Out[35]:
```
Mozilla/5.0                 2594
Mozilla/4.0                  601
GoogleMaps/RochesterNY       121
Opera/9.80                    34
TEST_INTERNET_AGENT           24
GoogleProducer                21
Mozilla/6.0                    5
BlackBerry8520/5.0.0.681       4
dtype: int64
```

In [36]:
```
# 윈도우 사용자와비윈도우 사용자 그룹으로 구분

cframe = frame[frame["a"].notna()].copy()
```

In [37]:
```
cframe["os"] = np.where(cframe["a"].str.contains("Windows"),
                        "Windows", "Not Windows")
cframe["os"].head(5)
```

Out[37]:
```
0        Windows
1    Not Windows
2        Windows
3    Not Windows
4        Windows
Name: os, dtype: object
```

In [38]:
```
by_tz_os = cframe.groupby(["tz", "os"])
```

In [39]:
```
agg_counts = by_tz_os.size().unstack().fillna(0)
agg_counts.head()
```

Out[39]:

os	Not Windows	Windows
tz		
	245.0	276.0
Africa/Cairo	0.0	3.0
Africa/Casablanca	0.0	1.0
Africa/Ceuta	0.0	2.0
Africa/Johannesburg	0.0	1.0

jupyter CH 13 Last Checkpoint: 2023.10.31 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 (ipykernel)

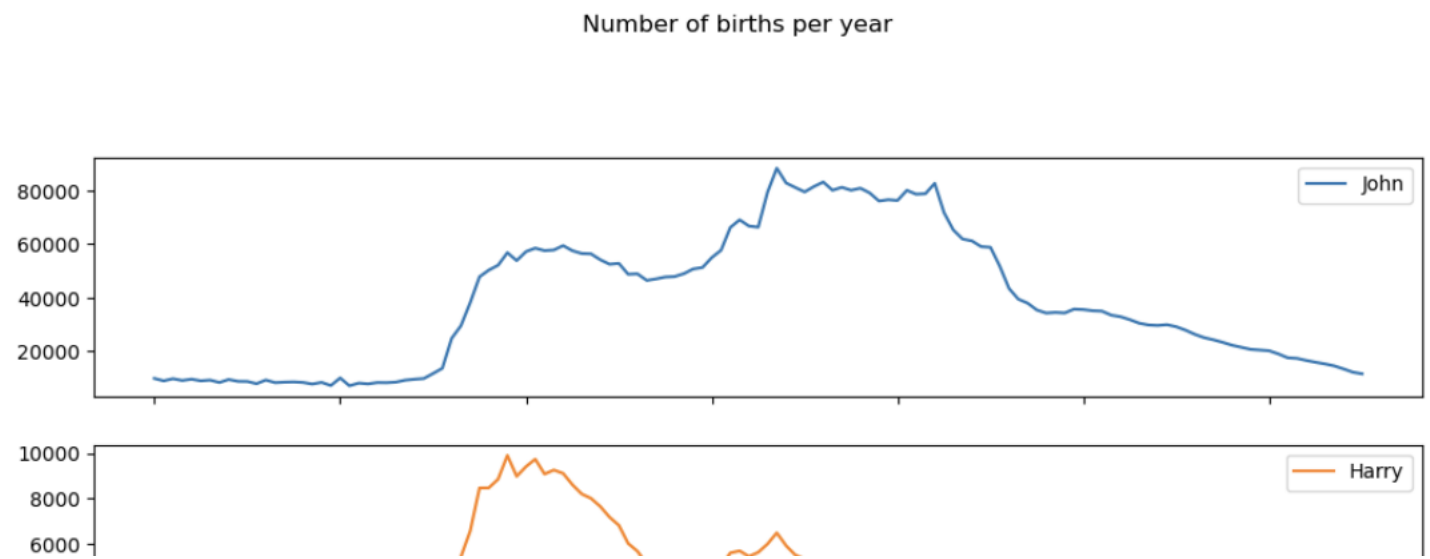
Run Code

In [89]:
```
total_births = top1000.pivot_table("births", index="year",
                                   columns="name",
                                   aggfunc=sum)
```

In [90]:
```
total_births.info()
subset = total_births[["John", "Harry", "Mary", "Marilyn"]]
subset.plot(subplots=True, figsize=(12, 10),
            title="Number of births per year")
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 131 entries, 1880 to 2010
Columns: 6868 entries, Aaden to Zuri
dtypes: float64(6868)
memory usage: 6.9 MB
```

Out[90]:
```
array([<AxesSubplot: xlabel='year'>, <AxesSubplot: xlabel='year'>,
       <AxesSubplot: xlabel='year'>, <AxesSubplot: xlabel='year'>],
      dtype=object)
```



jupyter CH 13 Last Checkpoint: 2023.10.31 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 (ipykernel)

Run Code

In [124]:
```
by_nutrient = ndata.groupby(["nutgroup", "nutrient"])

def get_maximum(x):
    return x.loc[x.value.idxmax()]

max_foods = by_nutrient.apply(get_maximum)[["value", "food"]]

# make the food a little smaller
max_foods["food"] = max_foods["food"].str[:50]
```

In [125]:
```
max_foods.loc["Amino Acids"]["food"]
```

Out[125]:
```
nutrient
Alanine                           Gelatins, dry powder, unsweetened
Arginine                               Seeds, sesame flour, low-fat
Aspartic acid                                     Soy protein isolate
Cystine                Seeds, cottonseed flour, low fat (glandless)
Glutamic acid                                     Soy protein isolate
Glycine                           Gelatins, dry powder, unsweetened
Histidine                  Whale, beluga, meat, dried (Alaska Native)
Hydroxyproline    KENTUCKY FRIED CHICKEN, Fried Chicken, ORIGINAL RE
Isoleucine        Soy protein isolate, PROTEIN TECHNOLOGIES INTERNAT
Leucine           Soy protein isolate, PROTEIN TECHNOLOGIES INTERNAT
Lysine            Seal, bearded (Oogruk), meat, dried (Alaska Native
Methionine                     Fish, cod, Atlantic, dried and salted
Phenylalanine     Soy protein isolate, PROTEIN TECHNOLOGIES INTERNAT
Proline                           Gelatins, dry powder, unsweetened
Serine            Soy protein isolate, PROTEIN TECHNOLOGIES INTERNAT
Threonine         Soy protein isolate, PROTEIN TECHNOLOGIES INTERNAT
Tryptophan          Sea lion, Steller, meat with fat (Alaska Native)
Tyrosine          Soy protein isolate, PROTEIN TECHNOLOGIES INTERNAT
Valine            Soy protein isolate, PROTEIN TECHNOLOGIES INTERNAT
Name: food, dtype: object
```