

14 while 文による繰り返し

上の問題は $3n + 1$ 問題に関するものでした。そこでは自然数 n に対して、もし n が偶数なら n を半分にし、 n が奇数なら n を $3n + 1$ にする、という操作を n の値を表示しながら、最終的に $n = 1$ になるまで繰り返し行うものでした。そこでは、for 文を使い、繰り返しの回数は最大 1000 回までと決めていましたが、この数列は何回のステップで $n = 1$ になるか分からないし、そもそもどんな自然数から始めても最後に $n = 1$ ということは証明されていません（反例も見つかっていません）。このように、何回繰り返すかわからない場合は for 文ではなく、次に説明する while 文によって記述しなければなりません。

14.1 while 文の文法

ある処理の繰り返しを行うときに、繰り返す回数がわかっている場合は for 文を使いますが、繰り返す回数分からない場合や永遠に繰り返しを行う場合には while 文によってプログラムを記述します。

while 文

```
1 while 条件A:
2     [ 処理X ]
3     [ 処理X ]
```

- 条件 A が True であるあいだ、ずっと処理 X を繰り返す。
- 条件 A が False なら処理 X は行われず while 文は終了する。もし A が True だったら処理 X が実行され、X の終了後にまた条件 A をチェックします。同様に、A が True ならば処理 X を行い False なら while 文は終わり。この繰り返しは、条件 A が True である限りずっと続きます。
- オプションとして break, continue を使うとより柔軟な制御が可能になります：
 - 処理 X の実行中に break が現れると強制的に while 文から抜けます。
 - 処理 X の実行中に continue が現れたら、処理 X を中断し条件 A を確認するプロセスに戻る。

while 条件A:



図4 while 文の文法

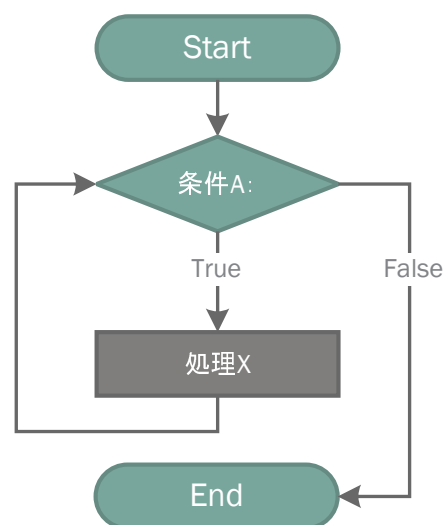


図5 while 文の処理の流れ

次の while 文の簡単な例を見てみましょう。

- ファイル名 : **while1.py**

```

1 a = 1
2 while a<10:    # aが10未満なら、3-4行目を行う。
3     print(a*a, end=' ')    # aの二乗をプリントする
4     a = a+1    # aの値を一つ増やす。2行目に戻る。
5
6 print('owari')    # while文を抜けたらこの処理を行う

```

実行結果の例

```
1 4 9 16 25 36 49 64 81 owari
```

ここで、上の説明での条件 A に相当するものは『 $a<10$ 』で、処理 X は 4,5 行目です。

上のプログラムのように、while 文の中にカウンター（一つずつ増える変数）と停止条件を書くことで、for 文と同じ処理を行うことができます。上のプログラム `while1.py` と同じ事を for 文で書くこともできます。

```

1 for a in range(1,10):
2     print(a*a, end=' ')
3
4 print('owari')

```

14.2 while 文の例

14.2.1 $3n+1$ 問題の数列の生成

while 文を用いて前の節の問題（ n が偶数なら半分にし、奇数なら $3n+1$ に変えることを繰り返してできる数列を作る問題）を書き直してみましょう：

- ファイル名 : **while2.py**

```

1 a = int(input('Input an integer: '))    # 数を入力させて、その数を aに入れる
2 print(a, end=' ')    # aの値をプリント
3
4 while a != 1:    # a≠1である限り以下の処理を繰り返し行う
5     if a%2 == 0:    # aが偶数なら
6         a = a//2
7     else:    # aが偶数でないなら
8         a = 3*a+1
9     print(a, end=' ')

```

実行結果の例

```
Input an integer: 19
19 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

前回の課題と同じ結果が得られました。もっと大きな数ではどうなるでしょうか？例えば $a=6171$ から始めると 261 回の繰り返しの末に最終的に $a=1$ となります。

実行結果の例

```
Input an integer: 6171
6171 18514 9257 27772 13886 6943 20830 10415 31246 15623 46870 23435 70306 35153 105460
... 略 ...
1300 650 325 976 488 244 122 61 184 92 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
```

前回のプログラム `collatz1.py` では繰り返しの回数に上限がありましたが、`while2.py` では無制限の繰り返しの計算を行うことができます（もちろんコンピューターの容量の制限の中ではありませんが）。

14.2.2 繰り返しが終わらないプログラム

次に while 文を使って永遠に終わらないプログラムを作ってみましょう：

- ファイル名 : **while3.py**

```

1 j=1
2 while True:      # 条件は永遠に真なので、以下をずっと繰り返す！
3     print('ひつじが'+str(j)+'匹')
4     j = j+1

```

実行結果の例

```

ひつじが 1 匹
ひつじが 2 匹
ひつじが 3 匹
...

```

この while 文では条件がいつでも真 (True) なので繰り返し処理が永遠に続きます。

プログラムを終了するには CTRL + C と入力します。

14.2.3 while 文の中で break, else, continue を使った例

while 文の繰り返しは break を書くことによって止めることができます：

- ファイル名 : **while4.py**

```

1 j=1
2 while True:
3     print('ひつじが'+str(j)+'匹')
4     j=j+1
5     if j > 123456: # もし羊の数の二乗が123456を超えたらwhile文を終える
6         break
7
8 print("Too much sheeps! I'm already asleep...zzz")

```

実行結果の例

```

ひつじが 1 匹
ひつじが 2 匹
ひつじが 3 匹
...
ひつじが 123456 匹
Too much sheeps! I'm already asleep...zzz.

```

while 文でも else を使うことができます。while 文の条件が False になったときだけに行う処理を書きます。while 直後の条件が成立しなければ else 以下を処理します：

- ファイル名 : **while_else.py**

```

1 a=0
2 while a<5:
3     a = a+1
4     print(a)
5 else:
6     print('owari')
7
8 print('END')

```

実行結果の例

```

1
2
3
4
5
saigo
END

```

上のプログラムは意味のないプログラムですが、後で紹介するように `break` を `while-else` 文の中で使うと効果的なプログラムを作ることができます。

次に `while` 文の中で `continue` を使った文を作ります。その前に、ある文字列の中に特定の文字 (列) が入っているかどうかを確認する方法を紹介します。次は文字列 `Shinshu` の中に `ns` をいう文字が入っているかを調べています。

```

1 >>> 'ns' in 'Shinshu'
2 True
3 >>> 'sn' in 'Shinshu'
4 False

```

さて、羊を数えたいのですが、数字の 4 は不吉に感じるので、匹数に 4 を含むものは飛ばしながら最大 15 匹まで数えることにしましょう。

- ファイル名 : `while_continue.py`

```

1 a=0
2 while a<15:
3     a=a+1
4     if '4' in str(a): # もし aの中に数字4が含まれていたら
5         continue # 次の printを行わずに whileの条件確認に戻る
6     else: # そうでなければ
7         print('羊が'+str(a)+'匹') # 羊を数える
8
9 print('zzz...')

```

実行結果の例

```

羊が 1 匹
羊が 2 匹
羊が 3 匹
羊が 5 匹
羊が 6 匹
羊が 7 匹
羊が 8 匹
羊が 9 匹
羊が 10 匹
羊が 11 匹
羊が 12 匹
羊が 13 匹
羊が 15 匹
zzz...

```

実は上のような処理は `for` 文を用いたほうが簡潔に書けます :

```

1 for i in range(1,16):
2     if not '4' in str(i): # もし iの中に数字4が含まれていないのなら
3         print('羊が'+str(i)+'匹')
4
5 print('zzz...')

```

14.2.4 フィボナッチ数列の生成

フィボナッチ数列 1, 1, 2, 3, 5, 8, 13, 21, 34, …, つまり漸化式

$$a_1 = 1, \quad a_2 = 1, \quad a_{n+1} = a_n + a_{n-1}, \quad n = 2, 3, 4, \dots \quad (1)$$

で定義される数列を表示するプログラムを作ってみましょう。ただし、数列の値が 100000000 を超えたら停止するものとします。この場合でも、いつ a_n が 100000000 を超えるか分からないので while 文を使います。プログラムは次のアルゴリズムに従って書くことにしましょう：

1. maxvalue = 100000000 と置く。
2. a=1 と置く。b=1 と置く。
3. while 文で a<maxvalue が成り立っている間は、次の処理 4-5 を繰り返す。
4. a の値をプリントする。
5. a, b の値をそれぞれ b, a+b に置き換える。
6. a<maxvalue が偽になって while 文を抜けたら、次の数列の値をプリントする。

● ファイル名 : while5.py

```

1 maxvalue = 100000000      # maxvalueを右辺の数字にセットする
2 a = 1                    # a = 1 とおく
3 b = 1                    # b = 1 とおく
4 while a < maxvalue:
5     print(a)              # ここが
6     a, b = b, a+b        # 繰り返されるブロック
7
8 print('The next value is', a)    # the next value isとaの値をプリントする

```

実行結果の例

```

1
1
2
3
5
...
63245986
The next value is 102334155

```

14.2.5 素数判定プログラム

次に与えられた自然数 (≥ 2) が素数かどうか判定するプログラムを作ってみましょう。与えられた数 n を 2 から順に \sqrt{n} 以下のすべての自然数で割ってみて、どれかの数で割り切れたら n は合成数、そうでなければ n は素数です。そこで次の手順で判定する事にします：

1. 数 n の値を入力させる (ただし $n \geq 2$ とする)。
2. $i = 2$ とする。
3. $i^2 \leq n$ である間は、次の手順 4-5 を繰り返す。
4. もし、 n で i で割り切れたら「 n は素数ではありません」とプリントして 3 の繰り返しを終了する。
5. そうでなければ i を $i + 1$ にする。
6. もし 3 の条件全てが偽であったら、「 n は素数です」とプリントする。

● ファイル名 : while_primeQ.py

```

1 n = input('Input an integer(>1): ') # キーボードから入力した数を変数nとする
2 n = int(n)                          # nを整数に変換

```

```

3 i = 2                                # i=2 とおく
4
5 while i*i <= n:                      # i*i ≤ nなら以下を繰り返す
6     if n % i == 0:                  # もし nがiで割り切れたら
7         print(n, 'is not a prime.') # nは素数ではないとプリント
8         break                       # while文抜けて12行目以下へ進む
9     else:                            # そうでなければ
10        i += 1                       # iの値を一つ増やす
11 else:                                # while文の条件が偽となったら
12     print(n, 'is prime.')          # nは素数であるとプリントする。
13
14 print('おわり')

```

実行結果の例

```

Input an integer: 1237
1237 は素数です。
おわり

```

実は上のプログラムでは $n = 1$ が素数と判定されてしまいます。

15 練習問題

15.1 問題：素数判定プログラム

上のプログラム (while_primeQ.py) を修正し、 $n = 1$ が素数でない判定されるようにしなさい。

15.2 問題：ユークリッドの互除法

ユークリッドの互除法とは自然数 a, b の最大公約数を求める次のアルゴリズムの事です。

- (i) $b = 0$ なら a が最大公約数である。
- (ii) $b \neq 0$ なら a と b の最大公約数は b と r の最大公約数に等しい。ただし r は a を b で割った余り。

次の手順を行う Python プログラムを書きなさい：

1. a, b の値を入力させる。
2. $b \neq 0$ である間は、次の処理 3,4,5 を繰り返し行う (while 文を使う)。 $b = 0$ ならステップ 6 へ。
3. a を b で割った余りを r と置く。
4. a に b を代入
5. b に r を代入
6. $b = 0$ になったらそのときの a が最大公約数なので a をプリントする。

次がプログラムの例です：

- ファイル名：gcd.py

```

1 a = int(input('Input an integer: '))
2 b = int(input('Input an integer: '))
3
4 while b != 0:
5     *****
6     *****
7     *****

```

```
8 |  
9 | print('The greatest common divisor is', a)
```

上の*****の部分を考え、プログラムを完成させなさい。