

Programování 1 pro matematiky

9. cvičení, 24-11-2021

tags: Programování 1 2021, středa

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:

```
1 text = """
2 Zmráka sa, stmieva sa, k noci sa chýli.
3 - - - - -
4 Od hory, od lesa tak plače, kvíli...
5 Výčitky neznámych duše sa chytia.
6 ...vyplniť nádeje nebolo sily -
7 zapadly, zapadly vo shone žitia...
8
9 Ohlaky nízko sú, tak letia, letia...!
10 Žaluje zúfale žaloby márne
11 ktos' príliš úbohý z šíreho sveta,
12 že veril, že čakal, že starne, starne...
13
14 Zmráka sa, stmieva sa. Shora i zdola
15 havrany veslujú do noci spešne...
```

```

16 ktos' príliš úbohý o pomoc volá,
17 do tvári hádže nám spomienky hriešne...
18 - - - - -
19 Zmráka sa, pôjdeme... Noc je už zpoľa.
20 ""
21
22 find_chars = "\n"
23 replace_chars = " "
24 remove_chars = ",.-'!" # - a ' jsou něco jiného než - a '.
25
26 table = text.maketrans(find_chars, replace_chars, remove_chars)
27 print(text.translate(table).split())
28
29 ['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', 'zpoľa']

```

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 odpovídajícím znakem v tomto paramtru.
z	Volitelný. Řetězec určující, které znaky mají být z původního řetězce odstraněny.

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.

```

1 >>> a = 1
2 >>> b = 2
3 >>> t = (a,b) # sbalení
4 >>> t
5 (1, 2)
6 >>> a = 2
7 >>> t
8 (1, 2)
9 >>> t[0]
10 1

```

```

11 >>> t[1]
12 2
13 >>> t[0] = 3
14 Traceback (most recent call last):
15   File "<pyshe11#292>", line 1, in <module>
16     t[0] = 3
17 TypeError: 'tuple' object does not support item assignment
18 >>> x, y = t # rozbalení
19 >>> x
20 1
21 >>> y
22 2

```

Funkce enumerate a zip

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

```

1 >>> mesta = ["Praha", "Brno", "Ostrava"]
2 >>> for i in range(len(mesta)):
3     print(i, mesta[i])
4
5
6 0 Praha
7 1 Brno
8 2 Ostrava
9 >>> for i, mesto in enumerate(mesta):
10     print(i, mesto)
11
12
13 0 Praha
14 1 Brno
15 2 Ostrava
16 >>> for u in enumerate(mesta):
17     print(u)
18
19
20 (0, 'Praha')
21 (1, 'Brno')
22 (2, 'Ostrava')

```

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

```

1 >>> text = """
2 Praha -2 0
3 Brno 0 -1
4 Ostrava 1 1
5 """
6 >>> mesta = []
7 >>> x = []
8 >>> y = []
9 >>> for radek in text.split("\n"):
10     if len(radek) == 0:
11         continue
12     veci = radek.split()
13     mesta.append(veci[0])

```

```

14     x.append(float(veci[1]))
15     y.append(float(veci[2]))
16
17
18 >>> mesta, x, y
19 ([ 'Praha', 'Brno', 'Ostrava'], [-2.0, 0.0, 1.0], [0.0, -1.0, 1.0])
20
21 # Standardní způsob:
22 >>> for i in range(len(mesta)):
23     print(mesta[i], x[i], y[i])
24
25
26 Praha -2.0 0.0
27 Brno 0.0 -1.0
28 Ostrava 1.0 1.0
29
30 # S využitím funkce zip:
31 >>> for mesto, x, y in zip(mesta, x, y):
32     print(mesto, x, y)
33
34
35 Praha -2.0 0.0
36 Brno 0.0 -1.0
37 Ostrava 1.0 1.0
38 >>>

```

Permutace a kombinace

Ukazovali jsme si, jak vygenerovat permutace všech prvků seznamu

```

1 def getPermutations(array):
2     if len(array) == 1:
3         return [array]
4     permutations = []
5     for i in range(len(array)):
6         # get all perm's of subarray w/o current item
7         perms = getPermutations(array[:i] + array[i+1:])
8         for p in perms:
9             permutations.append([array[i], *p])
10    return permutations
11
12 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`:

```

1 >>> import itertools
2 >>>
3 >>> numbers = list(range(4))
4 >>> numbers
5 [0, 1, 2, 3]
6 >>> itertools.permutations(numbers)
7 <itertools.permutations object at 0x00000206B7499C70>
8 >>> for p in itertools.permutations(numbers):
9     print(p)

```

```
10
11
12 (0, 1, 2, 3)
13 (0, 1, 3, 2)
14 (0, 2, 1, 3)
15 (0, 2, 3, 1)
16 (0, 3, 1, 2)
17 (0, 3, 2, 1)
18 (1, 0, 2, 3)
19 (1, 0, 3, 2)
20 (1, 2, 0, 3)
21 (1, 2, 3, 0)
22 (1, 3, 0, 2)
23 (1, 3, 2, 0)
24 (2, 0, 1, 3)
25 (2, 0, 3, 1)
26 (2, 1, 0, 3)
27 (2, 1, 3, 0)
28 (2, 3, 0, 1)
29 (2, 3, 1, 0)
30 (3, 0, 1, 2)
31 (3, 0, 2, 1)
32 (3, 1, 0, 2)
33 (3, 1, 2, 0)
34 (3, 2, 0, 1)
35 (3, 2, 1, 0)
36 >>>
```

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

```
1 # Kombinace:
2 >>> for c in itertools.combinations(numbers, 2):
3     print(c)
4
5 (0, 1)
6 (0, 2)
7 (0, 3)
8 (1, 2)
9 (1, 3)
10 (2, 3)
11
12 # Kombinace s opakováním:
13 >>> for c in itertools.combinations_with_replacement(numbers, 3):
14     print(c)
15
16 (0, 0, 0)
17 (0, 0, 1)
18 (0, 0, 2)
19 (0, 0, 3)
20 (0, 1, 1)
21 (0, 1, 2)
22 (0, 1, 3)
23 (0, 2, 2)
24 (0, 2, 3)
25 (0, 3, 3)
26 (1, 1, 1)
```

```

27 (1, 1, 2)
28 (1, 1, 3)
29 (1, 2, 2)
30 (1, 2, 3)
31 (1, 3, 3)
32 (2, 2, 2)
33 (2, 2, 3)
34 (2, 3, 3)
35 (3, 3, 3)
36
37 # kartézský součin
38 >>> for c in itertools.product(numbers, repeat = 2):
39     print(c)
40
41 (0, 0)
42 (0, 1)
43 (0, 2)
44 (0, 3)
45 (1, 0)
46 (1, 1)
47 (1, 2)
48 (1, 3)
49 (2, 0)
50 (2, 1)
51 (2, 2)
52 (2, 3)
53 (3, 0)
54 (3, 1)
55 (3, 2)
56 (3, 3)

```

Množiny

Množiny jsou vysoce optimalizované kontejnery s rychlým vyhledáváním:

```

1 >>> zvířata = {"kočka", "pes", "lev", "pes", "lev", "tygr"}
2 >>> zvířata
3 {'pes', 'tygr', 'lev', 'kočka'}
4 >>> "tygr" in zvířata # O(log n)
5 True
6 >>> set(["a", "b", "c"])
7 {'b', 'c', 'a'}
8 set("abrakadabra")
9 {'d', 'b', 'a', 'r', 'k'}
10 >>> set() # prázdná množina
11 set()
12 >>> {} # není prázdná množina!
13 {}
14 >>> type({})
15 <class 'dict'>

```

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

```

1 set("abrakadabra")
2 {'d', 'b', 'a', 'r', 'k'}

```

```

3 >>> a=set("abrakadabra")
4 >>> b=set("popokatepetl")
5 >>> "".join(sorted(a))
6 'abdkr'
7 >>> a & b # průnik
8 {'k', 'a'}
9 >>> a | b # sjednocení
10 {'d', 'b', 'o', 'l', 'p', 'e', 'a', 'r', 't', 'k'}
11 >>> a - b # rozdíl
12 {'d', 'b', 'r'}
13 >>> a.remove("r")
14 >>> a
15 {'d', 'b', 'a', 'k'}
16 >>> b.add("b")
17 >>> b
18 {'o', 'b', 'l', 'p', 'e', 'a', 't', 'k'}
19 >>> a == b
20 False

```

Slovníky

```

1 >>> teploty = { "Praha": 17, "Dillí": 42,
2 "Longyearbyen": -46 }
3 >>> teploty
4 {'Praha': 17, 'Dillí': 42, 'Longyearbyen': -46}
5 >>> teploty["Praha"]
6 17
7 >>> teploty["Debreceň"]
8 Traceback (most recent call last):
9   File "<pyshell#387>", line 1, in <module>
10     teploty["Debreceň"]
11 KeyError: 'Debreceň'
12 >>> teploty["Debreceň"] = 28
13 >>>
14 >>> del teploty["Debreceň"]
15 >>> "Debreceň" in teploty
16 False
17 >>> teploty["Miskolc"]
18 Traceback (most recent call last):
19   File "<pyshell#394>", line 1, in <module>
20     teploty["Miskolc"]
21 KeyError: 'Miskolc'
22 >>> teploty.get("Miskolc")
23 None
24 >>> teploty.get("Miskolc", 20)
25 20
26
27 # Iterujeme ve slovníku:
28 >>> for k in teploty.keys():
29     print(k)
30
31 Praha
32 Dillí
33 Longyearbyen
34 >>> for v in teploty.values():
35     print(v)
36

```

```

37 17
38 42
39 -46
40 >>> for k, v in teploty.items():
41     print(k, v)
42
43 Praha 17
44 Dillí 42
45 Longyearbyen -46
46 >>>

```

Comprehensions pro množiny a slovníky:

```

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

```

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

```

1 >>> from collections import defaultdict
2 >>> pocet = defaultdict(int)
3 >>> pocet['abc']
4 0
5 >>> from collections import defaultdict
6 >>> pocet = defaultdict(int)
7 >>> pocet["abc"]
8 0
9 # počítáme slova
10 >>> for w in "quick brown fox jumps over lazy dog".split():
11     pocet[w] += 1
12 >>> pocet
13 defaultdict(<class 'int'>, {'abc': 0, 'quick': 1, 'brown': 1, 'fox': 1,
14 'jumps': 1, 'over': 1, 'lazy': 1, 'dog': 1})
15 >>> list(pocet.items())
16 [('abc', 0), ('quick', 1), ('brown', 1), ('fox', 1), ('jumps', 1), ('over',
17 1), ('lazy', 1), ('dog', 1)]
18 # počítáme délky slov
19 >>> podle_delek = defaultdict(list)
20 >>> for w in "quick brown fox jumps over lazy dog".split():
21     podle_delek[len(w)].append(w)
22 >>> podle_delek
23 defaultdict(<class 'list'>, {5: ['quick', 'brown', 'jumps'], 3: ['fox',
24 'dog'], 4: ['over', 'lazy']})
25 >>>

```


