Python入門 第4回

総合科学研究会

2024.5.10

今回の内容

- ブール型について(復習)
- 論理演算子
- while文による反復処理

復習・ブール型

ブール型とは、論理値の**真(True)**、**偽(False)**の2つの値からなる型である。

例えば比較演算を考える。その比較が正しい場合(成り立つ場合)はTrue、そうでない場合はFalseを返す。

- A=5,B=2,C=2とすると

A>B→5>2→True

A<B→5<2→False

B==C→2==2→True

B!=C→2!=2→False

B>=C→2>=2→True

論理演算子

and条件やor条件、not条件等を論理演算という。

and条件: A and Bという条件式は、AがTrueであり、**かつ**BがTrueである時に限ってA and B全体がTrueになる。

or条件: A or Bという条件式は、AがTrue**もしくは**BがTrueの時にA or B全体がTrueになる。逆にAがFalseかつBがFalseならA or BはFalseとなる。

not条件: not Aという条件式は**Aの真偽値を反転させる。**AがTrueならnot AはFalseであり、AがFalseならnot AはTrueになる。

and演算

条件式Aと条件式Bがあるとする。
ここで、AとBがともに成り立つ時に限って何らかの処理を行いたい場合

```
if A and B:
    do ***
```

というようにand演算を用いる。
and演算の真理値表は右図の通りであり

AがTrueかつBがTrue→A and BはTrue
AもしくはBのどちらかでもFalse
→A and BはFalse
である。

	A=True	A=False
B=True	True	False
B=False	False	False

or演算

条件式Aと条件式Bのどちらか一方でもTrueなら、全体の条件式がTrueになるような条件式

→or演算を用いる。

条件文にor演算を用いる際は

```
if A or B:
    do ***
```

とする。真理値表は右の通りであり

AもしくはBがTrue→A or BがTrue

AかつBがFalse→A or BはFalse

となる。

	A=True	A=False
B=True	True	True
B=False	True	False

not演算

条件式AがTrueならFalse、AがFalseならTrueとしたい場合

→not演算を用いる。

条件式ではnot演算を以下のように用いる。

```
if not A:
    do ***
```

なお真理値表は下の通りになる

A=True	False
A=False	True

論理演算の組み合わせ

論理演算を組み合わせる事で、複雑な条件式を表現することが出来る。

例えば

A and (B or C)：AがTrueかつBまたはCがTrueならば全体がTrueになる。

(A or not B) and (not C or D)：AがTrueもしくはBがFalse、かつCがFalseもしくはDがTrueならば全体がTrueになる。

また、ド・モルガンの法則

not (A and B) = not A or not B : ($\overline{A \wedge B} = \bar{A} \vee \bar{B}$)

not (A or B) = not A and not B : ($\overline{A \vee B} = \bar{A} \wedge \bar{B}$)

も成り立つ。

演習問題3

次の問題を解いてみてください。

(1)昼ご飯を食べに食堂へ行きました。手持ちの現金を変数cashとして表します。現金が1000円以上あるならランチセット、500円以上ならラーメン、500円未満なら水と出力するプログラムを作成してください。

(2)3つの数値変数a,b,cのうち値が最大のものを返すプログラムを作成してください。

ヒント：比較を使って3つの変数から最大値を求めるとすると、実際にどうやって見つける？

(3) 数値変数nが3の倍数である時に"Fizz"、5の倍数である時に"Buzz"、15の倍数である時に"FizzBuzz"が出力されるようなプログラムを作成してください。

反復処理(whileとfor)

反復処理とは

反復処理はプログラムの制御構造のうちの1つであり、ある条件が満たされている間に特定の処理を繰り返し行うものである。

コンピュータの強みが最も発揮される処理といえる。

例：パスワード入力の処理

パスワードがあっていた場合：ドアが開く

パスワードが間違っていた場合：ドアは開かないが改めてパスワード入力をやり直すことが出来る

としたい場合

→「パスワードが間違っている間はドアが開かない」という反復処理により実装することが出来る。

while文による反復処理

Pythonにおける反復処理の一つが**while文**である。

while文の構文は右図のようになっている。

if文と同じように条件式の右に:(コロン)を書き、改行後インデント(半角スペース4つ分)を空けて繰り返したい処理を記述する。

if文と構文が似ているが、while文では条件式がTrueである間、while以下インデントをつけた処理を繰り返し実行する。

```
while 条件式:
    処理1
    処理2
    ...
```

条件式がTrueの間
インデントがついている処理1、
処理2、...を繰り返し実行する

while文の使用例①

1から100までの値の総和を計算するプログラム

連続する自然数の総和は

$$S_n = \frac{1}{2}n(n+1)$$

で与えられるが、このプログラムでは繰り返し処理で計算している。

公式に$n = 100$を代入した結果と比較してみると?

```
[7]  s = 0
     n = 1
     while n <= 100:
       s = s + n
       n = n + 1

     print(s)
```

0秒

```
5050
```

変数sに現在のnの値が加算されていく(n=1,2,3,…として脳内で計算してみると良い)

while文の使用例②

自然数の総和は既に一般の式が与えられていたが、

$$S_n = S_{n-1} + \frac{n+2}{2(n+1)} \ (n \geq 2)$$
$$S_1 = 0$$

のような和を一般的に求めるのは難しい

この場合でも、右のプログラムのように繰り返し処理を行う事で容易に値が求まる。

✓ 0秒

```
[8]  s = 0
     n = 1
     while n <= 1000:
       s = s + (n + 2)/(2*(n + 1))
       n = n + 1

     print(s)
```

```
503.2432349307743
```

while文の使用例③

右の図のプログラムはfinishと入力されるまで、

①文字列を入力する

②入力した文字列をそのまま返す

を繰り返すものである。

文字列変数textに入力した文字を格納しておき、それが”finish”と一致しない場合、ループが続くというプログラムになっている。

```
>>> text = "" # text に初期値として "" を設定する

    while text != "finish": # text が "finish" ではない間、処理を繰り返す
        # 文字を入力する
        text = input("finishと入力してください:")  # text に文字を読み込む
        print(text, "と入力されました。")

    print("終了しました")
```

```
> finishと入力してください: abc
abc と入力されました。
> finishと入力してください: finis
finis と入力されました。
> finishと入力してください: finish
finish と入力されました。
終了しました
```

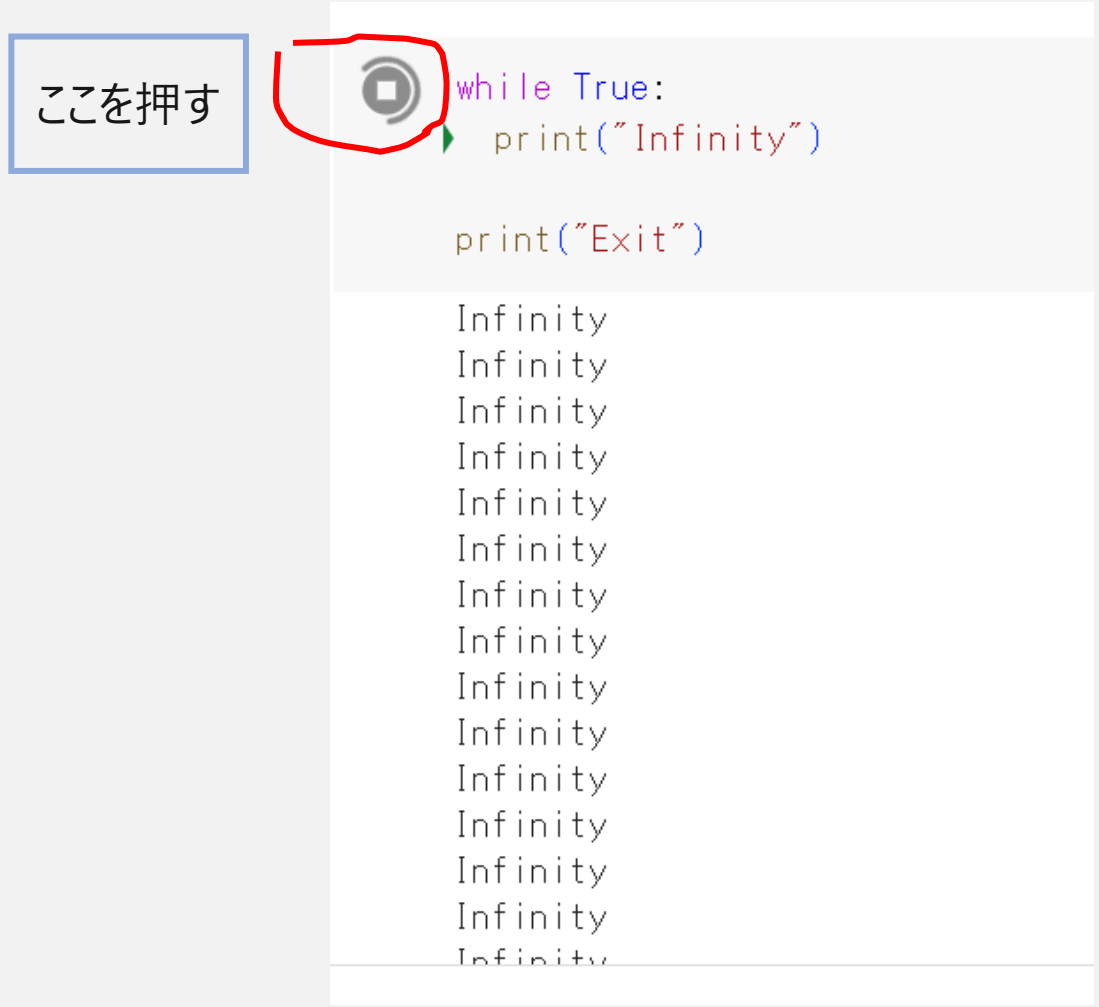
無限ループ

右のプログラムを実行してみると、それ以降ループ処理が終了する事はなく、延々と実行される。

これはwhile文の条件式が常にTrueとなるためである。

明らかに停止しないような繰り返し処理の事を無限ループと言ったりする。

Google Colabでは実行セルの左側の■を押す事でプログラムが強制的に終了する。無限ループになってしまった場合はこれを使う。



breakによる中断

ループを条件式に関わらず途中で強制的に終了させる構文が**break文**である。

右のプログラムではwhile以下で変数counterを1から加算していき、5以下であれば入力した数字を出力する処理を行っている。

この時、textに数字の”999”が入力されると、break文によってループが終了する。

```
counter = 1 # 現在、何回目かを記録する変数

while counter <= 5:  # counterの値が5以下なら繰り返す
    text = input("数字を入力してください")

    # 入力された文字が '999' なら
    if text == '999':
        # ループを中断する
        print("中断します")
        break

    number = int(text) # 入力した文字列を数値に変換する
    print(counter, "回目:", number * number) # 入力した数
    counter = counter + 1  # counter の値に 1 を加算する

print("終了しました")
```

continue文

break文はループの途中で強制終了するものだったが、途中でループの先頭に強制的に戻る**continue文**という機能もある。

右のプログラムはcontinue文を使ってiが奇数の時はそれ以降の処理を行わないようにループ先頭に戻るようにしている。

その結果、1から10までの偶数のみを出力するようになっている。

```
>>> i = 0
while i < 10:
    i = i + 1

    if (i % 2) == 1: # iを2で割った余りが1のとき
        continue

    print(i, "is even.") # 値を出力
```

```
2 is even.
4 is even.
6 is even.
8 is even.
10 is even.
```