`Programování` 1 pro matematiky

9. cvičení, 2-12-2021

tags: `Programovani 1 2021`, `čtvrtek`

Obsah:

0. Farní oznamy
1. Opakování: Řetězce
2. Opakování: Permutace a kombinace
3. Slovníky a množiny

Farní oznamy

1. **Materiály k přednáškám** najdete v GitHub repozitáři https://github.com/PKvasnick/Programovani-1. Najdete tam také kód ke cvičením a pdf soubory textů cvičením.
2. **Domácí úkoly**
    - Poslední úkoly jsem zadával přes víkend, takže do příštího víkendu máte čas.
    - Kdo dnes bude pozorně poslouchat, sem-tam se doví něco užitečného.
3. **Opakování** dnes bude stručné, ve skutečnosti si řekneme o pár nových věcech.

Kde se nacházíme

Příště začneme mluvit o třídách v Pythonu.

Opakování: řetězce v Pythonu

Dvojice metod, o kterých jsme nemluvili:

`str.maketrans()` a `str.translate/()`

Příklad namísto výkladu:

```
text = """
Zmráka sa, stmieva sa, k noci sa chýli.
– – – – – – – – – – – – – –
Od hory, od lesa tak plače, kvíli...
Výčitky neznámych duše sa chytia.
...Vyplniť nádeje nebolo sily –
zapadly, zapadly vo shone žitia...

Ohlaky nízko sú, tak letia, letia...!
Žaluje zúfale žaloby márne
ktos' príliš úbohý z šíreho sveta,
že veril, že čakal, že starne, starne...

Zmráka sa, stmieva sa. Shora i zdola
```

```
havrany veslujú do noci spešne...
ktos' príliš úbohý o pomoc volá,
do tvári hádže nám spomienky hriešne...
- - - - - - - - - - - - - - -
Zmráka sa, pôjdeme... Noc je už zpola.
"""

find_chars = "\n"
replace_chars = " "
remove_chars = ",.-'!" # - a ' jsou něco jiného než - a '.

table = text.maketrans(find_chars, replace_chars, remove_chars)
print(text.translate(table).split())

['Zmráka', 'sa', 'stmieva', 'sa', 'k', 'noci', 'sa', 'chýli', 'Od', 'hory',
'od', 'lesa', 'tak', 'plače', 'kvíli', 'Výčitky', 'neznámych', 'duše', 'sa',
'chytia', 'Vyplniť', 'nádeje', 'nebolo', 'sily', 'zapadly', 'zapadly', 'vo',
'shone', 'žitia', 'Ohlaky', 'nízko', 'sú', 'tak', 'letia', 'letia',
'Žaluje', 'zúfale', 'žaloby', 'márne', 'ktos', 'príliš', 'úbohý', 'z',
'šíreho', 'sveta', 'že', 'veril', 'že', 'čakal', 'že', 'starne', 'starne',
'Zmráka', 'sa', 'stmieva', 'sa', 'Shora', 'i', 'zdola', 'havrany',
'veslujú', 'do', 'noci', 'spešne', 'ktos', 'príliš', 'úbohý', 'o', 'pomoc',
'volá', 'do', 'tvári', 'hádže', 'nám', 'spomienky', 'hriešne', 'Zmráka',
'sa', 'pôjdeme', 'Noc', 'je', 'už', 'zpola']
```

Syntaxe:

string.maketrans(*x, y, z*)

Parametr	Popis
x	Povinný. Pokud uvedete jediný parametr, musí to být slovník, který určuje, jak provést záměnu. Pokud uvedete dva nebo tři parametry, musí tady být řetězec, určující znaky, které chcete zaměnit.
y	Volitelný. Řetězec stejné délky jako parametr x. Každý znak v prvním paramtru bude zaměněn odpovídajícíím znakem v tomto parametru.
z	Volitelný. Řetězec určující, které znaky mají být z původního řetězce odstraněny.

Mimochodem, pokud je našim úkolem spočítat nějakou statistiku nad délkami slov, můžeme udělat něco mnohem jednoduššího:

```
>>> text = """
Zmráka sa, stmieva sa, k noci sa chýli.
- - - - - - - - - - - - - - -
Od hory, od lesa tak plače, kvíli...
Výčitky neznámych duše sa chytia.
...Vyplniť nádeje nebolo sily -
zapadly, zapadly vo shone žitia...

Ohlaky nízko sú, tak letia, letia...!
Žaluje zúfale žaloby márne
ktos' príliš úbohý z šíreho sveta,
že veril, že čakal, že starne, starne...
```

```
>>> 
Zmráka sa, stmieva sa. Shora i zdola
havrany veslujú do noci spešne...
ktos' príliš úbohý o pomoc volá,
do tvári hádže nám spomienky hriešne...
- - - - - - - - - - - - - - -
Zmráka sa, pôjdeme... Noc je už zpola.
"""
>>> # Kde se v textu nachází znaky, které nejsou písmeny:
>>> whitespace = [i for i in range(len(text)) if not text[i].isalpha()]
>>> # Hledáme počty znaků mezi dvěma následujícími ne-písmeny:
>>> wordlengths = [whitespace[i] - whitespace[i-1] -1 for i in
range(1,len(whitespace))]
>>> wordlengths
[6, 2, 0, 7, 2, 0, 1, 4, 2, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 4, 0, 2, 4, 3, 5, 0, 5, 0, 0,
0, 7, 9, 4, 2, 6, 0, 0, 0, 0, 7, 6, 6, 4, 0, 0, 7, 0, 7, 2, 5, 5, 0, 0, 0,
0, 6, 5, 2, 0, 3, 5, 0, 5, 0, 0, 0, 0, 6, 6, 6, 5, 4, 0, 6, 5, 1, 6, 5, 0,
2, 5, 0, 2, 5, 0, 2, 6, 0, 6, 0, 0, 0, 0, 6, 2, 0, 7, 2, 0, 5, 1, 5, 7, 7,
2, 4, 6, 0, 0, 0, 4, 0, 6, 5, 1, 5, 4, 0, 2, 5, 5, 3, 9, 7, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 6, 2, 0, 7, 0, 0, 0, 3, 2, 2, 5, 0]
>>> # Ještě musíme odfiltrovat "slova" s nulovou délkou.
>>> wordlengths = [whitespace[i] - whitespace[i-1] -1 for i in
range(1,len(whitespace)) if whitespace[i] - whitespace[i-1] -1 > 0]
>>> wordlengths
[6, 2, 7, 2, 1, 4, 2, 5, 2, 4, 2, 4, 3, 5, 5, 7, 9, 4, 2, 6, 7, 6, 6, 4, 7,
7, 2, 5, 5, 6, 5, 2, 3, 5, 5, 6, 6, 6, 5, 4, 6, 5, 1, 6, 5, 2, 5, 2, 5, 2,
6, 6, 6, 2, 7, 2, 5, 1, 5, 7, 7, 2, 4, 6, 4, 6, 5, 1, 5, 4, 2, 5, 5, 3, 9,
7, 6, 2, 7, 3, 2, 2, 5]
```

n-tice

n-tice je neměnná (immutable) struktura, která obsahuje několik objektů, které logicky patří k sobě, například souřadnice x, y bodu v rovině, den, měsíc a rok v datumu a pod.

```
>>> a = 1
>>> b = 2
>>> t = (a,b) # sbalení
>>> t
(1, 2)
>>> a = 2
>>> t
(1, 2)
>>> t[0]
1
>>> t[1]
2
>>> t[0] = 3
Traceback (most recent call last):
  File "<pyshell#292>", line 1, in <module>
    t[0] = 3
TypeError: 'tuple' object does not support item assignment
>>> x, y = t # rozbalení
```

```
>>> x
1
>>> y
2
```

Funkce `enumerate` a `zip`

Chceme položky i s indexy. Standardní kód je iterovat přes index:

```
>>> mesta = ["Praha", "Brno", "Ostrava"]
>>> for i in range(len(mesta)):
    print(i, mesta[i])


0 Praha
1 Brno
2 Ostrava
>>> for i, mesto in enumerate(mesta):
    print(i, mesto)


0 Praha
1 Brno
2 Ostrava
>>> for u in enumerate(mesta):
    print(u)


(0, 'Praha')
(1, 'Brno')
(2, 'Ostrava')
```

Načítáme města a jejich souřadnice, a pak chceme iterovat přes trojice. Standardní kód je zase iterovat přes index:

```
>>> text = """
Praha -2 0
Brno 0 -1
Ostrava 1 1
"""
>>> mesta = []
>>> x = []
>>> y = []
>>> for radek in text.split("\n"):
    if len(radek) == 0:
        continue
    veci = radek.split()
    mesta.append(veci[0])
    x.append(float(veci[1]))
    y.append(float(veci[2]))


>>> mesta, x, y
(['Praha', 'Brno', 'Ostrava'], [-2.0, 0.0, 1.0], [0.0, -1.0, 1.0])

# Standardní způsob:
```

```
>>> for i in range(len(mesta)):
    print(mesta[i], x[i], y[i])


Praha -2.0 0.0
Brno 0.0 -1.0
Ostrava 1.0 1.0

# S využitím funkce zip:
>>> for mesto, x, y in zip(mesta, x, y):
    print(mesto, x, y)


Praha -2.0 0.0
Brno 0.0 -1.0
Ostrava 1.0 1.0
>>>
```

Permutace a kombinace

Ukazovali jsme si, jak vygenerovat permutace všech prvků seznamu

```
def getPermutations(array):
    if len(array) == 1:
        return [array]
    permutations = []
    for i in range(len(array)):
        # get all perm's of subarray w/o current item
        perms = getPermutations(array[:i] + array[i+1:])
        for p in perms:
            permutations.append([array[i], *p])
    return permutations

print(getPermutations([1,2,3]))
```

a jsme implementovali generátor permutací. Podobně lze spočítat kombinace bez opakování i s opakováním, a také vytvořit jejich generátor. Protože to jsou velice důležité metody, jsou implementovány v modulu `itertools`:

```
>>> import itertools
>>>
>>> numbers = list(range(4))
>>> numbers
[0, 1, 2, 3]
>>> itertools.permutations(numbers)
<itertools.permutations object at 0x00000206B7499C70>
>>> for p in itertools.permutations(numbers):
    print(p)


(0, 1, 2, 3)
(0, 1, 3, 2)
(0, 2, 1, 3)
(0, 2, 3, 1)
(0, 3, 1, 2)
(0, 3, 2, 1)
```

```
(1, 0, 2, 3)
(1, 0, 3, 2)
(1, 2, 0, 3)
(1, 2, 3, 0)
(1, 3, 0, 2)
(1, 3, 2, 0)
(2, 0, 1, 3)
(2, 0, 3, 1)
(2, 1, 0, 3)
(2, 1, 3, 0)
(2, 3, 0, 1)
(2, 3, 1, 0)
(3, 0, 1, 2)
(3, 0, 2, 1)
(3, 1, 0, 2)
(3, 1, 2, 0)
(3, 2, 0, 1)
(3, 2, 1, 0)
>>>
```

Podobně máme v `itertools` funkce `combinations`, `combinations_with_replacement` a `product`:

```
# Kombinace:
>>> for c in itertools.combinations(numbers, 2):
    print(c)

(0, 1)
(0, 2)
(0, 3)
(1, 2)
(1, 3)
(2, 3)

# Kombinace s opakováním:
>>> for c in itertools.combinations_with_replacement(numbers, 3):
    print(c)

(0, 0, 0)
(0, 0, 1)
(0, 0, 2)
(0, 0, 3)
(0, 1, 1)
(0, 1, 2)
(0, 1, 3)
(0, 2, 2)
(0, 2, 3)
(0, 3, 3)
(1, 1, 1)
(1, 1, 2)
(1, 1, 3)
(1, 2, 2)
(1, 2, 3)
(1, 3, 3)
(2, 2, 2)
(2, 2, 3)
(2, 3, 3)
```

```
(3, 3, 3)

# Kartézský součin
>>> for c in itertools.product(numbers, repeat = 2):
    print(c)

(0, 0)
(0, 1)
(0, 2)
(0, 3)
(1, 0)
(1, 1)
(1, 2)
(1, 3)
(2, 0)
(2, 1)
(2, 2)
(2, 3)
(3, 0)
(3, 1)
(3, 2)
(3, 3)
```

Množiny

Množiny jsou vysoce optimalizované kontejnery s rychlým vyhledáváním:

```
>>> zvířata = {"kočka", "pes", "lev", "pes", "lev", "tygr"}
>>> zvířata
{'pes', 'tygr', 'lev', 'kočka'}
>>> "tygr" in zvířata # O(log n)
True
>>> set(["a", "b", "c"])
{'b', 'c', 'a'}
set("abrakadabra")
{'d', 'b', 'a', 'r', 'k'}
>>> set() # prázdná množina
set()
>>> {} # není prázdná množina!
{}
>>> type({})
<class 'dict'>
```

Množiny využívají stromové struktury a algoritmy pro rychlé vyhledávání a modifikaci. Vytváření množin a operace:

```
set("abrakadabra")
{'d', 'b', 'a', 'r', 'k'}
>>> a=set("abrakadabra")
>>> b=set("popokatepetl")
>>> "".join(sorted(a))
'abdkr'
>>> a & b # průnik
{'k', 'a'}
>>> a | b # sjednoceni
{'d', 'b', 'o', 'l', 'p', 'e', 'a', 'r', 't', 'k'}
```

```
>>> a - b # rozdíl
{'d', 'b', 'r'}
>>> a.remove("r")
>>> a
{'d', 'b', 'a', 'k'}
>>> b.add("b")
>>> b
{'o', 'b', 'l', 'p', 'e', 'a', 't', 'k'}
>>> a == b
False
```

Slovníky

```
>>> teploty = { "Praha": 17, "Dill´ı": 42,
"Longyearbyen": -46 }
>>> teploty
{'Praha': 17, 'Dill´ı': 42, 'Longyearbyen': -46}
>>> teploty["Praha"]
17
>>> teploty["Debrecen"]
Traceback (most recent call last):
  File "<pyshell#387>", line 1, in <module>
    teploty["Debrecen"]
KeyError: 'Debrecen'
>>> teploty["Debrecen"] = 28
>>>
>>> del teploty["Debrecen"]
>>> "Debrecen" in teploty
False
>>> teploty["Miskolc"]
Traceback (most recent call last):
  File "<pyshell#394>", line 1, in <module>
    teploty["Miskolc"]
KeyError: 'Miskolc'
>>> teploty.get("Miskolc")
None
>>> teploty.get("Miskolc", 20)
20

# Iterujeme ve slovníku:
>>> for k in teploty.keys():
    print(k)

Praha
Dill´ı
Longyearbyen
>>> for v in teploty.values():
    print(v)

17
42
-46
>>> for k, v in teploty.items():
    print(k, v)

Praha 17
Dill´ı 42
```

```
Longyearbyen -46
>>>
```

Comprehensions pro množiny a slovníky:

```
>>> [i % 7 for i in range(50)]
[0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0]
>>> {i % 7 for i in range(50)}
{0, 1, 2, 3, 4, 5, 6}
>>> {i : i % 7 for i in range(50)}
{0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 0, 8: 1, 9: 2, 10: 3, 11: 4, 12: 5, 13: 6, 14: 0, 15: 1, 16: 2, 17: 3, 18: 4, 19: 5, 20: 6, 21: 0, 22: 1, 23: 2, 24: 3, 25: 4, 26: 5, 27: 6, 28: 0, 29: 1, 30: 2, 31: 3, 32: 4, 33: 5, 34: 6, 35: 0, 36: 1, 37: 2, 38: 3, 39: 4, 40: 5, 41: 6, 42: 0, 43: 1, 44: 2, 45: 3, 46: 4, 47: 5, 48: 6, 49: 0}
>>>
```

`defaultdict` - slovník s defaultní hodnotou *pro počítání*

```
>>> from collections import defaultdict
>>> pocet = defaultdict(int)
>>> pocet['abc']
0
>>> from collections import defaultdict
>>> pocet = defaultdict(int)
>>> pocet["abc"]
0
# počítáme slova
>>> for w in "quick brown fox jumps over lazy dog".split():
    pocet[w] += 1
>>> pocet
defaultdict(<class 'int'>, {'abc': 0, 'quick': 1, 'brown': 1, 'fox': 1, 'jumps': 1, 'over': 1, 'lazy': 1, 'dog': 1})
>>> list(pocet.items())
[('abc', 0), ('quick', 1), ('brown', 1), ('fox', 1), ('jumps', 1), ('over', 1), ('lazy', 1), ('dog', 1)]

# počítáme délky slov
>>> podle_delek = defaultdict(list)
>>> for w in "quick brown fox jumps over lazy dog".split():
    podle_delek[len(w)].append(w)

>>> podle_delek
defaultdict(<class 'list'>, {5: ['quick', 'brown', 'jumps'], 3: ['fox', 'dog'], 4: ['over', 'lazy']})
>>>
```