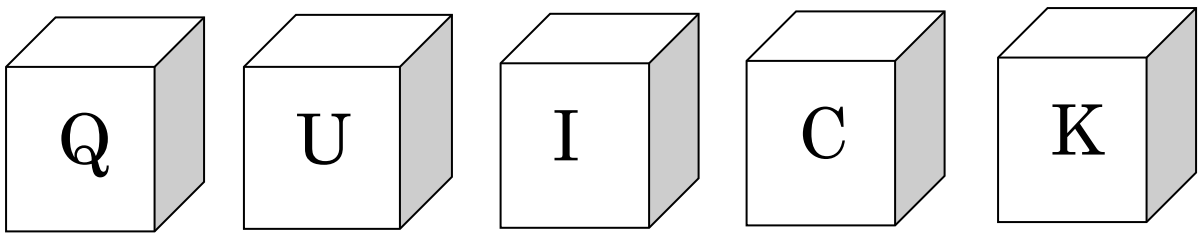
Quick Check

- Grammer＆Usage



>> Python で学び直す数学【集合・確率編】～
　例題を解きながら**集合とリストの使い方を理解しよう**

最近、ニュースなどで「機械学習」や「AI」といったフレーズを耳にすることも増えてきたのではないでしょうか。数学的な方法をベースにしながら、モノを考えるといっても、実際にどう使えばよいだろうとピンと来ない方のほうが多いのではないでしょうか。コンピュータの世界で数学がどのように使えるのか、ということを Python の基本的な書き方を確認しながら考えていきたいと思います。

「数学的な問題を Python で簡単なスクリプトを作って動作を確認する」ということを複数回に分けて行っていきたいと思います。Python に慣れるという点でも手を動かして考える機会にして頂ければ幸いです。

今回は、Python で学び直す数学【集合・確率編】の確認をしていきたいと思います。

○ 集合・確率について

数学の授業で、「確率を求めるときに、図や一覧表を描いたり、ベン図や集合の要素数を利用して、場合の数を求め、全体でどれだけの数があるか正しく把握してください」と習った方も多いと思います。今一度、集合の基礎から学んでいきましょう。

集合：はっきりと区別できて、同じ性質を持ったデータの集まり
確率：ある試行を行ったとき、その結果として起こりうるすべての事象のうち、特定の事象になる割合

まずは、集合の特徴についてみていきます。

・集合の定義

数学の世界で「1 から 10 までの自然数」の集合は、

A = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

のように {} を使って表記します。

「A」は集合に付けた名前で、 {} に含まれる個々の値を要素といいます。
１つの集合に同じ要素が含まれることはなく、また、集合の要素には順番という概念がありません。

下のように「1 から 10 までの自然数」の集合を考えた場合、
B = {2 ,5, 6, 7, 1, 3, 4, 9, 10, 8}

集合 A と集合 B は同じものとして扱われます。

< Type Python >

Python には set という集合を扱うデータ型があります

```
>>> A = {1,2,3,4,5,6,7,8,9,10}
```

のようにすると、集合 A を定義できます。

集合 A の要素の順番を変えて、集合 B を定義して集合Aと比べてみましょう。

```
>>> B = {3, 1, 4, 5, 9, 6, 2, 7, 8, 10}
>>> A == B
True
```

もう一つ、集合の要素には重複した値が含まれないことも確認しておきましょう。

```
>>> A
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
>>> A.add(10)
>>> A
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
>>> len(A)
10
```

len() 関数を利用すると、集合の要素数を確認できることも覚えておきましょう。

○全体集合、部分集合

集合 A のすべての要素が集合 U に含まれるとき、集合 A を集合 U の「部分集合」といい、

U⊃A

のように表します

Python では「＜＝」を使って部分集合かどうかを確認することができます

```
>>> U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}   ●→  集合 U を定義
>>> A = {2, 4, 6, 8, 10}                  ●→  集合 A を定義
>>> A <= U                 ●→  集合 A が U に含まれるか確認
True
```

部分集合が属する集合 U のことを全体集合といいます。

○集合演算（和集合・積集合・差集合）

・和集合

2 つの集合の全要素を合わせた集合を和集合といい、
　　A∪B
のように表します。Python では、｜演算子を使って和集合を取得できます。

```
>>> A = {2, 4, 6, 8, 10}          ●→　集合 A を定義
>>> B = {3, 6, 9}                 ●→　集合 B を定義
>>> A ｜ B                 ●→　集合 A と B の和集合
{2, 3, 4, 6, 8, 9, 10}
```

・積集合

2 つの集合に共通する要素の集合を積集合といい、
　　A∩B
のように表します

Python では、「&演算子」を使って積集合を取得できます

```
>>> A = {2, 4, 6, 8, 10}          ●→　集合 A を定義
>>> B = {3, 6, 9}                 ●→　集合 B を定義
>>> A & B                  ●→　集合 A と B の積集合
{6}
```

・差集合

一方の集合から、もう一方の集合を引いたものを差集合といいます。
どちらの集合から引くかによって結果が変わるので注意してください。

Python では、算術演算子の「－演算子」を使って差集合を取得できます。

```
>>> A = {2, 4, 6, 8, 10}          ●→　集合 A を定義
>>> B = {3, 6, 9}                 ●→　集合 B を定義
>>> A － B                 ●→　集合 A から集合 B の要素を除く
{2, 4, 8, 10}
>>> B － A                 ●→　集合 B から集合 A の要素を除く
{3, 9}
```

⇒ ここで、添付資料「集合の要素数」シートにある演習問題を解いてみましょう

■集合の要素数

問題

1から10までの自然数の中で
「2の倍数」または「3の倍数」はいくつあるでしょうか？
下線に入る式を記入してください。

```
>>> A = {2, 4, 6, 8, 10}      ●→ 集合Aを定義
>>> B = {3, 6, 9}             ●→ 集合Bを定義
>>> _____                     ●→ 和集合（AまたはB）の要素数
7
```

類似するケースとして、Python での辞書, リストの使い方について確認していきましょう。

○ 辞書の使い方 － キーと値のペアでデータを管理

辞書（dictionary）とは、「キー：値」をカンマで区切って並べ、全体を「{}」で囲み、1 組のデータをキーとそれに対応するペアで管理するデータ型です。

辞書オブジェクトを生成

```
>>> animal = {'dog':'犬', 'cat':'猫', 'bird':'鳥'}
[Enter]　…辞書を作成

>>> animal
{'dog':'犬', 'cat':'猫', 'bird':'鳥'}
```

キーと値のペアを変更・追加

```
>>> animal['cat'] = 'ネコ'
[Enter]　…「'cat'」の値を変更

>>> animal
{'dog':'犬', 'cat':'ネコ', 'bird':'鳥'}

>>> animal['mouse'] = 'ネズミ'
{'dog':'犬', 'cat':'ネコ', 'bird':'鳥', 'mouse':'ネズミ'}
```

・キー／値の一覧を取得する

key()　… 辞書内の「すべてのキー」を取得
values() … 辞書内の「すべての値」を取得
items() … 辞書内の「キーと値のペアの一覧」を取得

```
>>> blood_type = {‘a’:’A 型’, ’b’:’B 型’, ’ab’:’AB 型’, ’o’:’O 型’}

>>> blood_type.keys()　[Enter]
dict_keys([‘a’, ‘b’, ‘ab’, ‘o’])

>>> blood_type.values()　[Enter]
dict_values([‘A 型’, ‘B 型’, ‘AB 型’, ‘O 型’])

>>> blood_type.items()　[Enter]
dict_items([('a', 'A 型'), ('b', 'B 型'), ('ab', 'AB 型'), ('o', 'O 型')])
```

keys()、values()、items()メソッドの戻り値は、for 文で順に取得できます。

```
dic = {“春”:”Spring”, “夏”:”Summer”, “秋”:”Autumn”, “冬”:”Winter”}

for jap, eng in dic.items():
        print(“{}:{}”.format(jap, eng))
```

参考：format メソッドは変数の値を埋め込みたい時などに使われます。

例えば　‘str1{}:{}str2’.format(var1, var2)　では、
1 番目の{} にメソッド引数の var1 が、2 番目の{}にメソッド引数の var2 が埋め込まれます。

⇒ ここで、添付資料「辞書の使い方」シートにある演習問題を解いてみましょう

問題：
空欄を埋めて、全ての要素を下のように表示するプログラムを完成させてください。

```
Javaの利用者は 12人
C#の利用者は　8人
VBの利用者は　　5人
JavaScriptの利用者は 15人

lang = { “Java” :12, “C#” :8, “VB” :5, “JavaScript” :15}

 1  l, n in  2  :
  print( “{}の利用者は{}人” .format(l, n))
```

○場合の数

　次は、図や一覧表を描いたり、ベン図や集合の要素数を利用したりして、場合の数の例題を解いていきましょう。

順列：n 個のものから r 個（r ≤ n）を選んで並べる並べ方　nPr は

　nPr　= n (n-1) (n-2) … (n-r+1) =　n! / (n-r)!　(通り)

たとえば、1～9 の 9 種類の数字を使ってできる 2 桁の数字は　　9×8 = 72 (通り)
Python では、以下のように求めることができます。

```
>>>import itertools
>>> num = {1, 2, 3, 4, 5, 6, 7, 8, 9}          ●→　データを定義
>>> A = set(itertools.permutations(num, 2))
●→　num の中から 3 個を選ぶ順列で集合を生成

>>> len(A)                                    ●→　A の要素数を調べる
72
>>> for a in A:                               ●→　A の全要素にアクセスするループ
        print(a)                              ●→　要素を画面に出力
```

⇒ ここで、添付資料「順列」シートにある演習問題を解いてみましょう

■順列計算

問題
1～5の数字を使ってできる3桁の数字の個数を求め、
すべての要素を画面に出力してみましょう。
下線に入る式を記入してください。

```
>>>import itertools
>>> num = {1, 2, 3, 4, 5}      ●→　データを定義
>>> A =  ____1____ :解答入力
●→　numの中から3個を選ぶ順列で集合を生成


>>> ____2____ :解答入力      ●→　Aの要素数を調べる
60
>>> for a in A:              ●→　Aの全要素にアクセスするループ
        print(a)              ●→　要素を画面に出力
```

階乗：n の階乗は、n! と記載します。1～n までの整数の積のことをいいます。

Python で階乗を計算するときは、math モジュールの factorial() 関数を使いましょう。

```
>>>import math
>>> math.factorial(5)          ●→　5 の階乗を求める
120                            ●→ 表示された結果
```

⇒ ここで、添付資料「階乗」シートにある演習問題を解いてみましょう

■階乗計算

問題
「1」「2」「3」「4」「5」「6」「7」「8」「9」の9つの数字を使ってできる
9桁の数字はいくつありますか？
ただし、同じ数字は使えないものとします。

Pythonで階乗を計算するときは、mathモジュールのfactorial() 関数を使いましょう。

```
>>> import math
>>> _____          ●→　9の階乗を求める
                   ●→ 表示される結果
```

組み合わせ：n 個のものから r 個を選び出す組み合わせ nCr は

nCr　= nPr / r! = n! / r! (n-r)!　（通り）

たとえば、1～5 の 5 種類の数字から 2 つの数字を選ぶ組み合わせは　5×4 / 2×1 = 10（通り）

組み合わせが何通りかを調べるときは、itertools モジュールの combinations() 関数を使います。
1～5 のうち、2 個の数字を選んだ場合は以下のようになります。

```
>>> num = {1, 2, 3, 4, 5}          ●→ データを定義
>>> A = set(itertools.combinations(num,2))
●→　num の中から 2 個を選ぶ組み合わせ
```

⇒ ここで、添付資料「組み合わせ」シートにある演習問題を解いてみましょう

■【演習】組み合わせ

問題
「1」「2」「3」「4」「5」「6」「7」「8」「9」の中から4つの数字を選んでください。選び方はいくつありますか？
数字の並べ方は考えずに、9つの中から4つを選ぶ方法は何通りあるかを調べてみましょう
下線に入る式を記入してください。

```
>>> num = {1, 2, 3, 4, 5, 6, 7, 8, 9}        ●→ データを定義
>>> A = _____1_____ :解答入力
●→  numの中から4個を選ぶ組み合わせ
>>> ____2____ ：解答入力
126
>>> for a in A:                    ●→ 表示された結果
    print(a)
```

■【演習】確率

問題
サイコロを振ったら、最終的に1の目が出る確率は 1/6 になるー。本当かどうか確かめてみましょう。
10000回サイコロを振ってみて、1の目が出る回数を確認し、確率を求めてみましょう。

以下のコードを実行して、確率がいくつになるか確認してください。

```
import random
# サイコロを振る
cnt = 0          # 1 が出た回数
for i in range(10000):
    dice = random.randint(1,6)
    if dice == 1:
        cnt += 1

# 確率を求める
p = cnt / 10000
print(p)
```

＝＝

以前学校で学んできた内容をもとに Python でスクリプトを実行しながら確認できるのは面白いなと感じる方もいらっしゃるかもしれません。自分にできる範囲のものから少しずつ Python にも挑戦してみようかなと思っていただければ幸いです。

以上となります。

参考文献：

・大津真『基礎 Python: 入門から実践へステップアップ』インプレス（2016 年）
・谷尻かおり『文系プログラマーのための Python で学び直す高校数学』日経 BP 社（2021 年）