

問題集（アシアルテキスト対応）Ⅵ 繰り返しと条件分岐の組み合わせ

繰り返し（反復）と、選択（分岐）を組み合わせると、複雑なプログラムを作ることができます。

これまで、様々な問題を解くために、その手順（アルゴリズム）が考えられてきましたが、多くのアルゴリズムは繰り返し（反復）と選択（分岐）の組み合わせで出来ています（※もちろん、繰り返しも条件分岐も使わない問題・アルゴリズムもあります）。

たとえば、「2つの整数の最大公約数を求める」という問題を解く手順の1つに、『ユークリッドの互除法』があります。この手順も、繰り返し（反復）と選択（分岐）の組み合わせで出来ています。

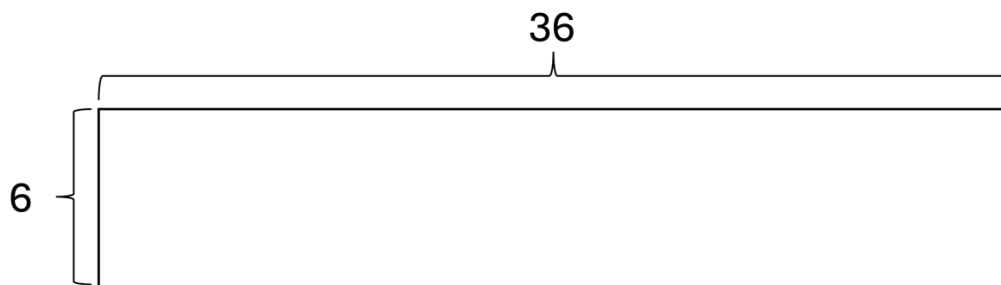
この問題では、「ユークリッドの互除法」を実現するプログラムを作成しますが、プログラムを書く前に、図も使いながら、その考え方を確認します。

ユークリッドの互除法の基本の考え方は、およそ次の通りです。

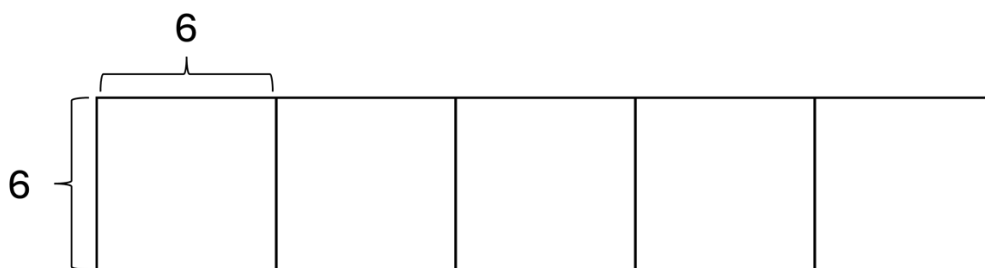
1. 2つの値を、それぞれ長方形の辺の長さとし見なす
2. 1.で得られた長方形を、「短辺の長さを辺の長さにする正方形」に分ける。具体的には、長辺の長さを短辺の長さで割り、余りを求める
 - (ア) 余らずに全て正方形に分けることができたなら、短辺の長さが最大公約数である
 - (イ) 余りが1なら、最大公約数は1となる。2つの値は互いに素である（※2つの整数の間に、公約数が1しかない）
 - (ウ) あまりが0、1でなければ、余った部分の長方形に注目して、手順の1に戻る

例を用いて説明します。

1つ目の例は、6と36の最大公約数を求める例です。6と36の2つの値を、長方形の辺の長さとし見なします。図にすると下の通りです。

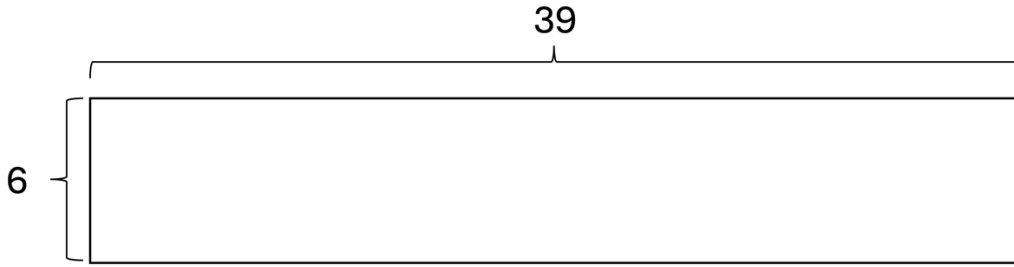


短辺（6）の長さを辺とする正方形で、この長方形を区切っていく（＝長辺（36）を短辺（6）で割ると、余りがありません）。

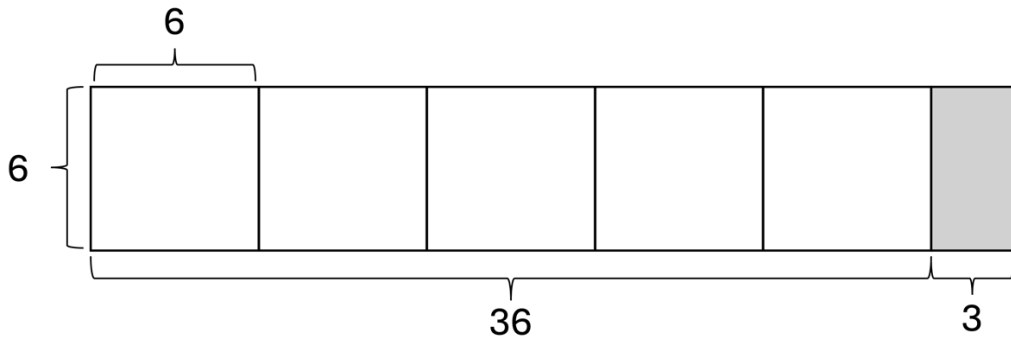


このとき、短辺の長さ（6）が、6と36の最大公約数になります。

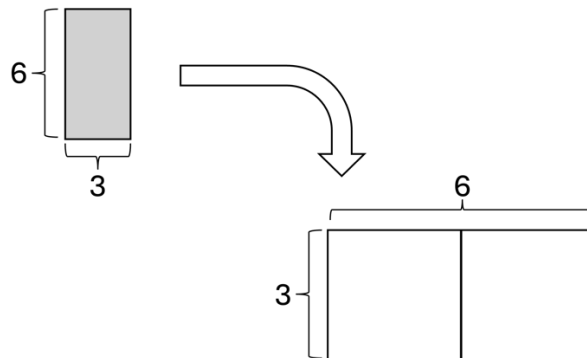
次に、6と39の最大公約数を求める例を考えます。2つの値を、長方形の辺の長さとしなします。



短辺（6）で長辺（39）を割ると、余りが生じます。

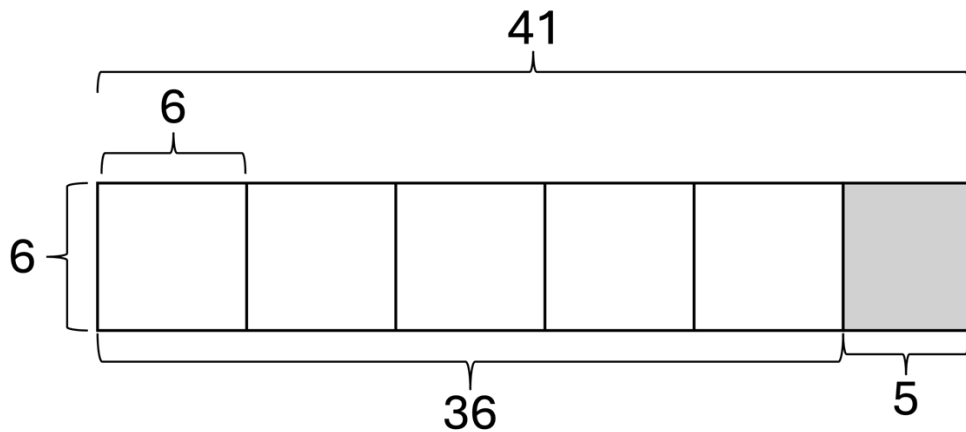


余りの部分に注目すると、長方形と見なせます。

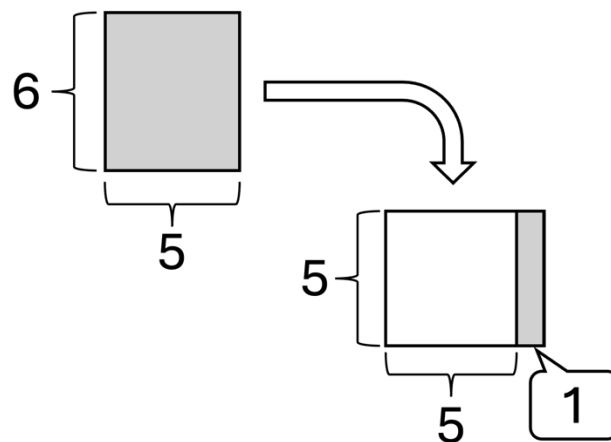


余りの部分の長方形について、短辺（3）の値で長辺（6）を割ります。今度は余りが0になります。従って、6と39の最大公約数は3です。

最後の例は、6と41です。2つの値を、長方形の辺の長さとしなします。



短辺（6）で長辺（41）を割った余りは5で、余った部分は短辺5、長辺6の長方形になります。



短辺（5）で長辺（6）を割ると、余りは1です。

この場合、6と41の間の最大公約数は1ということになり、互いに素であることになります。

以上が、問題「2つの整数の最大公約数を求める」の、1つの解法の手順『ユークリッドの互除法』です。

短辺で長辺を割って、余りを求め、その余りを使って短辺を割って…と、繰り返します。その繰り返しの中で、

- もし、余りが0になったら最大公約数が分かったことになります。
- もし、余りが1なら、最大公約数は1で、2つの整数は互いに素であると分かったことになります。

ユークリッドの互除法は、繰り返しと、条件分岐の組み合わせで出来ています。

以下の問題を解く際は、『Pythonで学ぶプログラミング入門』第6章 繰り返し（反復）と選択（分岐）の組み合わせがヒントになります。

問題 関数の定義と利用

プログラムの冒頭で、ユーザーから2つの整数の入力を受け付けます。

```
a = int(input("1つ目の整数を入力してください"))  
b = int(input("2つ目の整数を入力してください"))
```

ここで入力された2つの整数について、最大公約数を求めるプログラムを作ります。

1. 入力された値を比較して、大きい方を前の変数に代入する

変数 a と、変数 b には、ユーザーが入力した整数が入っています。

この変数 a, b の値を比較して、値が大きい方が変数 a に、小さい方が変数 b に入っているようにしてください。

※最初から a の方に大きい値が入っていたら、交換する必要はありません。

```
if a < b:  
    temp = a  
  
print("大きい値:a = ", a)  
print("小さい値:b = ", b)
```

2. 大きい値を小さい値で割った、余りを求める

変数 a を変数 b で割った余りを求め、それを変数 amari に代入してください。

```
amari =
```

3. ユークリッドの互除法を使い、最大公約数を求める

ユークリッドの互除法を使い、最大公約数を求めるプログラムを作ってください。

- 繰り返しを使います。
- 余りが0になったとき、割る数が最大公約数です。
 - それ以上計算する必要は無いので、break で繰り返しを終了します。
- 余りが1になったとき、2つの数は互いに素(1しか公約数が無い関係)です。
 - それ以上計算する必要は無いので、break で繰り返しを終了します。

```
while True:
    amari =
    if amari == :
        print("最大公約数は", b, "です")
        break
    elif amari == :
        print("最大公約数は 1 で、2つの値は互いに素です")
        break
    else:
        a =
        b =
```

(補足) なぜユークリッドの互除法のような手順を使うのでしょうか？

2つの整数の最大公約数は、次の手順でも求めることができます。

1. それぞれの整数の約数を全て挙げる
2. それぞれの約数の中から、共通するものを選ぶ
3. 共通する約数の中から、最大のものを選ぶ

2桁か3桁の、小さい値の最大公約数を求める際には、この手順は簡単で便利です。約数の数はそれぞれ数個か十数個程度だからです。

しかし、「2つの12桁の数字」のような、大きな数字同士の最大公約数を求める場合、それぞれの約数の数を数え上げるのも大変です。

ユークリッドの互除法の手順は、大きな値でも、比較的少ない回数の計算で最大公約数を求めることができます。

ある問題を解く手順(アルゴリズム)が、複数あることがあります。手順それぞれに特徴・長所・短所があり、状況に応じて使い分けることで、問題をうまく解くことができます。