *Interpunkce* — `.` `,` `;` `:` `?` `!` `“` `(` `)` (jeden znak)
- *Slovo* — řetězec znaků, neobsahující interpunkci, mezery ani konce řádků.
- Konce řádků, mezery a tabulátory *(whitespace)* ignorujte.

Soubor `analyza_textu.py`

Analýza textu



Počáteční stav $q_0 = 0$.

```
def analyzuj(f):
  interpunkce="""".,;:?!"'()"""
  mezery=" \t\n"
  stav=0 # počáteční stav - mimo slovo
  while True:
    c=f.read(1) # přečti jeden znak
    if c=="":   # konec souboru
      return
    if stav==0:
      if c in interpunkce:
        print("Interpunkce: ",c)
      elif c not in mezery:  # je to znak slova
        slovo=c                # akumulátor slova
        stav=1
    else:  # stav==1 - uvnitř slova
      if c not in interpunkce and c not in mezery:
        slovo+=c # pořád uvnitř slova
      else:      # slovo končí
        print("Slovo:       ",slovo)
        stav=0
        if c in interpunkce:
          print("Interpunkce: ",c)

if __name__=="__main__":
  analyzuj(sys.stdin)
```

```
Terminal> python3 analyza_textu.py


Kam jdeš? Stůj!


Slovo:         Kam
Slovo:         jdeš
Interpunkce:   ?
Slovo:         Stůj
Interpunkce:   !
```

Nedeterministický automat
(Non deterministic finite automaton — NFA)

- Přechodová funkce vrací množinu stavů, $f : Q \times A \to 2^Q$, automat si jeden nedeterministicky ‘vybere’.
- Slovo je přijímáno automatem, pokud existuje posloupnost výběrů vedoucí do koncového stavu.
- Implementace:
  - Při simulaci si udržujeme množinu možných stavů.
  - Automaticky převedeme na deterministický automat s 2^n stavy.
  - Často lze nalézt ekvivalentní deterministický automat s méně stavy.

Nedeterministický automat
(Non deterministic finite automaton — NFA)

- Přechodová funkce vrací množinu stavů, $f : Q \times A \to 2^Q$, automat si jeden nedeterministicky ‘vybere’.
- Slovo je přijímáno automatem, pokud existuje posloupnost výběrů vedoucí do koncového stavu.
- Implementace:
  - Při simulaci si udržujeme množinu možných stavů.
  - Automaticky převedeme na deterministický automat s 2^n stavy.
  - Často lze nalézt ekvivalentní deterministický automat s méně stavy.

- Automat přijímající slova končící na *“ce”*
- Automat přijímající klíčová slova Pythonu.
- Automat přijímající celá nebo reálná čísla.

Regulární výrazy v Pythonu

- Jazyk přijímaný konečným automatem = regulární jazyk.
- Regulární jazyk lze popsat *regulárním výrazem*.
- Pro regulární výrazy existují knihovny a nástroje.

Regulární výrazy v Pythonu

- Jazyk přijímaný konečným automatem = regulární jazyk.
- Regulární jazyk lze popsat *regulárním výrazem*.
- Pro regulární výrazy existují knihovny a nástroje.

Syntaxe

`.`	jakýkoliv znak
`[`*M*`]`	jakýkoliv znak z množiny *M*, lze `[a-z]`
`[^`*M*`]`	jakýkoliv znak mimo *M*
`*`	libovolný počet opakování předchozího
`+`	jedno a více opakování předchozího
`?`	žádné nebo jedno opakování předchozího
`()`	skupina
`|`	alternativy
`\`	následující znak ztrácí speciální význam

(Další viz. dokumentace)

Příklad — jméno proměnné

```
import re
p=re.compile(r'[a-zA-Z_][a-zA-Z0-9_]*')
print(p.fullmatch("moje_promenna12"))
```

```
<_sre.SRE_Match object; span=(0, 15), match='moje_promenna12'>
```

```
print(p.fullmatch("moje{}_promenna"))
```

```
None
```

```
print(p.fullmatch("12moje{}_promenna"))
```

```
None
```

```
print(p.match("moje{}_promenna"))
```

```
<_sre.SRE_Match object; span=(0, 4), match='moje'>
```

`r''` znamená *raw* řetězec, bez interpretace `\n`, `\t`, ...

Příklad — jméno proměnné (2)

reimplementace `is_variable_name`

```
variable_name_regexp=re.compile(r'[a-zA-Z_][a-zA-Z0-9_]*')

def is_variable_name(s):
    return variable_name_regexp.fullmatch(s) is not None

print(is_variable_name("moje_promenna12"))
```

```
True
```

```
print(is_variable_name("moje{}_promenna"))
```

```
False
```

```
print(is_variable_name("12moje{}_promenna"))
```

```
False
```

Příklad — reálné číslo

```
p=re.compile(r'[+-]?[0-9]+(\.[0-9]*)?([eE][+-]?[0-9]+)?')
print(p.fullmatch("-314"))
```

```
<_sre.SRE_Match object; span=(0, 4), match='-314'>
```

```
print(p.fullmatch("3.14"))
```

```
<_sre.SRE_Match object; span=(0, 4), match='3.14'>
```

```
print(p.fullmatch("2341e-23"))
```

```
<_sre.SRE_Match object; span=(0, 8), match='2341e-23'>
```

```
print(p.fullmatch("-2341e+23"))
```

```
<_sre.SRE_Match object; span=(0, 9), match='-2341e+23'>
```

```
print(p.fullmatch("-2341.e"))
```

```
None
```

```
print(p.fullmatch("278h"))
```

```
None
```

Přeskakování komentářů a regulární výrazy

```
import sys
import re

line_pattern=re.compile(r"([^#]*)(#.+)?\n")

def preskoc_komentare(f):
  # vytiskne obsah souboru 'f' s vynechanymi komentari
  for line in f.readlines(): # čte řádku po řádce
    print(line_pattern.fullmatch(line).group(1))

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f: # otevři textový soubor
    preskoc_komentare(f)  # pokud se povedlo, preskakuj
```

Soubor `preskoc_komentare3.py`

```
Terminal> python3 preskoc_komentare3.py preskoc_komentare.py
```

```
import sys

def preskoc_komentare(f):

  stav=0
  while True:
    c=f.read(1)
    if c=="":
      return
    if stav==0:
      if c=="
        stav=1
      else:
        print(c,end="")
    else:
      if c=="\n":
        stav=0
        print(c,end="")
      else:
        pass

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f:
    preskoc_komentare(f)
```

Přeskakování komentářů a regulární výrazy

Verze, kterou nezmate # v řetězci.

```
# Řešení pomocí regulárních výrazů, které nezmate # mezi " "

import sys
import re

line_pattern=re.compile(r'(([^#"]*("[^"]*")?)*)(#.+)?\n')

def preskoc_komentare(f):
  # vytiskne obsah souboru 'f' s vynechanymi komentari
  for line in f.readlines(): # čte řádku po řádce
    print(line_pattern.fullmatch(line).group(1))

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f: # otevři textový soubor
    preskoc_komentare(f)  # pokud se povedlo, preskakuj
```

```
Terminal> python3 preskoc_komentare4.py preskoc_komentare.py
```

```
import sys

def preskoc_komentare(f):

  stav=0
  while True:
    c=f.read(1)
    if c=="":
      return
    if stav==0:
      if c=="#":
        stav=1
      else:
        print(c,end="")
    else:
      if c=="\n":
        stav=0
        print(c,end="")
      else:
        pass

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f:
    preskoc_komentare(f)
```

```
Terminal> python3 preskoc_komentare4.py preskoc_komentare.py
```

```
import sys

def preskoc_komentare(f):

  stav=0
  while True:
    c=f.read(1)
    if c=="":
      return
    if stav==0:
      if c=="#":
        stav=1
      else:
        print(c,end="")
    else:
      if c=="\n":
        stav=0
        print(c,end="")
      else:
        pass

if __name__=="__main__":
  with open(sys.argv[1],'rt') as f:
    preskoc_komentare(f)
```

Cvičení: doplňte řetězce oddělené ’ ’.

Co konečný automat nedovede

ani regulární výraz

- Nemají neomezenou paměť:
- Příklady *neregulárních jazyků:*
  - Řetězec 0^n1^n.
  - Správně uzávorkované výrazy s neomezenou hloubkou uzávorkování.

Co konečný automat nedovede

ani regulární výraz

- Nemají neomezenou paměť:
- Příklady *neregulárních jazyků:*
  - Řetězec 0^n1^n.
  - Správně uzávorkované výrazy s neomezenou hloubkou uzávorkování.
- Chomského hierachie — jazyky regulární $\subseteq$ bezkontextové $\subseteq$ kontextové $\subseteq$ neomezené.

Konečné automaty a regulární výraze

Konečné automaty

- Jednoduchý výpočetní model
  - Rychlý, dobře teoreticky prozkoumaný.
  - Omezující.
- Složitější implementace, existují nástroje pro její generování.
- Vhodné pro řízení jednoduchých systémů.
- Vhodné pro první krok analýzy textu i kódu v programovacích jazycích.
- Nedeterministické konečné automaty.

Regulární výrazy

- Teoreticky ekvivalentní konečným automatům.
- Snadné a rychlé použití, řada vyzkoušených a optimalizovaných knihoven.
- Některé složitější konstrukce jsou značně nepřehledné.
- Omezené množství výstupních akcí. Nelze použít pro řízení systému.