集合 / 順列 / 組合せ

情報科　飯島　涼

目次

- 集合
  - 何故集合を学ぶのか（論理と集合 / データと集合）
  - Pythonにおける集合の扱いについて

- 順列 / 組合せ
  - なぜ順列・組み合わせを学ぶのか
    - ゲーム（Numeron）で体感する，計算するデータ量の削減について
  - Pythonにおける順列・組み合わせについて

参考文献

- 教科書（いつもの）
- 結城浩　プログラマの数学 第2版 SB Creative 2018
- 谷尻かおり　文系プログラマーのためのPythonで学びなおす高校数学，日経BP，2019

集合をなぜ学ぶのか？

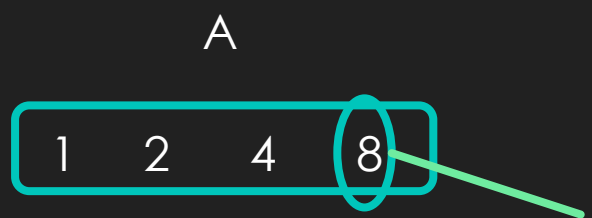
- 今どの範囲のものごとを対象にして議論をしているのかを明らかにするため（一般的，数学）
- 2つの集合の要素の間に，何かしらのルール・対応関係を設定して議論を進めることがある（ゲーム，数学，プログラミング）
- 巨大なデータを処理する（プログラミング）
  - データベース処理
  - ビッグデータ
- 創造的なアイディアを考えるときのブレインストーミング（その他）
  - 意外な二つの要素を足し合わせる
  - 何かから特定の要素を取り除いて新しい価値を作り出す
  - 議論がconflictしたときの折衷案を出す

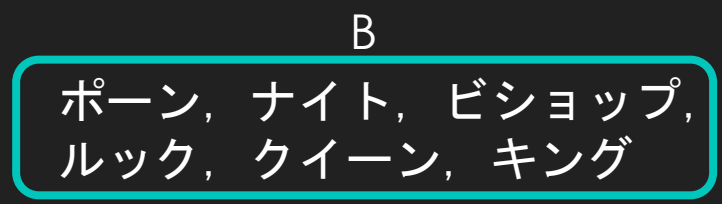
集合

[定義]　集合 (set)

　　議論の対象とするものを集めてひとまとまりにしたもの

集合Aを 8の約数の集合とする.　　集合Bをチェスの駒の集合とする.



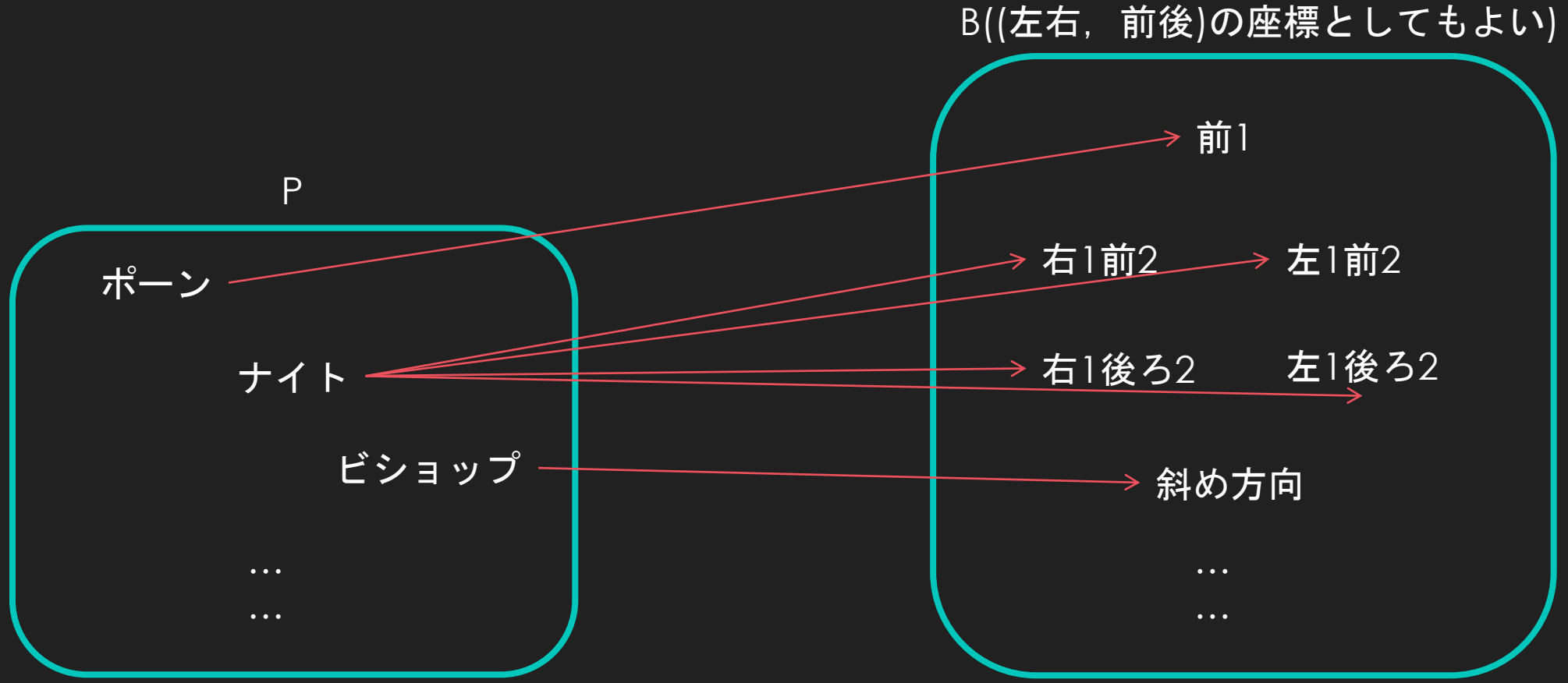


1 ∈ A, 3 ∉ A

クイーン ∈ B,　飛車 ∉ B

ボードゲームのルールをおぼえているときに無意識にやっていること

Pをチェスの駒の集合，Bを駒の動ける場所の集合とする.



ある集合からある集合への対応関係を，(PからBへの)写像という

特別な集合

[定義] 空集合　Φ

要素を一つも持たない集合

[定義] 全体集合・普遍集合 U

議論の対象とするものすべてを集めてひとまとまりにした集合

[例] 中学数学での普遍集合は，実数全体 **R**

ミルクボーイの例の漫才における普遍集合は，朝食のメニュー全体

=> 議論を始める前に，議論の対象を明らかにしてから始める

集合の表し方

- 外延的記法
  - 集合の要素をすべて列挙する.
    - チェスの駒の集合をPとすると，P = {ポーン，ルック，ナイト，ビショップ，クイーン，キング}
    - 空集合をΦとすると， Φ = {}
- 内包的記法
  - 集合の要素が満たす条件を記述する {集合に含まれるもの | 左にある変数の条件}
    - 偶数の集合をE とすると， E = {2n | n ∈ **N**, **N**は整数全体の集合}
    - 普遍集合を実数としたとき，$\{x \mid 2x^2 + 2 = 0\}$ = Φ

Pythonにおける集合の記述

集合の外延的記法と同じ

```
# 集合Aの定義
A = {1, 2, 3, 4, 5 ,6, 7, 8, 9}

# 集合Bの定義
B = {8, 4, 1, 2, 7, 5, 3, 9, 6}

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

print("A=Bであるか?: ", A==B) #要素の順番は考慮しない
print("重複する要素がある場合: ", Bd) #同じ要素は1つずつしか存在しない

# 文字列も集合の要素として扱える
C = {"pawn", "knight", "bishop", "rook", "castle", "queen", "king"}

```

```
A=Bであるか?:  True
重複する要素がある場合:  {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

リストを集合化する関数 set( )

- 巨大なデータから，重複を取り除きながら取り出しをするとき

```
1 Data = [3, 1, 3, 4, 5, 12, 23, 22, 22, 23, 12, 24, 26, 3, 1, 3, 4, 5, 12, 23, 22, 22, 23, 12, 24, 26]
2 print(Data)
3 S = set(Data)
4 print(S)
```

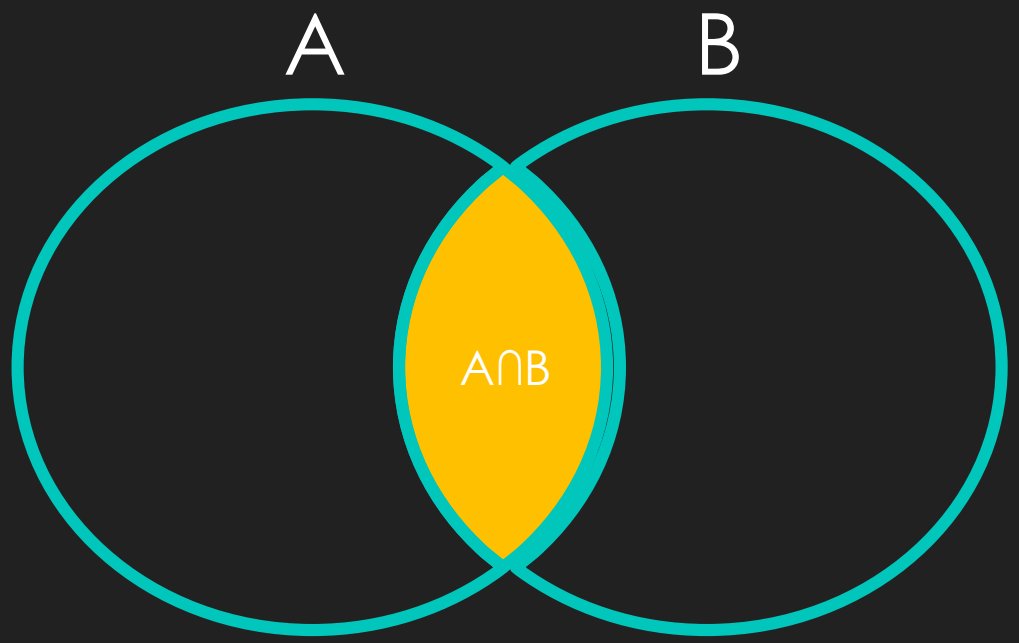
```
[3, 1, 3, 4, 5, 12, 23, 22, 22, 23, 12, 24, 26, 3, 1, 3, 4, 5, 12, 23, 22, 22, 23, 12, 24, 26]
{1, 3, 4, 5, 12, 22, 23, 24, 26}
```

集合の演算

- 共通部分　$A \cap B = \{x \mid x \in A$　かつ $x \in B\}$
- 和集合　$A \cup B = \{x \mid x \in A$ または $x \in B\}$
- 補集合 $A^c = \{x \mid x \notin A\}$
- 部分集合 $A \subset B = \{x \mid$ すべての $x \in A$ について，$x \in B$が成立$\}$

[定義]共通部分　$A \cap B = \{x \mid x \in A \quad \text{かつ} \quad x \in B\}$

- イメージ

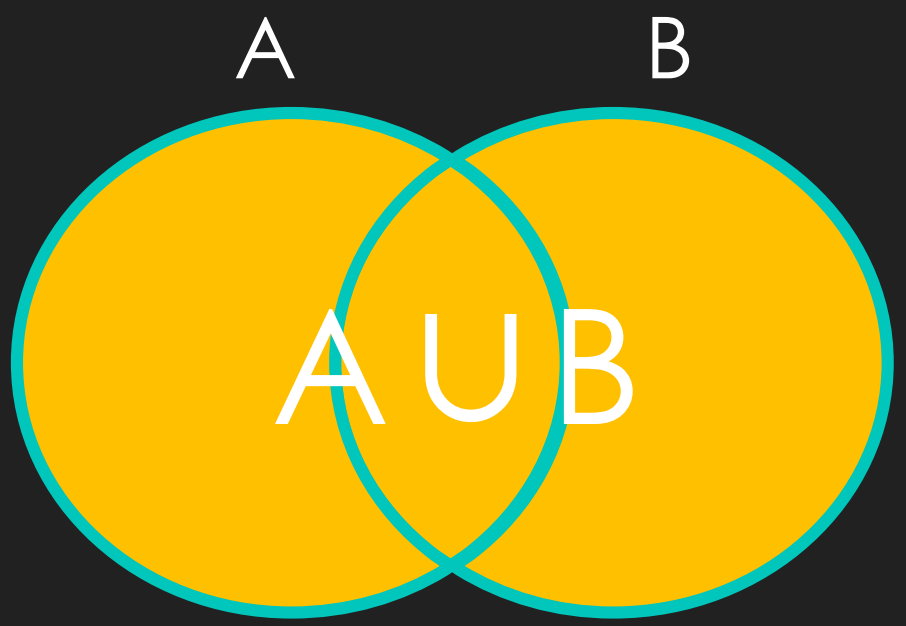


例) Aが3の倍数，Bが5の倍数としたとき，
A = {3, 6, 9, 12, 15, 18, … …}
B = {5, 10, 15, 20, 25, … …}
A∩B = {15, 30, 45, … … }

プログラミング上の問題
=> FizzBuzz

[定義] 和集合 $A \cup B = \{x \mid x \in A \text{ または } x \in B\}$

- イメージ



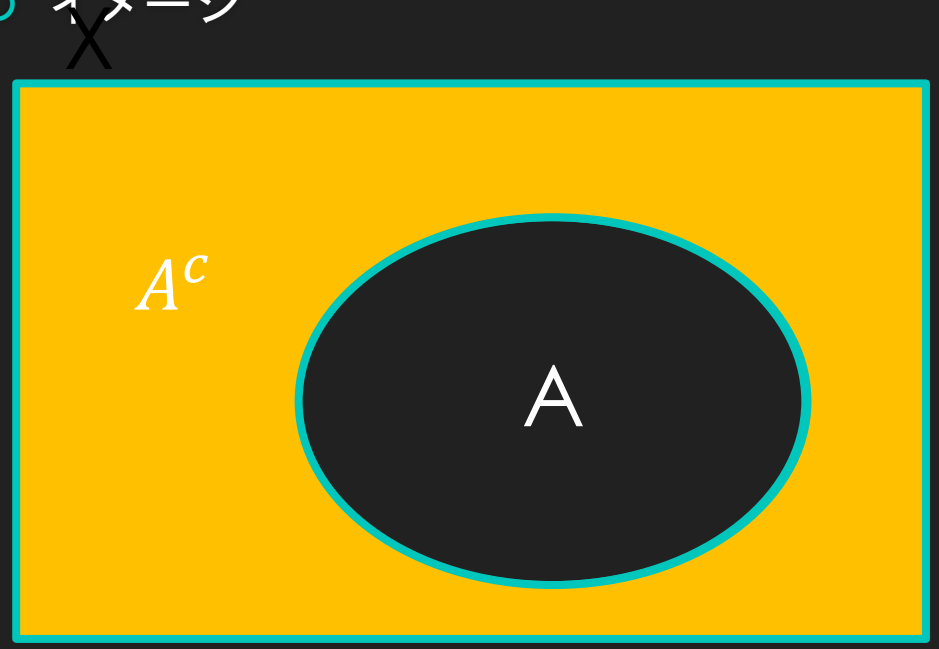
例) Aが3の倍数，Bが5の倍数としたとき，
A = {3, 6, 9, 12, 15, 18, … …}
B = {5, 10, 15, 20, 25, … …}
A ∪ B = {3, 5, 6, 10, 9, 12, 15, 20}

プログラミング上の問題
=> FizzBuzz

論理記号の ∨ (または)に相当する

[定義] 補集合 $A^c = \{x \mid x \notin A\}$

- イメージ

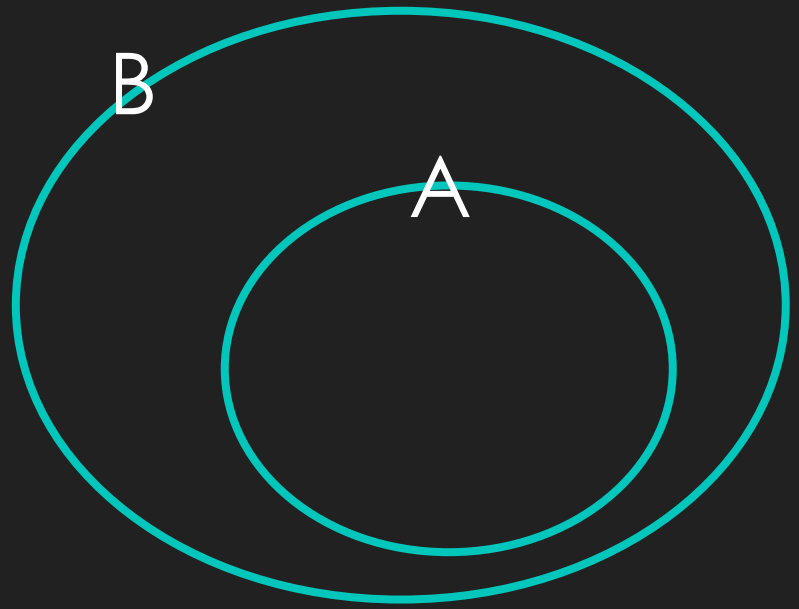


例) Aが3の倍数, Uが自然数全体の集合としたとき,
A = {3, 6, 9, 12, 15, 18, … …}
A^c = {1, 2, 4, 5, 7, 8, 10, 11, 13, … … }

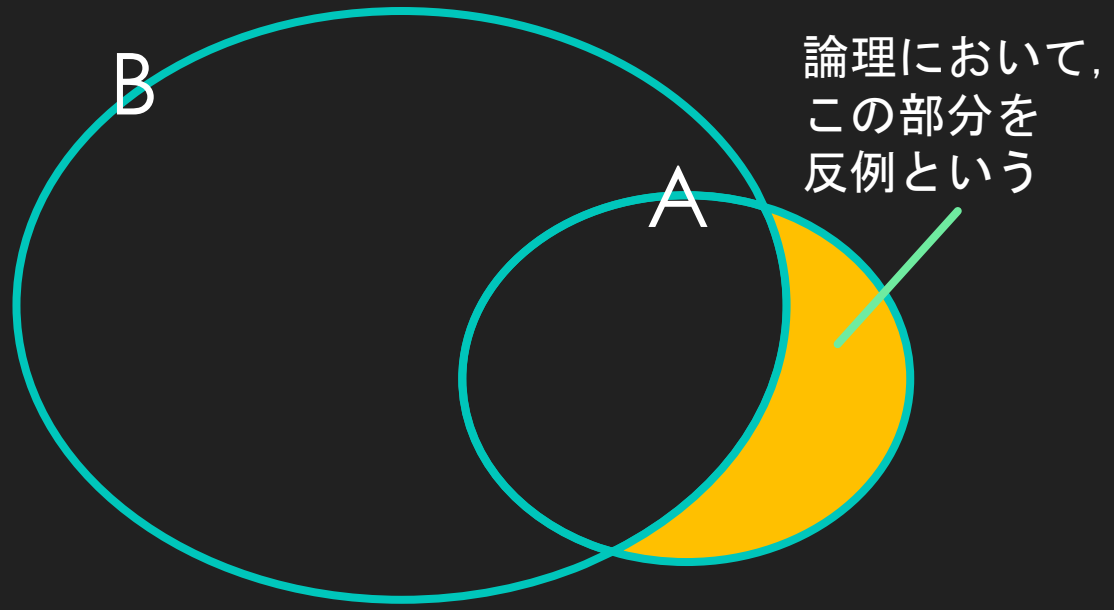
[定義] 部分集合 $A \subset B = \{x \mid$ すべての $x \in A$ について，$x \in B$が成立$\}$

論理記号の$\rightarrow$ (ならば)に相当する

イメージ



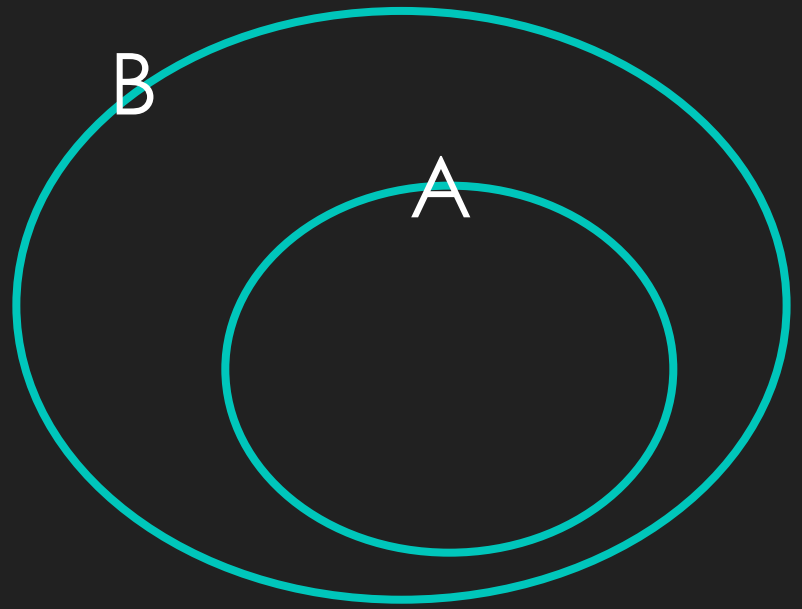
部分集合でない例
（ならばが成立しない例）

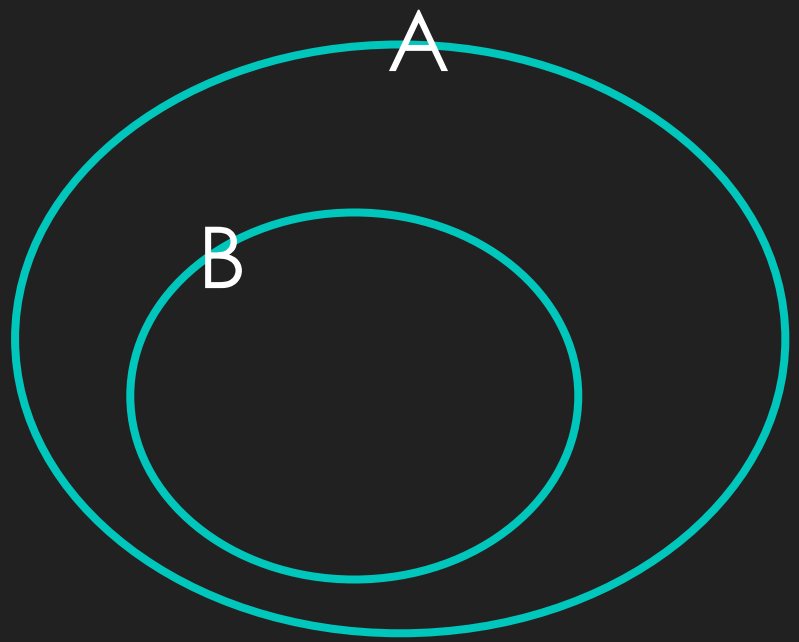


[定義]　等しい (=)

2つの集合AとBが持っている要素が同じとき，AとBは**等しい**と定義する.

A ⊂ B で，かつ A ⊃ Bであるとき，　A = B が成り立つ.





Pythonにおける集合の演算

- 積集合・和集合の演算

```
1 A = {2, 4, 6, 8, 10}
2 B = {3, 6, 9}
3 print("A", A)
4 print("B", B)
5 print("AとBの積集合", A & B)
6 print("AとBの和集合", A | B)
7 print("A=Bの確認", A==B)
```

```
A {2, 4, 6, 8, 10}
B {9, 3, 6}
AとBの積集合 {6}
AとBの和集合 {2, 3, 4, 6, 8, 9, 10}
A=Bの確認 False
```

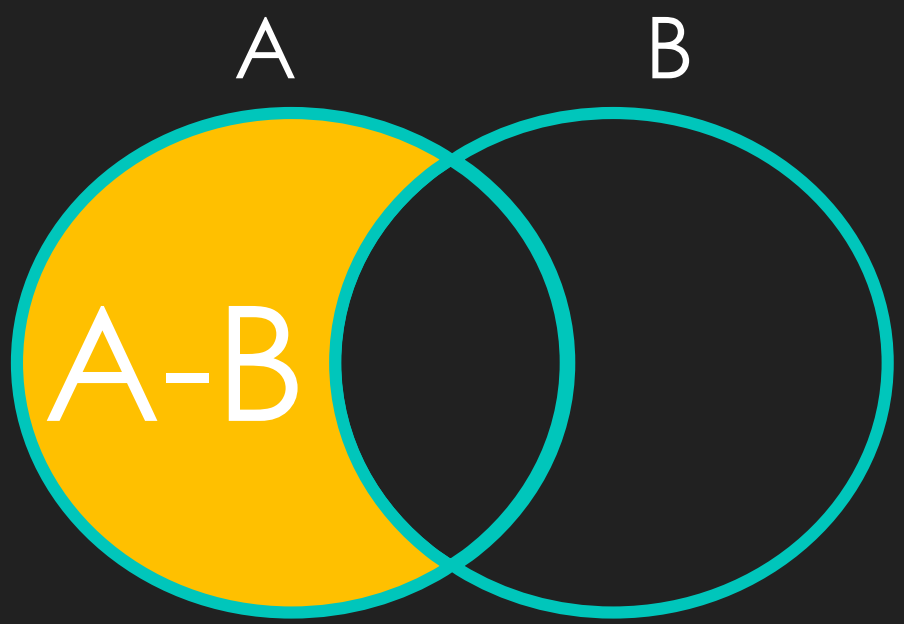
- 部分集合の判定

```
1 U = set(range(1, 11))
2 A = set(range(2, 11, 2))
3 print("AがUの部分集合であるか?", A <= U)
4 print("UはAの部分集合であるか?", U <= A)
```

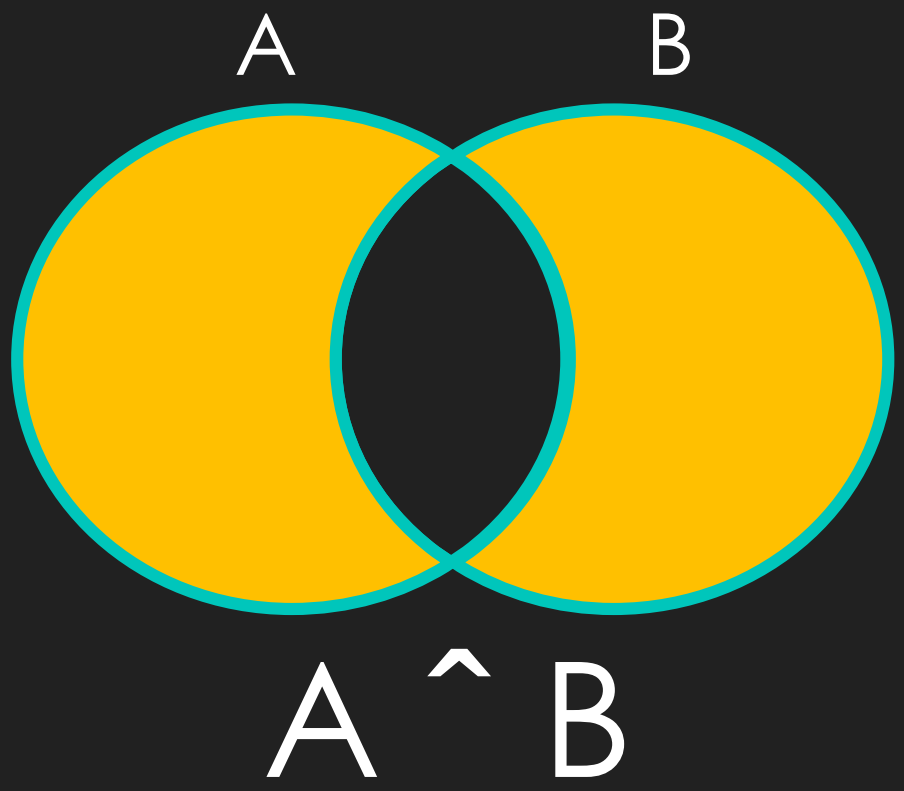
```
AがUの部分集合であるか? True
UはAの部分集合であるか? False
```

集合の演算　応用
（高校でやらないが重要なもの）

[定義] 差集合 $A - B = \{x \mid x \in A \text{ かつ } x \notin B\}$



[定義] 対称差 $A \hat{} B = \{x \mid (A \cup B) - (A \cap B)\}$



差集合 /対称差の演算

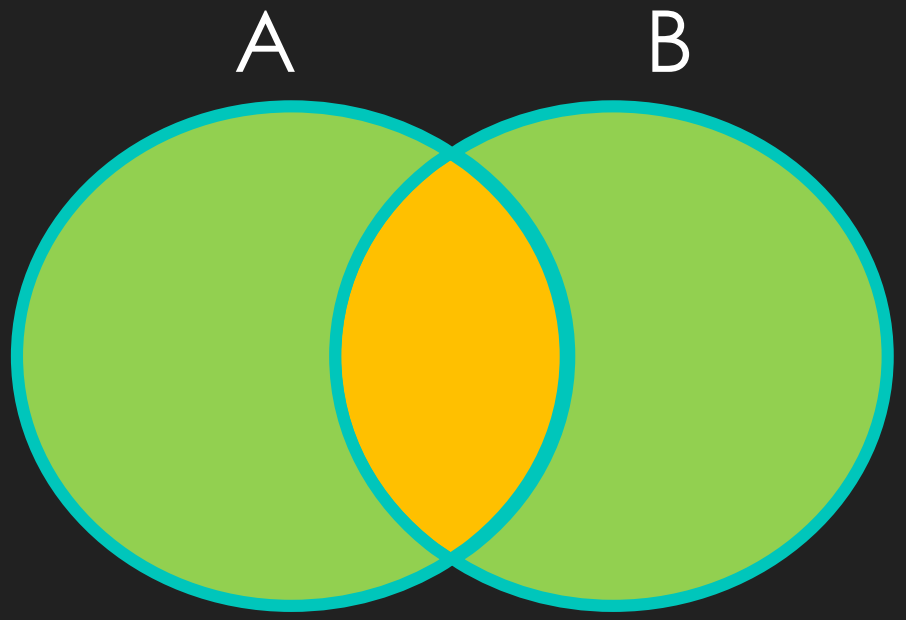
```
1 A = {2, 4, 6, 8, 10}
2 B = {3, 6, 9}
3 print("A", A)
4 print("B", B)
5 print("A-B (差集合)", A - B)
6 print("A^B(対称差)", A ^ B)
7
```

```
A {2, 4, 6, 8, 10}
B {9, 3, 6}
A-B (差集合) {8, 2, 10, 4}
A^B(対称差) {2, 3, 4, 8, 9, 10}
```

包含と排除の原理
The Principle of Inclusion and Exclusion

和集合の個数を数える際の考え方

$$|A \cup B| = |A| + |B| - |A \cap B|$$



問題例: FizzBuzz
http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=2441

【演習】補集合

Pythonには，補集合を求める関数が標準では存在しないため，自分で関数を作る必要がある．

これまでに習った集合の演算を活用して，Uを全体集合，AをUの部分集合としたときの，A^c(Aの補集合)を求めてください．

検証用

U = set(range(1, 10))

A = set(range(2, 10, 2))

できた人は，全体集合の補集合が空集合になること，空集合の補集合が全体集合になることを確認してください．

集合の演算

- 集合をA, B ,Cであらわしたとき，以下が成立

法則	
A ∪ A = A	べき等律
A ∩ A = A	
(A ∪ B) ∪ C = A ∪ (B ∪ C)	結合律
(A ∩ B) ∩ C = A ∩ (B ∩ C)	
A ∪ B = B ∪ A	交換律
A ∩ B = B ∩ A	
A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)	分配律
A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)	

集合の演算

法則	名前
$A \cup \Phi = A$	同一律
$A \cap X = A$	
$A \cup X = X$	
$A \cap \Phi = \Phi$	
$A \cup A^c = X$	補元律
$A \cap A^c = \Phi$	
$X^c = \Phi$	
$\Phi^c = X$	
$A^{cc} = A$	対合律
$(A \cup B)^c = A^c \cap B^c$	ド・モルガンの法則
$(A \cap B)^c = A^c \cup B^c$	

論理と集合の演算　を比較する

- ∧を∩，　∨をUに対応させると，同じ等式が成り立つ

法則
A U A = A
A ∩ A = A
(A U B) U C = A U (B U C)
(A ∩ B) ∩ C = A ∩ (B ∩ C)
A U B = B U A
A ∩ B = B ∩ A
A U (B ∩ C) = (A U B) ∩ (A U C)
A ∩ (B U C) = (A ∩ B) U (A ∩ C)

法則
p ∨ p ≡ p
p ∧ p ≡ p
(p ∨ q) ∨ r ≡ p ∨ (q ∨ r)
(p ∧ q) ∧ r ≡ p ∧ (q ∧ r)
p ∨ q ≡ q ∨ p
p ∧ q ≡ q ∧ p
p ∨ (q ∧ r) ≡ (p ∨ q) ∧ (p ∨ r)
p ∧ (q ∨ r) ≡ (p ∧ q) ∨ (p ∧ r)

論理と集合の演算

法則
$A \cup \Phi = A$
$A \cap X = A$
$A \cup X = X$
$A \cap \Phi = \Phi$
$A \cup A^c = X$
$A \cap A^c = \Phi$
$X^c = \Phi$
$\Phi^c = X$
$A^{cc} = A$
$(A \cup B)^c = A^c \cap B^c$
$(A \cap B)^c = A^c \cup B^c$

法則
$p \vee f \equiv p$
$p \wedge t \equiv p$
$p \vee t \equiv t$
$p \wedge f \equiv f$
$p \vee \neg p \equiv t$
$p \wedge \neg p \equiv f$
$\neg t \equiv f$
$\neg f \equiv t$
$\neg\neg p \equiv p$
$\neg(p \vee q) \equiv \neg p \wedge \neg q$
$\neg(p \wedge q) \equiv \neg p \vee \neg q$

Φ，Xはそれぞれ何に対応している?

Φ => f (矛盾命題)
X => t (トートロジー)

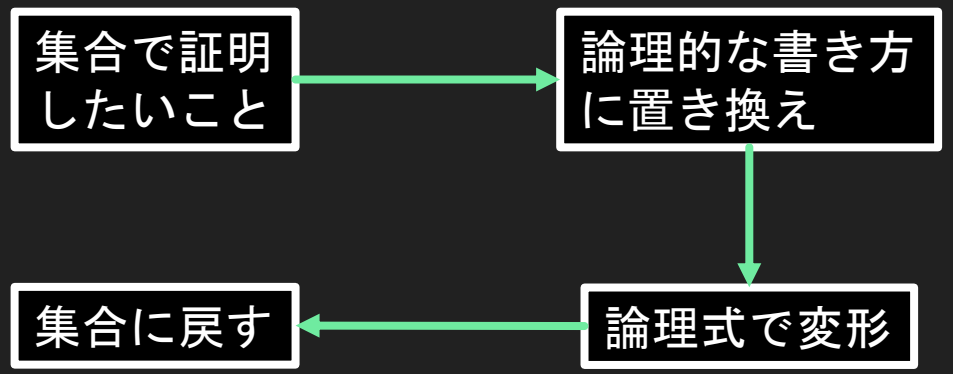
補集合は¬(でない)に対応

集合の演算　証明

集合 A = B の証明方法

1. A ⊃ B かつ A ⊂ B を示す.
2. 論理同値 (≡)を用いて，$x \in A \equiv x \in B$が　成立することを示す.

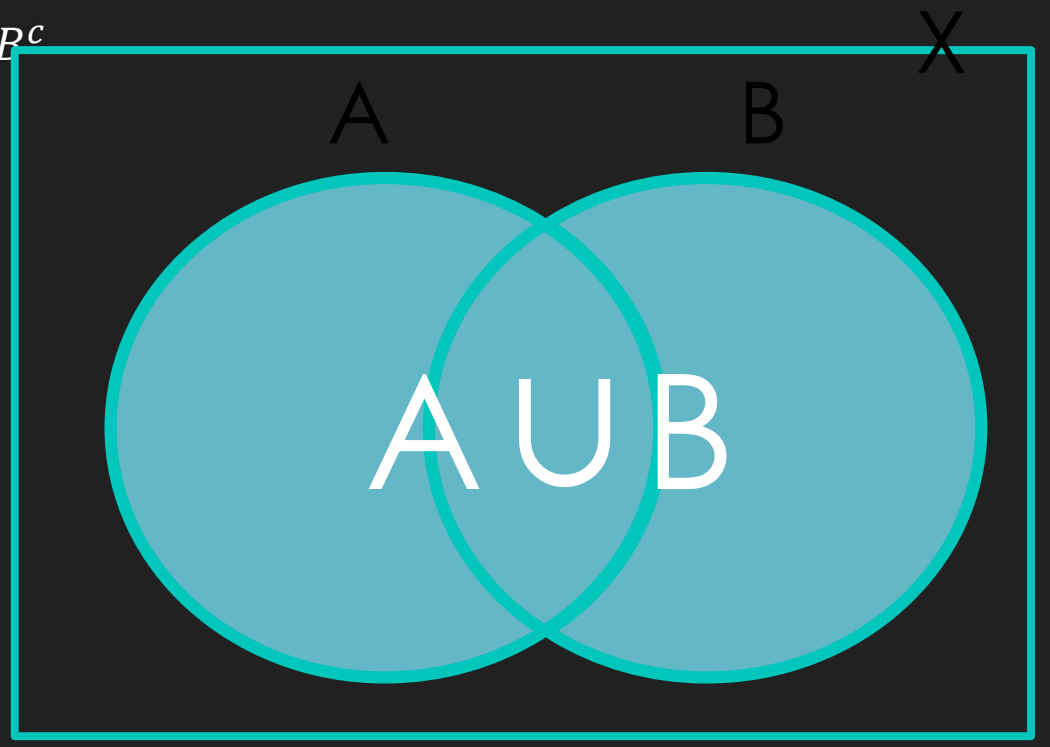
=> 意味が変わらないように言い換えながら，論理的に同じことを導く = 演繹的推論



例

$(A \cup B)^c = A^c \cap B^c$

$x \in (A \cup B)^c$ とすると，
$\leftrightarrow x \notin (A \cup B)$ (補集合の定義)
$\leftrightarrow (x \notin A) \wedge (x \notin B)$ (論理のド・モルガンの法則)
$\leftrightarrow A^c \wedge B^c$ (補集合の定義)
$\leftrightarrow A^c \cap B^c$



ほかの集合も同様にして，定義と，論理を組み合わせて証明ができる．（課題）

順列/組合せ

- なぜ順列・組み合わせを学んだのか?
  - プログラムの挙動すべてを列挙して，エラー処理や障害に対応するため
    - もれなく・だぶりなく（すべての可能性を列挙していなければ，ユーザはかいくぐってくる）
  - データ間の比較の量や計算の量を見積もるため
    - 指定した駅間で最も早い乗り換えルートを提示する
    - 計算に必要な個数や組み合わせをなるべく減らして，プログラムを早く動作させる（スーパーコンピュータ）

順列とは

ある集合から，いくつかのものを取り出して，順序を付けて並べること
（それをすべて列挙すること）

$$nPr = n\,(n-1)(n-2)(n-3)\,\ldots\ldots(n-r+1)$$

例) 0-9までの数字から，3つ選んだ時の順列

012　021

013　023

014　024

…　…

【Game】「順列の総数を減らしてコンピュータの計算の量を減らす」とは?

Numeron

2人で遊べるゲームです

1. 0-9までの数字から3つの数字を選ぶ（重複不可）
2. その数字をGMに報告する
3. 各ターンで，交互に相手の数字を推測しGMに報告する
4. GMは，報告された数字と，本当の数字を見比べて，以下の通りヒントを出す

   数字，場所ともに正解となっている個数を，eat として報告する

   数字はあっているが，場所が異なる数の個数を，bite として報告する

   ex) 相手の実際の数字が **518** で，推測した数字が**521**の場合

   GMのhintコール: 1 eat 1 bite

順列を求めるコード

- itertools: 順列・組み合わせを列挙するのに使えるライブラリ

```
1 import itertools
2
3 num = {1, 2, 3, 4, 5}
4 A = list(itertools.permutations(num, 3))
5
6 for a in A:
7   print(a)
8
9 # 5P3
10 print(len(A))
```

重複順列

- ある集合から，重複して取り出すことを許した順列

$$n\Pi r = n^3$$

重複順列を実装

```python
import itertools

num = {1, 2, 3, 4, 5}
A = list(itertools.product(num, num, num))

for a in A:
  print(a)

# 5Π3
print(len(A))
```

```
(1, 1, 1)
(1, 1, 2)
(1, 1, 3)
(1, 1, 4)
(1, 1, 5)
(1, 2, 1)
(1, 2, 2)
(1, 2, 3)
(1, 2, 4)
(1, 2, 5)
(1, 3, 1)
(1, 3, 2)
```

```
(1, 3, 3)
(1, 3, 4)
(1, 3, 5)
(1, 4, 1)
(1, 4, 2)
(1, 4, 3)
(1, 4, 4)
(1, 4, 5)
(1, 5, 1)
(1, 5, 2)
(1, 5, 3)
(1, 5, 4)
```

```
(1, 5, 5)
(2, 1, 1)
(2, 1, 2)
(2, 1, 3)
(2, 1, 4)
(2, 1, 5)
(2, 2, 1)
(2, 2, 2)
(2, 2, 3)
(2, 2, 4)
(2, 2, 5)
(2, 3, 1)
```

この並び，どこかで… …

[復習] ネストされたfor文

- ネストされたfor文の挙動と，重複順列の中身は同じになる
  => 重複順列や順列の考え方で，for文のループが何回動作するかを見積もることができる

```
1 import itertools
2
3 num = {1, 2, 3, 4, 5}
4
5 for i in num:
6   for j in num:
7     for k in num:
8       print((i, j, k))
9
```

組合せ

ある集合から，決まった個数を取り出す方法について考える（並び順は考えない）

$$nCr = \frac{nPr}{r!}$$

参考になるゲーム (Yahtzee，ヨット)

組合せを表現するプログラム

```
1 import itertools
2 num = {1, 2, 3, 4, 5}
3
4 A = list(itertools.combinations(num, 3))
5
6 for a in A:
7   print(a)
8
9 #nCr
10 print(len(A))
```

```
(1, 2, 3)
(1, 2, 4)
(1, 2, 5)
(1, 3, 4)
(1, 3, 5)
(1, 4, 5)
(2, 3, 4)
(2, 3, 5)
(2, 4, 5)
(3, 4, 5)
10
```

集合・順列・組み合わせに関する問題

Aizu Online Judge の問題例

- FizzBuzz　http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=2441

AtCoder の問題例

- **B - 0 or 1 Swap**　https://atcoder.jp/contests/abc135/tasks/abc135_b
- **C - One-stroke Path**　https://atcoder.jp/contests/abc054/tasks/abc054_c
- **C - Count Order**　https://atcoder.jp/contests/abc150/tasks/abc150_c

その他: 数学の教科書・参考書・JOMの過去問等を参照して興味のある問題を考えてみてください