Analýza dat pomocí vestavěných nástrojů Pythonu

Petr Pošík

Katedra kybernetiky, FEL ČVUT v Praze

OI, B4B33RPH - Řešení problémů a hry, 2016

Credits

David Beazley (http://www.dabeaz.com/): **Builtin Superheros!**

PyData Chicago, 2016

https://www.youtube.com/watch?v=j6VSAsKAj98 (https://www.youtube.com/watch?v=j6VSAsKAj98)

Otevřená data

Z http://otevrenadata.cz (http://otevrenadata.cz):

> Otevřená data jsou informace a čísla bezplatně a volně dostupná na internetu ve strukturované a strojově čitelné podobě a jsou zpřístupněna způsobem, který jejich využití neklade zbytečné technické či jiné překážky.

- Nejzajímavější by byla data ze státní správy:
  - Ministerstva
  - ČOI, Statistický úřad
  - Města a obce
  - ...
- V ČR bohužel stále v plenkách, mnoho zveřejněných dat už předzpracováno a sumarizováno, což do značné míry znemožňuje vlastní analýzu.

City of Chicago data portal

City of Chicago data portal: https://data.cityofchicago.org/ (https://data.cityofchicago.org/)

- Data prakticky o všem:
  - Rozpočet, platy úředníků
  - Kriminalita
  - Opravy děr na silnicích
  - ...

Náš projekt

Data: Food Inspections https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5 (https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5)

- Soubor o velikosti >160 MB s více než 130 tis. řádky.

- Využijeme jen vestavěné nástroje Pythonu.

(Specializované nástroje, např. `pandas`, by tuto analýzu umožňily provést ještě snazším způsobem; my ale chceme demonstrovat sílu vestavěných nástrojů.)

Pracovní adresář

In [1]:

```
import os
os.getcwd()
```

Out[1]:

```
'C:\\P\\0Teaching\\rph\\repos\\rph-lectures\\data-analysis'
```

Import dat

Pomocí modulu `csv` Pythonu načteme data jako **seznam slovníků**.

- Možná nejméně efektivní repezentace dat.
- Ale velmi snadno se s ní pracuje.

In [2]:

```
import csv
with open('Food_Inspections.csv', 'r', encoding='utf-8') as f:
    food = list(csv.DictReader(f))
len(food)
```

Out[2]:

```
136481
```

Načetlo se přes 130 tisíc záznamů. A takto vypadá jeden z nich:

In [3]:

```
food[0]
```

Out[3]:


```
{'AKA Name': 'RED APPLE FOOD & LIQUOR INC',
 'Address': '317 E 51ST ST ',
 'City': 'CHICAGO',
 'DBA Name': 'RED APPLE FOOD & LIQUOR INC',
 'Facility Type': 'Grocery Store',
 'Inspection Date': '11/29/2016',
 'Inspection ID': '1756213',
 'Inspection Type': 'Complaint',
 'Latitude': '41.80195464654846',
 'License #': '10894',
 'Location': '(41.80195464654846, -87.61868295742251)',
 'Longitude': '-87.61868295742251',
 'Results': 'Fail',
 'Risk': 'Risk 2 (Medium)',
 'State': 'IL',
 'Violations': '33. FOOD AND NON-FOOD CONTACT EQUIPMENT UTENSILS CLEAN, FREE
 OF ABRASIVE DETERGENTS - Comments: CLEAN/SANITIZE BOTTOM SHELF OF PREP TABL
E AT MEAT PREP AREA IN REAR. | 41. PREMISES MAINTAINED FREE OF LITTER, UNNEC
ESSARY ARTICLES, CLEANING  EQUIPMENT PROPERLY STORED - Comments: REMOVE CLUT
TER AT BOTTOM SHELF OF REAR PREP TABLE, ORGANIZE AND MAINTAIN AT ALL TIMES.
 | 35. WALLS, CEILINGS, ATTACHED EQUIPMENT CONSTRUCTED PER CODE: GOOD REPAI
R, SURFACES CLEAN AND DUST-LESS CLEANING METHODS - Comments: REPLACE/REPAIR
 WATER DAMAGED CEILING TILES AT REAR AND FRONT DISPLAY FLOOR. | 18. NO EVIDE
NCE OF RODENT OR INSECT OUTER OPENINGS PROTECTED/RODENT PROOFED, A WRITTEN L
OG SHALL BE MAINTAINED AVAILABLE TO THE INSPECTORS - Comments: OBSERVED 30 M
ICE DROPPINGS ON FLOOR ALONG WALLBASE AT REAR BEVERAGE, PAPER GOOD STORAGE A
REA. REMOVE DROPPINGS CLEAN/SANITIZE AFFECTED AREAS. SERIOUS VIOLATION 7-38-
020.',
 'Zip': '60615'}
```


Tento typ dat město publikuje o každé inspekci. Najdete tam název podniku, adresu, typ podniku, datum inspekce, výsledky inspekce, porušení předpisů, atd.

Výsledky inspekcí

Zkusme se do dat trochu zavrtat. Může nás zajímat, jaké jsou všechny možné výsledky inspekcí. Použijme set comprehension. Následující kód projde všechny záznamy o inspekcích a posbírá jejich výsledky do množiny. Využijeme přitom vlastnosti, že množina neuchovává duplicitní hodnoty.

In [4]:

```
{row['Results'] for row in food}
```

Out[4]:


```
{'Business Not Located',
 'Fail',
 'No Entry',
 'Not Ready',
 'Out of Business',
 'Pass',
 'Pass w/ Conditions'}
```

Výsledkem inspekce je tedy nejspíš jedna několika předdefinovaných hodnot.

Inspekce s negativním výsledkem

Zkusíme dále analyzovat ty inspekce, které skončily s negativním výsledkem `Fail`. Profiltrujeme data pomocí list comprehension, tj. vytvoříme seznam jen inspekcí s výsledkem `Fail`.

In [5]:

```
fail = [row for row in food if row['Results'] == 'Fail']
len(fail)
```

Out[5]:

```
26568
```

In [6]:

```
len(fail) / len(food)
```

Out[6]:

```
0.1946644587891355
```

Cca 26 tis. inspekcí z cca 130 tis. bylo negativních, tj. téměř 20 %. OK, so 25 thousand inspections out of 130 thousand failed.

Nejhorší místo v Chicagu, kde se najíst

Když už jsme se vydali tímto směrem, kde je v Chicagu nejhorší místo, kde se najíst?

Spočtěme, kolikrát každý podnik při inspekci selhal. Podnik identifikujeme podle pole "DBA Name" (DBA - Doing Business As). Využijeme `Counter`, který inicializujeme hodnotami generátorovéno výrazu. A podívejme se na prvních 5 položek:

In [7]:

```
from collections import Counter
worst = Counter(row['DBA Name'] for row in fail)
worst.most_common(5)
```

Out[7]:

```
[('SUBWAY', 217),
 ('DUNKIN DONUTS', 143),
 ("MCDONALD'S", 94),
 ('7-ELEVEN', 49),
 ('MCDONALDS', 42)]
```

Abychom byli féroví, důvodem, proč tyto podniky mají nejvíc negativních inspekcí je patrně to, že také mají po Chicagu nejvíce provozoven...

Další věc, která je zajímavá, je **jak se vlastně píše McDonalds?** :-)

To je ale velmi běžný problém při zpracování dat. Nejdříve bychom měli data nějak vyčistit. Pokud se na sloupeček DBA Name podíváte podrobněji, zjistíte, že některé hodnoty používají malá písmena, některá velká, některá obsahují apostrofy, jiná ne ...

Potřebovali bychom na hodnoty aplikovat transformaci, která např. odstraní z názvů apostrofy a převede je na velká písmena:

In [8]:

```
"McDonald's".replace("'", "").upper()
```

Out[8]:

```
'MCDONALDS'
```

Abychom tuto tranformaci mohli aplikovat na každý záznam, použijeme relativně novou funkci Pythonu 3.5, která se může zdát neintuitivní a exotická. Berte to jako příklad, co lze dělat pomocí vestavěných nástrojů.

Následující řádek obsahuje list comprehension. Na každý záznam seznamu `fail` aplikujeme transformaci. Transformace je uvedena ve složených závorkách. Vezme slovník `row` tak, jak je, ale pak jeho pole `'DBA Name'` přepíše novou hodnotou odvozenou od té původní. Právě část `**row, key: value` je nová v Pythonu 3.5: vezme slovník, sloučí ho s novým párem `key: value` a vrátí nový slovník.

In [9]:

```
fail = [ {**row, 'DBA Name': row['DBA Name'].replace("'",'').upper()}
        for row in fail ]
```

A nyní můžeme zopakovat naší analýzu, kde je nejhorší místo k jídlu v Chicagu:

In [10]:

```
worst = Counter(row['DBA Name'] for row in fail)
worst.most_common(5)
```

Out[10]:

```
[('SUBWAY', 240),
 ('MCDONALDS', 181),
 ('DUNKIN DONUTS', 155),
 ('7-ELEVEN', 52),
 ('CHIPOTLE MEXICAN GRILL', 49)]
```

Toto je jen jeden krok čištění dat, nedostatky stále existují, např. `'MC DONALDS'` s mezerou, ale to nyní opravovat nebudeme.

Nejhorší adresa, kde se v Chicagu najíst

Můžeme se raději ptát: kde je nejhorší adresa, kde se můžeme v Chicagu najíst?

In [11]:

```
bad = Counter(row['Address'] for row in fail)
bad.most_common(5)
```

Out[11]:

```
[('11601 W TOUHY AVE ', 213),
 ('324 N LEAVITT ST ', 64),
 ('500 W MADISON ST ', 61),
 ('2300 S THROOP ST ', 39),
 ('100 W RANDOLPH ST ', 33)]
```

Výrazně největší počet negativních inspekcí byl zaznamenán na 11601 West Touhy Avenue. Ale možná, že toto vysoké číslo má původ v dávno minulém období a v současnosti je už vše v pořádku. Možná je to naopak. Možná je to stabilní trend. Pojďme to zjistit.

Podíváme se na počet negativních inspekcí podle jednotlivých let. Sestavíme slovník čítačů, kde klíčem ve slovníku bude rok inspekce.

In [12]:

```
from collections import defaultdict
by_year = defaultdict(Counter)
for row in fail:
    year = row['Inspection Date'][-4:]
    by_year[year][row['Address']] += 1
by_year['2016'].most_common(5)
```

Out[12]:

```
[('11601 W TOUHY AVE ', 48),
 ('324 N LEAVITT ST ', 12),
 ('2300 S THROOP ST ', 12),
 ('500 W MADISON ST ', 7),
 ('600 E GRAND AVE ', 7)]
```

In [13]:

```
by_year['2015'].most_common(5)
```

Out[13]:

```
[('11601 W TOUHY AVE ', 39),
 ('500 W MADISON ST ', 12),
 ('324 N LEAVITT ST ', 9),
 ('307 S KEDZIE AVE ', 9),
 ('12 S MICHIGAN AVE ', 8)]
```

In [14]:

```
by_year['2014'].most_common(5)
```

Out[14]:

```
[('11601 W TOUHY AVE ', 32),
 ('500 W MADISON ST ', 17),
 ('324 N LEAVITT ST ', 15),
 ('113-125 N GREEN ST ', 12),
 ('131 N CLINTON ST ', 10)]
```

Zdá se, že počet negativních kontrol za 1 rok na adrese 11601 Touhy Avenue je 2x-3x větší než na jakékoli jiné adrese a že je to stabilní trend. Ale možná je to tím, že je na této adrese mnoho stravovacích podniků...

Co je na adrese 11601 W TOUHY AVE?

Vraťme se k našemu seznamu (čítači) špatných adres:

In [15]:

```
bad.most_common(5)
```

Out[15]:

```
[('11601 W TOUHY AVE ', 213),
 ('324 N LEAVITT ST ', 64),
 ('500 W MADISON ST ', 61),
 ('2300 S THROOP ST ', 39),
 ('100 W RANDOLPH ST ', 33)]
```

In [16]:

```
address = _[0][0]
address
```

Out[16]:

```
'11601 W TOUHY AVE '
```

V datech o inspekcích najdeme také pole `'Location'`, kde najdeme zeměpisnou šířku a délku příslušející k adrese podniku. Vytvořme množinu všech lokací podniků s negativní výsledkem kontroly na naší adrese:

In [17]:

```
locations = {row['Location'] for row in fail if row['Address'] == address}
```

In [18]:

```
locations
```

Out[18]:

```
{'(42.008536400868735, -87.91442843927047)'}
```

Hmmm, jediná lokace je přiřazena všem těmto kontrolám. Co množina jmen podniků, kde byly negativní kontroly?

In [19]:

```
names = {row['DBA Name'] for row in fail if row['Address'] == address}
```

In [20]:

```
names
```

Out[20]:

{'AMERICAN AIRLINES',
 'AMERICAS DOG',
 'ANDIAMOS OHARE, LLC',
 'ARAMARK AT UNITED AIRLINES',
 'ARGO TEA',
 'ARGO TEA CAFE-OHARE T2',
 'AUNTIE ANNES',
 'AUNTIE ANNES PRETZELS',
 'B JS MARKET',
 'BRITISH AIRWAYS',
 'BURRITO BEACH',
 'CAFFE MERCATO',
 'CHICAGO BLACKHAWKS STANLEYS T2 BAR',
 'CHICAGO NEWS & GIFTS',
 'CHILIS T - 3',
 'CHILIS T-I',
 'CHILIS- G CONCOURSE',
 'CNN',
 'DELTA SKY CLUB',
 'EFIES CANTEEN INC',
 'ELIS CHEESECAKE',
 'FARMERS FRIDGE',
 'FRESH ON THE FLY',
 'FRONTERA TORTAS BY RICK BAYLESS GATE K4 T3',
 'FRONTERA TORTAS BY RICK BAYLESS',
 'GARRETT POPCORN SHOPS',
 'GATEGOURMET',
 'GOLD COAST DOGS',
 'GREEN MARKET',
 'HILTON OHARE',
 'HOST INTERNATIONAL B05',
 'HOST INTERNATIONAL INC',
 'HOST INTERNATIONAL INC, CHILIS T-2',
 'HOST INTERNATIONAL INC-GOOSE ISLAND T3',
 'HOST INTERNATIONAL INC-PRAIRIE TAP',
 'HOST INTERNATIONAL INC.',
 'HOT DOG EXPRESS',
 'HUDSON',
 'HUDSON NEWS',
 'HUDSON NEWS OHARE JOINT VENTURE',
 'ICE BAR',
 'INTELLIGENTSIA',
 'JAMBA JUICE',
 'KOREAN AIR LOUNGE',
 'LA TAPENADE GATE H14',
 'LA TAPENADES GATE H14',
 'LOU MITCHELLS EXPRESS INC',
 'MACARONI GRILL',
 'MCDONALDS',
 'MCDONALDS # 12785',
 'MCDONALDS # 17274',
 'MCDONALDS RESTAURANT',
 'NATURAL BREAK',
 'NUTS ON CLARK',
 'OHARE BAR',
 'OHARE HILTON HOTEL',
 'PARADES A CHICAGO BAR',
 'PUBLICAN TAVERN K1',
 'REGGIOS PIZZA EXPRESS',
 'ROCKY MOUNTAIN CHOCOLATE FACTORY',

```
 'RUSH STREET',
 'SALAD WORKS',
 'SARAHS CANDIES',
 'SAS',
 'SKYBRIDGE RESTAURANT & BAR',
 'STARBUCKS',
 'STARBUCKS HK APEX',
 'STARBUCKS L03',
 'SUBWAY SANDWICH',
 'THE GODDESS & GROCER',
 'THE GREAT AMERICAN BAGEL',
 'TOCCO',
 'TORTAS FRONTERA',
 'TRAVEL TRADERS #3081 @ HILTON OHARE',
 'TUSCANY CAFE',
 'UNITED CLUB',
 'UNITED CLUB ,T-1  CONCOURSE C',
 'UNITED CLUB, TERMINAL 2 CONCOURSE F',
 'UNITED CLUB,TERMINAL 1 CONCOURSE B SOUTH',
 'UNITED FIRST INTERNATIONAL LOUNGE T1,CONCOURSE C',
 'UNITED POLARIS LOUNGE - ORD',
 'WOLFGANG EXPRESS',
 'WOLFGANG PUCK, T-3',
 'ZOOTS'}
```

Na té adrese je ale podniků! Co tedy vlastně je na této adrese? Když adresu zadáte do Google Maps, zjistíte, že patrně patří ...

... **O'Hare International Airport**!

O'Hare

Zaostřeme tedy jen na kontroly na letišti O'Hare:

In [21]:

```
ohare = [row for row in fail if row['Address'].startswith('11601 W TOUHY')]
len(ohare)
```

Out[21]:

```
214
```

Hm, v souhrnu negativních kontrol podle adresy jsme na této adrese viděli jen 213 kontrol. Kde se tedy vzala ta 214.?

Pravděpodobně další nekonzistence v datech. Ověřme to:

In [22]:

```
{ row['Address'] for row in ohare }
```

Out[22]:

```
{'11601 W TOUHY AVE ', '11601 W TOUHY AVE T2 F12'}
```

Ano, správně. Adresa u jedné negativní kontroly identifikuje i specifický terminál.

Takže, kolik podniků na letišti O'Hare mělo nějakou negativní kontrolu?

In [23]:

```
bus_names = { row['DBA Name'] for row in ohare }
len(bus_names)
```

Out[23]:

```
84
```

To skoro vypadá jako všechny podniky na letišti. :-)

Nejhorší místo, kde se najíst na letišti O'Hare

Podívejme se tentokrát na pole `'AKA Name'`, kde bude nejspíš rozlišený každý jednotlivý podnik.

In [24]:

```
c = Counter(row['AKA Name'] for row in ohare)
c.most_common(10)
```

Out[24]:

```
[('MACARONI GRILL (T3-K2)', 6),
 ('United Employee Cafeteria (T1 C LL)', 6),
 ("CHILI'S  TOO (T3-H2)", 5),
 ('Gategourmet (BLDG 741)', 5),
 ('ADMIRALS CLUB/AMERICAN AIRLINES (T3/H&K)', 5),
 ('ARGO TEA  (T3 ROTUNDA)', 4),
 ("CHILI'S (T1-B14)", 4),
 ('WOLFGANG PUCK (T3 K1)', 4),
 ('TOCCO (T5 M-07)', 4),
 ("HILTON O'HARE", 4)]
```

No, vypadá to, že tam není tak úplně jedno nejhorší místo, kde se najíst. Mnoho podniků má podobný počet negativních kontrol a tyto podniky jsou rozprostřeny přes různé terminály.

Seskupení inspekcí

Pokračujme v práci s daty z O'Hare. Zkusme seskupit negativní inspekce podle podniků. Podnik identifikujeme podle čísla licence.

In [25]:

```
inspections = defaultdict(list)
for row in ohare:
    inspections[row['License #']].append(row)
```

Vytvořili jsme slovník, kde klíčem je číslo licence každého podniku a hodnotou je seznam všech příslušných negativních kontrol.

In [26]:

```
inspections.keys()
```

Out[26]:

```
dict_keys(['2487937', '2192963', '34192', '2261733', '1926528', '34212', '20
09092', '2069938', '2232034', '1333242', '34220', '1675026', '2125246', '236
3762', '1878675', '2013208', '34167', '1069379', '2289524', '0', '1069382',
 '1120626', '1879164', '34183', '1042895', '2114331', '1381615', '2016727',
 '1885160', '34142', '15531', '34139', '1224624', '2016732', '1927556', '133
3092', '2277391', '2232035', '2492747', '2289495', '2487849', '2141979', '13
33098', '34235', '1884292', '34205', '34210', '2289527', '1942304', '228429
4', '2009095', '2284027', '2363771', '85188', '2428080', '2192969', '189807
5', '34169', '1333235', '34190', '2447055', '1947515', '2261728', '2289515',
 '2013206', '2363763', '23894', '1879167', '2124574', '37170', '34201', '188
8807', '2487848', '1879166', '2103989', '1909539', '2016729', '2289525', '56
367', '34219', '2192968', '1974743', '56366', '2204037', '1356711', '249275
4', '2363766', '1954648', '2492748', '1884293', '1141505', '29570', '212548
9', '2451545', '34229', '34173', '2289511', '34224', '2184012', '2289531',
 '1909532', '2289520', '1142116', '2299087', '2363760', '2492753', '246399
1', '2017724', '2109577', '2289084', '34199', '2487932', '34203', '1916161',
 '34146', '2021757', '34154', '2146327', '51206', '1718776', '34211', '22773
63', '34234', '2124567', '1916219', '2428079', '64032', '1621425'])
```

In [27]:

```
inspections['34192']
```

```
  'Location': '(42.008536400868735, -87.91442843927047)',
  'Longitude': '-87.91442843927047',
  'Results': 'Fail',
  'Risk': 'Risk 1 (High)',
  'State': 'IL',
  'Violations': '18. NO EVIDENCE OF RODENT OR INSECT OUTER OPENINGS PROTEC
TED/RODENT PROOFED, A WRITTEN LOG SHALL BE MAINTAINED AVAILABLE TO THE INS
PECTORS - Comments: OBSERVED OVER 10 LIVE SMALL FLIES ON WALLS AROUND DISH
MACHINE. INSTRUCTED MANAGER TO REMOVE SMALL FLIES BY DATE OF REINSPECTION,
 ALSO MUST SANITIZE ALL EFFECTED AREAS. SERIOUS VIOLATION 7-38-020. | 29.
 PREVIOUS MINOR VIOLATION(S) CORRECTED 7-42-090 - Comments: THE FOLLOWING
 PREVIOUS MINOR VIOLATIONS FROM REPORT #1424223 DATED ON 04/15/14 REMAINS,
 #33 THE FOLLOWING NOT CLEAN- FOOD/GREASE BUILD UP ON ALL PREP AREA SHELVE
S AND EXTERIOR OF ALL COOKING/COOLING EQUIPMENT, #34 OBBSERVED FOOD/DIRT B
UILD UP ON FLOOR UNDER AND AROUND ALL COOKING AND COOLING EQUIPMENT, MUST
 CLEAN FLOOR THROUGHOUT AND MAINTAIN AND #35 OBSERVED FOOD BUILD UP ON WAL
LS UNDER AND BEHIND DISHMACHINE, MUST CLEAN WALLS IN DISHWASH AND PREP ARE
A. INSTRUCTED MANAGER ALL PREVIOUS MINOR VIOLATIONS MUST BE CORRECTED BY D
ATE OF REINSPECTION. SERIOUS VIOLATION 7-42-090.',
  'Zip': '60666'}
```

Můžou nás zajímat např. data inspekcí v určitém podniku. Můžeme je získat např. takhle:

In [28]:

```
[row['Inspection Date'] for row in inspections['34192']]
```

Out[28]:

```
['04/07/2016', '09/04/2014', '09/20/2011', '01/26/2010']
```

Samozřejmě jsme stejná data mohli získat z původních dat:

In [29]:

```
[row['Inspection Date'] for row in ohare if row['License #'] == '34192']
```

Out[29]:

```
['04/07/2016', '09/04/2014', '09/20/2011', '01/26/2010']
```

Která z uvedených metod je užitečnější záleží na situaci. Pokud plánujete dělat více analýz, kde se vám budou hodit seskupená data, pak je první možnost lepší.

Nejčastější nedostatky

Podívejme se na popis nedostatků ve výsledcích inspekce?

In [30]:

```
ohare[1]
```

Out[30]:


```
{'AKA Name': 'UNITED POLARIS LOUNGE  (T1 C16)',
 'Address': '11601 W TOUHY AVE ',
 'City': 'CHICAGO',
 'DBA Name': 'UNITED POLARIS LOUNGE - ORD',
 'Facility Type': 'Restaurant',
 'Inspection Date': '11/29/2016',
 'Inspection ID': '1976122',
 'Inspection Type': 'License',
 'Latitude': '42.008536400868735',
 'License #': '2492747',
 'Location': '(42.008536400868735, -87.91442843927047)',
 'Longitude': '-87.91442843927047',
 'Results': 'Fail',
 'Risk': 'Risk 1 (High)',
 'State': 'IL',
 'Violations': '12. HAND WASHING FACILITIES: WITH SOAP AND SANITARY HAND DRY
ING DEVICES, CONVENIENT AND ACCESSIBLE TO FOOD PREP AREA - Comments: INSTRUC
TED TO PROVIDE HAND SOAP AND HAND DRYING DEVICE OR PAPER TOWELS AT ALL HAND
 SINK.\nCRITICAL VIOLATION 7-38-030 | 2. FACILITIES TO MAINTAIN PROPER TEMPE
RATURE - Comments: ALL COOLER MUST MAINTAIN COLD HOLDING TEMPERATURE OF 40F
 OR BELOW AND FREEZER AT 0F OR BELOW. | 8. SANITIZING RINSE FOR EQUIPMENT AN
D UTENSILS:  CLEAN, PROPER TEMPERATURE, CONCENTRATION, EXPOSURE TIME - Comme
nts: NOTED INOPERABLE DISH MACHINES AT THIS TIME. A MECHANICAL  DISH WASHING
 MACHINES SHOULD PROVIDE A FINAL SANITIZING RINSE OF EITHER 50 ??? 100 PPM C
HLORINE (FOR CHEMICAL SANITIZING MACHINE) OR 180F FINAL RINSE (FOR HOT WATER
 SANITIZING MACHINE).  MUST PROVIDE  ADEQUATE  SANITIZER  CONCENTRATION  OF
 25 PPM AT 120F, 50 PPM AT 75F, 100 PPM AT 55F.\nCRITICAL VIOLATION 7-38-030
 | 32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND
 MAINTAINED - Comments: INSTRUCTED TO INSTALL SPLASH GUARD OR DIVIDER BETWEE
N EXPOSED HAND SINK AND 3 COMPARTMENT BAR SINK DRAINBOARD BEHIND BAR.',
 'Zip': '60666'}
```


Pole `'Violations'` má jistou vnitřní strukturu. Jednotlivé položky jsou odděleny svislítky (`|`). Každý nedostatek má číslo, (doufejme) standardizovaný název a mohou být připojeny komentáře.

Můžeme se pokusit extrahovat z těchto polí nějaké informace. Podívejme se nejprvve, jak můžeme zpracovat jeden text z pole `'Violations'`:

In [31]:

```
v = ohare[1]['Violations']
v
```

Out[31]:

```
'12. HAND WASHING FACILITIES: WITH SOAP AND SANITARY HAND DRYING DEVICES, CO
NVENIENT AND ACCESSIBLE TO FOOD PREP AREA - Comments: INSTRUCTED TO PROVIDE
 HAND SOAP AND HAND DRYING DEVICE OR PAPER TOWELS AT ALL HAND SINK.\nCRITICA
L VIOLATION 7-38-030 | 2. FACILITIES TO MAINTAIN PROPER TEMPERATURE - Commen
ts: ALL COOLER MUST MAINTAIN COLD HOLDING TEMPERATURE OF 40F OR BELOW AND FR
EEZER AT 0F OR BELOW. | 8. SANITIZING RINSE FOR EQUIPMENT AND UTENSILS:  CLE
AN, PROPER TEMPERATURE, CONCENTRATION, EXPOSURE TIME - Comments: NOTED INOPE
RABLE DISH MACHINES AT THIS TIME. A MECHANICAL  DISH WASHING MACHINES SHOULD
 PROVIDE A FINAL SANITIZING RINSE OF EITHER 50 ??? 100 PPM CHLORINE (FOR CHE
MICAL SANITIZING MACHINE) OR 180F FINAL RINSE (FOR HOT WATER SANITIZING MACH
INE).  MUST PROVIDE  ADEQUATE  SANITIZER  CONCENTRATION  OF 25 PPM AT 120F,
 50 PPM AT 75F, 100 PPM AT 55F.\nCRITICAL VIOLATION 7-38-030 | 32. FOOD AND
 NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND MAINTAINED - C
omments: INSTRUCTED TO INSTALL SPLASH GUARD OR DIVIDER BETWEEN EXPOSED HAND
 SINK AND 3 COMPARTMENT BAR SINK DRAINBOARD BEHIND BAR.'
```

Chceme nejprve rozdělit řetězec podle svislítek:

In [32]:

```
v.split('|')
```

Out[32]:

```
['12. HAND WASHING FACILITIES: WITH SOAP AND SANITARY HAND DRYING DEVICES, C
ONVENIENT AND ACCESSIBLE TO FOOD PREP AREA - Comments: INSTRUCTED TO PROVIDE
 HAND SOAP AND HAND DRYING DEVICE OR PAPER TOWELS AT ALL HAND SINK.\nCRITICA
L VIOLATION 7-38-030 ',
 ' 2. FACILITIES TO MAINTAIN PROPER TEMPERATURE - Comments: ALL COOLER MUST
 MAINTAIN COLD HOLDING TEMPERATURE OF 40F OR BELOW AND FREEZER AT 0F OR BELO
W. ',
 ' 8. SANITIZING RINSE FOR EQUIPMENT AND UTENSILS:  CLEAN, PROPER TEMPERATUR
E, CONCENTRATION, EXPOSURE TIME - Comments: NOTED INOPERABLE DISH MACHINES A
T THIS TIME. A MECHANICAL  DISH WASHING MACHINES SHOULD PROVIDE A FINAL SANI
TIZING RINSE OF EITHER 50 ??? 100 PPM CHLORINE (FOR CHEMICAL SANITIZING MACH
INE) OR 180F FINAL RINSE (FOR HOT WATER SANITIZING MACHINE).  MUST PROVIDE
 ADEQUATE  SANITIZER  CONCENTRATION  OF 25 PPM AT 120F, 50 PPM AT 75F, 100 P
PM AT 55F.\nCRITICAL VIOLATION 7-38-030 ',
 ' 32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND
 MAINTAINED - Comments: INSTRUCTED TO INSTALL SPLASH GUARD OR DIVIDER BETWEE
N EXPOSED HAND SINK AND 3 COMPARTMENT BAR SINK DRAINBOARD BEHIND BAR.']
```

Také bychom měli odstranit bílé znaky a budeme ignorovat (odstraníme) komentáře:

In [33]:

```
[s[:s.find('- Comments')].strip() for s in v.split('|')]
```

Out[33]:

```
['12. HAND WASHING FACILITIES: WITH SOAP AND SANITARY HAND DRYING DEVICES, C
ONVENIENT AND ACCESSIBLE TO FOOD PREP AREA',
 '2. FACILITIES TO MAINTAIN PROPER TEMPERATURE',
 '8. SANITIZING RINSE FOR EQUIPMENT AND UTENSILS:  CLEAN, PROPER TEMPERATUR
E, CONCENTRATION, EXPOSURE TIME',
 '32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND
 MAINTAINED']
```

Pokud chceme, můžeme si na úpravu řetězce udělat funkci:

In [34]:

```
def strip_comments(s):
    return s[:s.find('- Comments')].strip()
[strip_comments(s) for s in v.split('|')]
```

Out[34]:

```
['12. HAND WASHING FACILITIES: WITH SOAP AND SANITARY HAND DRYING DEVICES, C
ONVENIENT AND ACCESSIBLE TO FOOD PREP AREA',
 '2. FACILITIES TO MAINTAIN PROPER TEMPERATURE',
 '8. SANITIZING RINSE FOR EQUIPMENT AND UTENSILS:  CLEAN, PROPER TEMPERATUR
E, CONCENTRATION, EXPOSURE TIME',
 '32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND
 MAINTAINED']
```

Fajn. Nyní můžeme tuto transformaci aplikovat na každé pole `'Violations'` každé inspekce a z výsledků zkonstruovat čítač.

In [35]:

```
c = Counter(
    strip_comments(s)
    for row in ohare
    for s in row['Violations'].split('|')
)
sum(x for x in c.values())
```

Out[35]:

```
999
```

In [36]:

```
c.most_common(5)
```

Out[36]:

```
[('33. FOOD AND NON-FOOD CONTACT EQUIPMENT UTENSILS CLEAN, FREE OF ABRASIVE
 DETERGENTS',
  124),
 ('34. FLOORS: CONSTRUCTED PER CODE, CLEANED, GOOD REPAIR, COVING INSTALLED,
 DUST-LESS CLEANING METHODS USED',
  122),
 ('35. WALLS, CEILINGS, ATTACHED EQUIPMENT CONSTRUCTED PER CODE: GOOD REPAI
R, SURFACES CLEAN AND DUST-LESS CLEANING METHODS',
  98),
 ('18. NO EVIDENCE OF RODENT OR INSECT OUTER OPENINGS PROTECTED/RODENT PROOF
ED, A WRITTEN LOG SHALL BE MAINTAINED AVAILABLE TO THE INSPECTORS',
  94),
 ('32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND
 MAINTAINED',
  77)]
```

Pokud předcházejícímu příkazu není rozumět, můžete postupovat po krocích.

Extrahujme nejprve texty nedostatků:

In [37]:

```
all_violation_texts = [row['Violations'] for row in ohare]
len(all_violation_texts)
```

Out[37]:

```
214
```

Pak můžeme každý text s nedostatky rozdělit na popis jednotlivých nedostatků:

In [38]:

```
all_violations = []
for vtext in all_violation_texts:
    all_violations.extend(vtext.split('|'))
len(all_violations)
```

Out[38]:

```
999
```

Pak můžeme odstranit bílé znaky a komentáře z každého popisu:

In [39]:

```
all_violation_codes = [strip_comments(v) for v in all_violations]
len(all_violation_codes)
```

Out[39]:

```
999
```