Python入門(2)

このノートでは：

- モジュール
- 標準ライブラリのモジュール
- csvファイルの読み込み
- 乱数と時間のライブラリ

モジュールとimport文

Pythonコードを複数のファイルで作り、それらを使っていく、モジュールは、Pythonコードをまとめたファイルに過ぎない。モジュールにするために特別なことをする必要はない。すべてのPythonコードがほかのコードからモジュールとして使えるようになっている。

他のモジュールのコードは、import文で参照する。インポートしたモジュールのコードや変数がプログラム内で使えるようになる。

次の例で標準ライブラリのモジュールを紹介する。

In [1]:
```
import sys
# Pythonのバージョンを出力する
print(sys.version)
```

```
3.7.10 (default, Feb 20 2021, 21:17:23)
[GCC 7.5.0]
```

In [2]:
```
from datetime import date

令和３年度昭和の日 = date(2021,4,29)
令和３年度昭和の日
```

Out[2]: datetime.date(2021, 4, 29)

In [3]:
```
令和３年度昭和の日.isoformat()
```

Out[3]: '2021-04-29'

In [4]:
```
今日 = date.today()
今日.isoformat()
```

Out[4]: '2021-04-23'

In [6]:
```
from datetime import timedelta
一日 = timedelta(days=1)
明日 = 今日 + 一日
print("明日",明日.isoformat())
昨日 = 今日 - 一日
print("昨日",昨日.isoformat())
```

```
明日 2021-04-24
昨日 2021-04-22
```

In [7]:
```
import time
今 = time.time() #現地時間
time.ctime(今)
```

Out[7]: 'Fri Apr 23 03:10:10 2021'

In [8]:
```
#colabサーバの現地時間
#fmt = "%A, %B %d, %Y, 現地時間: %I:%M:%S%p"
fmt = "%A, %B %d, %Y, local: %I:%M:%S%p"
time.strftime(fmt)
```

Out[8]: 'Friday, April 23, 2021, local: 03:10:12AM'

In [9]:
```
import math

print("π=",math.pi)
print("e=",math.e)
print("log_10(100) = ",math.log10(100))
print("log_2(1024) = ",math.log2(1024))
```

```
π= 3.141592653589793
e= 2.718281828459045
log_10(100) =  2.0
log_2(1024) =  10.0
```

$$\sqrt{7-4\sqrt{3}} = 2-\sqrt{3}$$

$$2-\sqrt{3}$$

$$= \sqrt{(2-\sqrt{3})^2}$$

$$= \sqrt{4-4\sqrt{3}+3}$$

$$= \sqrt{7-4\sqrt{3}}$$

In [10]:
```
import numpy as np

err = np.sqrt(7-4*np.sqrt(3)) - (2 - np.sqrt(3))
print("誤差 =",err)
```

```
誤差 = 6.661338147750939e-16
```

In [11]:
```
#十進浮動小数点算術
from decimal import Decimal, getcontext
getcontext().prec = 108
a = Decimal(3)
b = Decimal(2)
c = Decimal(7)
err = (c-b*b* a.sqrt()).sqrt() - (b - a.sqrt())
print("誤差 =",err)
```

```
誤差 = 9E-108
```

In [12]:
```
import numpy as np

np.sqrt(7-4*np.sqrt(3)) == (2 - np.sqrt(3))
```

Out[12]:
```
False
```

上記の式は論理的に同じですが、計算機の結果は10^{-16}の誤差が得られるため、機械の判定では同じではない

- 乱数の生成

In [13]:
```
import random

# [0..1]の乱数を生成する
乱数データ = [random.random() for _ in range(3)]
print("(1)3個の乱数：", 乱数データ)
乱数データ = [random.random() for _ in range(3)]
print("(2)3個の乱数：", 乱数データ)
```

```
(1)3個の乱数： [0.4687784639714808, 0.2014090959839051, 0.9595591704511106]
(2)3個の乱数： [0.9546723654979178, 0.24822382986668567, 0.883475817516484]
```

In [14]:
```
# 疑似乱数（決定論的数）
random.seed(4)
乱数データ = [random.random() for _ in range(3)]
print("(1)3個の乱数：", 乱数データ)
random.seed(4)
乱数データ = [random.random() for _ in range(3)]
print("(2)3個の乱数：", 乱数データ)
```

```
(1)3個の乱数： [0.23604808973743452, 0.1031660342307158, 0.396058242610681]
(2)3個の乱数： [0.23604808973743452, 0.1031660342307158, 0.396058242610681]
```

In [15]:
```
# [0..9]の乱数を生成する
乱数データ = [random.randrange(10) for _  in range(10)]
print("10個の乱数：",乱数データ)
```

```
10個の乱数： [2, 1, 1, 0, 6, 8, 4, 0, 3, 8]
```

データを読み込む

- csvモジュール
- matplotlib, timeモジュール

GoogleドライブをColab環境にマウントする

In [16]:
```
from google.colab import drive
drive.mount('/gdrive')
```

```
Mounted at /gdrive
```

あらかじめ用意するフォルダとデータ:

自分のGoogleドライブに「Spring_Practice_2021」というフォルダが存在し、「names.csv」というファイルを格納されている。

このファイルにはアルファベットで書かれた「名」と「姓」の二つの列を含まれている。

目標：ファイルを読み込み、辞書データに変換し、それぞれの「名」の出現頻度を集計してからプロットする。同様にそれぞれの「姓」の出現頻度を集計してからプロットする

In [17]:
```
#names.csvを入れたフォルダに移動する。
%cd '/gdrive/My Drive/Spring_Practice_2021'
```

```
/gdrive/My Drive/Spring_Practice_2021
```

In [18]:
```
%ls
```

```
Basic20210420.ipynb         Basic20210423_v01.ipynb  test.gdraw
Basic20210420_JP.ipynb      Basic20210427.ipynb      test.py
Basic20210420_old.ipynb     Basic20210427_v01.ipynb  v_test.csv
Basic20210420-ver01.ipynb   matplotlib_jp.ipynb
Basic20210423.ipynb         names.csv
```

In [19]:
```
%more names.csv
```

In [20]:
```
# csvファイルを読み込む

import csv

# csvファイルから内容の順序付き辞書で読み込む
def read_dict(f, h):
    input_file = csv.DictReader(open(f), fieldnames=h)
    return input_file

def od_to_d(od): #OrderedDict型をdict型に変換
    return dict(od)

if __name__ == "__main__":
    csv_f = 'names.csv'
    headers = ['名', '姓']
    r_dict = read_dict(csv_f, headers)
    dict_ls = []
    # https://note.nkmk.me/python-collections-ordereddict/ を参照する
    print('¥ncsvを順序付き辞書にする例:')
    for i, row in enumerate(r_dict):
        r = od_to_d(row)
        dict_ls.append(r)
        if i < 3:
            print(row)
    print('¥n辞書オブジェクトのリストの例:')
    for i, row in enumerate(dict_ls):
        if i < 3:
            print(row)
```

```
csvを順序付き辞書にする例:
OrderedDict([('名', 'Adam'), ('姓', 'Baum')])
OrderedDict([('名', 'Adam'), ('姓', 'Zapel')])
OrderedDict([('名', 'Al'), ('姓', 'Bino')])

辞書オブジェクトのリストの例:
{'名': 'Adam', '姓': 'Baum'}
{'名': 'Adam', '姓': 'Zapel'}
{'名': 'Al', '姓': 'Bino'}
```

In [21]:
```
type(dict_ls)
```

Out[21]:
```
list
```

In [22]:
```
len(dict_ls)
```

Out[22]:
```
166
```

In [23]:
```
dict_ls[-3:]
```

Out[23]:
```
[{'名': 'Will', '姓': 'Power'},
 {'名': 'Willie', '姓': 'Waite'},
 {'名': 'Willie', '姓': 'Makeit'}]
```

In [24]:
```
firstname_lst = [dict_ls[n]['名'] for n in range(len(dict_ls))]
firstname_lst[:3]
```

Out[24]:
```
['Adam', 'Adam', 'Al']
```

In [25]:
```
surname_lst = [dict_ls[n]['姓'] for n in range(len(dict_ls))]
surname_lst[:3]
```

Out[25]:
```
['Baum', 'Zapel', 'Bino']
```

In [26]:
```
def count_lst(lst):
  n_counts = {}
  for n in lst:
    if n in n_counts:
      n_counts[n] += 1
    else:
      n_counts[n] = 1
  return n_counts

name_counts = count_lst(firstname_lst)
surname_counts = count_lst(surname_lst)
```

In [27]:
```
top30 = sorted(name_counts.items(), key=lambda x: x[1], reverse=True)
```

```
top30=top30[:30]
```

In [28]:
```
d_top30=dict(top30)
```

- 別のノートブックで説明している

  ######colabで日本語フォントを使う

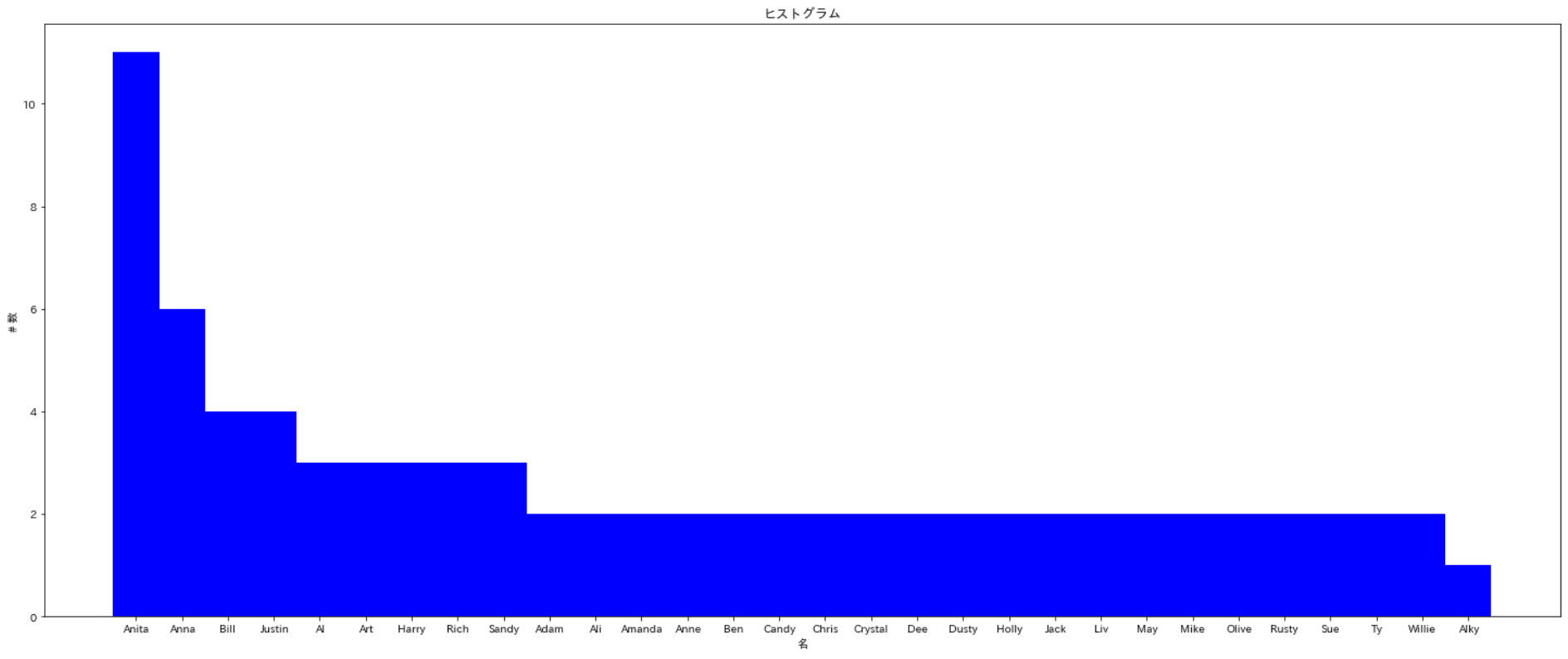
!cd /usr/local/lib

!pip install japanize-matplotlib #サーバ仮想環境にインストール

In [30]:
```
!cd /usr/local/lib
!pip install japanize-matplotlib
```

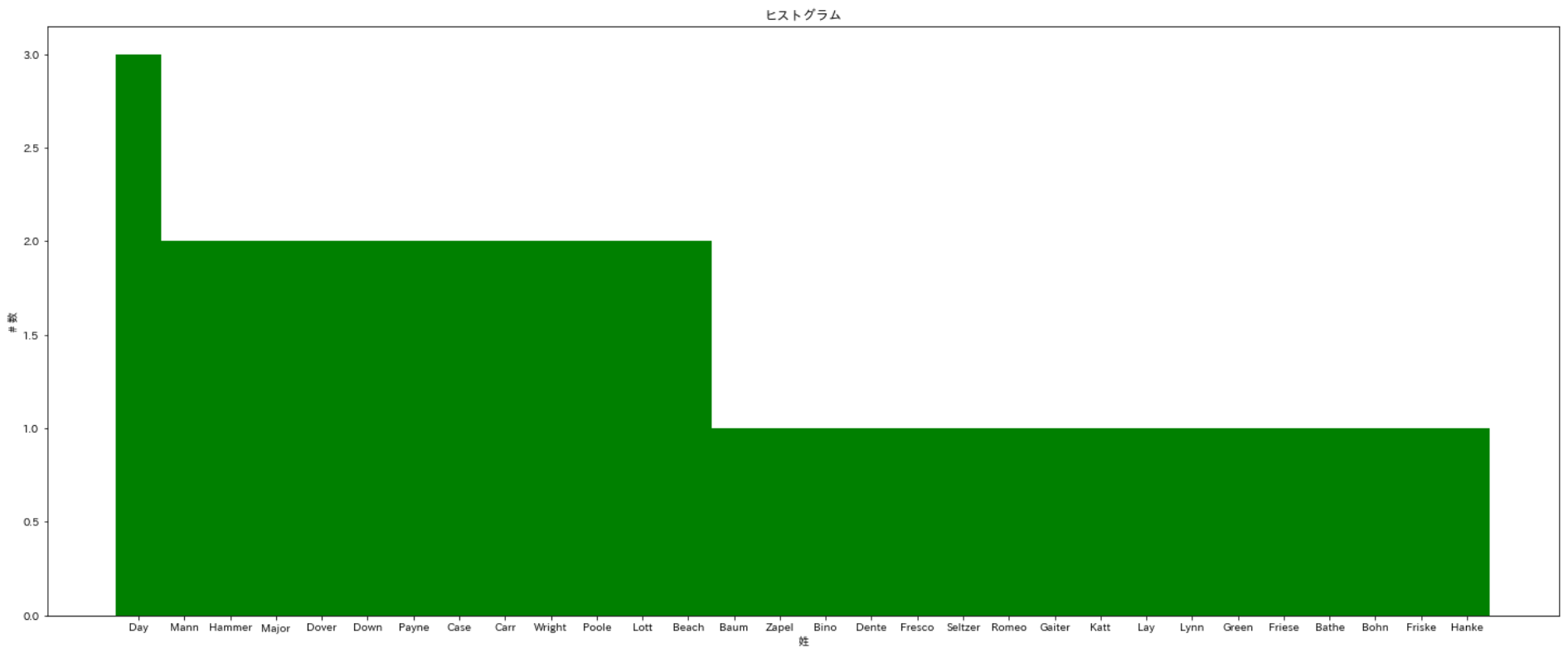
```
Collecting japanize-matplotlib
  Downloading https://files.pythonhosted.org/packages/aa/85/08a4b7fe8987582d99d9bb7ad0ff1ec75439359a7f9690a0dbf2dbf98b15/japanize-matplotlib-1.1.3.tar.gz (4.1MB)
     |████████████████████████████████| 4.1MB 6.6MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from japanize-matplotlib) (3.2.2)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (1.19.5)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (2.4.7)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib->japanize-matplotlib) (1.15.0)
Building wheels for collected packages: japanize-matplotlib
  Building wheel for japanize-matplotlib (setup.py) ... done
  Created wheel for japanize-matplotlib: filename=japanize_matplotlib-1.1.3-cp37-none-any.whl size=4120276 sha256=e280390ab78a7e54f480d989b08b460aac93020dafaed3748361c487bf996bcd
  Stored in directory: /root/.cache/pip/wheels/b7/d9/a2/f907d50b32a2d2008ce5d691d30fb6569c2c93eefcfde55202
Successfully built japanize-matplotlib
Installing collected packages: japanize-matplotlib
Successfully installed japanize-matplotlib-1.1.3
```

In [31]:
```
import japanize_matplotlib
import matplotlib.pyplot as plt

width = 1.0     # 棒グラフをヒストグラムのように見せる
plt.figure(figsize=(25,10))
ax = plt.axes()
ax.set_xticklabels(d_top30.keys())

plt.bar(list(d_top30.keys()), list(d_top30.values()), width, color='blue')
plt.xlabel("名")
plt.ylabel("# 数")
plt.title("ヒストグラム");
plt.show()
```



In [32]:
```
surname_top30 = sorted(surname_counts.items(), key=lambda x: x[1], reverse=True)
s_top30=surname_top30[:30]
d_s_top30=dict(s_top30)
```

In [33]:
```
width = 1.0     # 棒グラフをヒストグラムのように見せる
plt.figure(figsize=(25,10))
ax = plt.axes()
ax.set_xticklabels(d_s_top30.keys())

plt.bar(list(d_s_top30.keys()), list(d_s_top30.values()), width, color='green')
plt.xlabel("姓")
plt.ylabel("# 数")
plt.title("ヒストグラム");
plt.show()
```



- 氏名の姓：Dayはどのような「名」を持つ。以下に操作を行う。
- 氏名の名：Anitaはどのような「姓」を持つ。以下に操作を行う。

In [39]:
```
for n in range(len(dict_ls)):
  if dict_ls[n]['姓']=='Day':
    print("名:",dict_ls[n]['名'],' 姓：',dict_ls[n]['姓'])
```

```
名: Holly  姓:  Day
名: May  姓:  Day
名: Sonny  姓:  Day
```

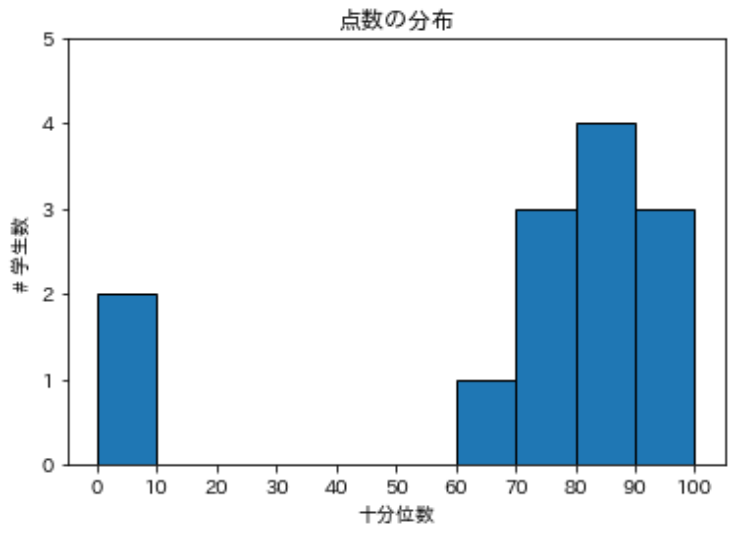
In [40]:
```
for n in range(len(dict_ls)):
  if dict_ls[n]['名']=='Anita':
    print("名:",dict_ls[n]['名'],' 姓：',dict_ls[n]['姓'])
```

```
名: Anita  姓:  Bathe
名: Anita  姓:  Bohn
名: Anita  姓:  Friske
名: Anita  姓:  Hanke
名: Anita  姓:  Goodman
名: Anita  姓:  Job
名: Anita  姓:  Knapp
名: Anita  姓:  Little
名: Anita  姓:  Mann
名: Anita  姓:  Plummer
名: Anita  姓:  Schhauer
```

- matplotlibのプロット

In [35]:
```
import japanize_matplotlib
import matplotlib.pyplot as plt
from collections import Counter

点数 = [83, 95, 91, 87, 70, 0, 85, 82, 100, 67, 73, 77, 0]
十分位数 = Counter(min(_ // 10 * 10, 90) for _ in 点数) #ステップ関数
print("点数の十分位数は",十分位数)
# anaconda環境ならば、日本語のフォントを表示できる。
# コメントした命令が実行できる
# plt.rcParams['font.family'] = 'IPAPGothic'
plt.bar([x + 5 for x in 十分位数.keys()],十分位数.values(),10,edgecolor=(0, 0, 0))
plt.axis([-5, 105, 0, 5])
plt.xticks([10 * i for i in range(11)])    # x軸ラベルに[0, 10, ..., 100]
plt.xlabel("十分位数")
plt.ylabel("# 学生数")
plt.title("点数の分布");
```

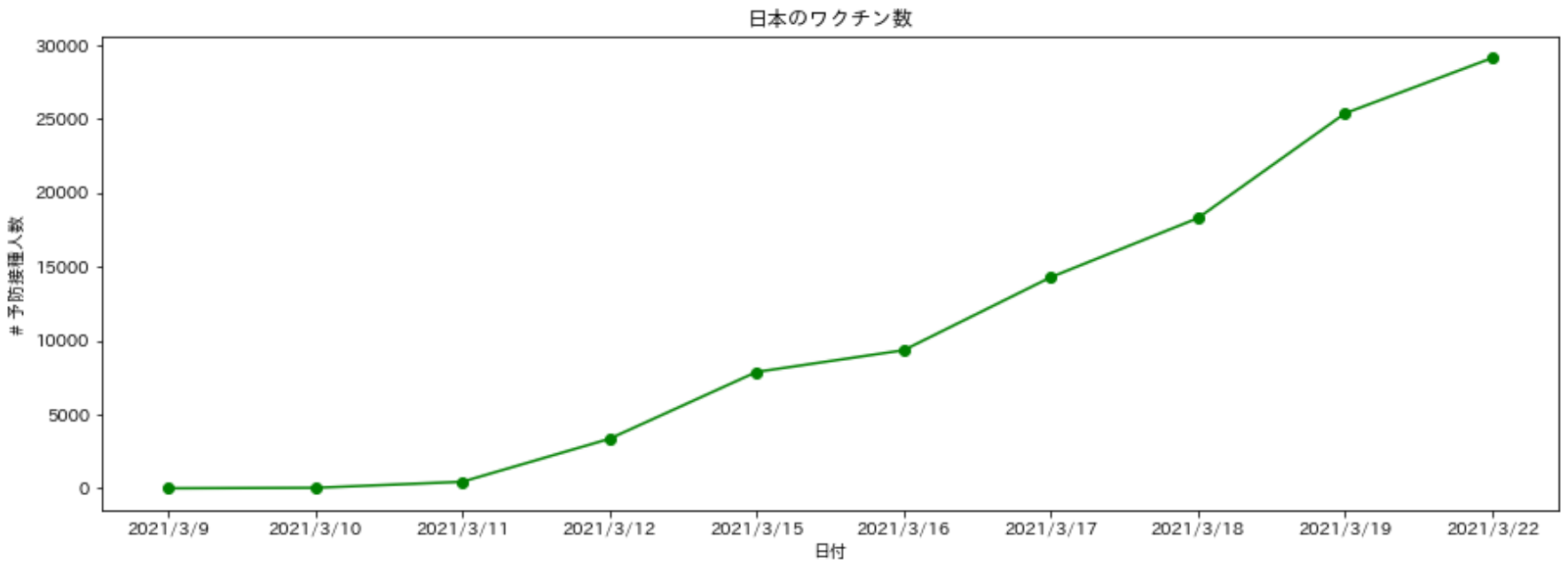
```
点数の十分位数は Counter({80: 4, 90: 3, 70: 3, 0: 2, 60: 1})
```



In [37]:
```
日付 = ['2021/3/9','2021/3/10','2021/3/11','2021/3/12',
        '2021/3/15','2021/3/16','2021/3/17','2021/3/18',
```

```
        '2021/3/19','2021/3/22']
ワクチン数 = [0,35,443,3348,7877,9347,14289,18289,25381,29129]
print("日本のワクチン数")
import matplotlib.pyplot as plt
plt.figure(figsize=(15,5))

#color=green, marker=円, linestyle
plt.plot(日付,ワクチン数,c='g',marker='o',ls='solid');
plt.xlabel("日付")
plt.ylabel("# 予防接種人数")
plt.title("日本のワクチン数");
```

日本のワクチン数



In [ ]: