

# マイクラフトで プログラミング

第17回 アイテムを識別してみよう

# 目標

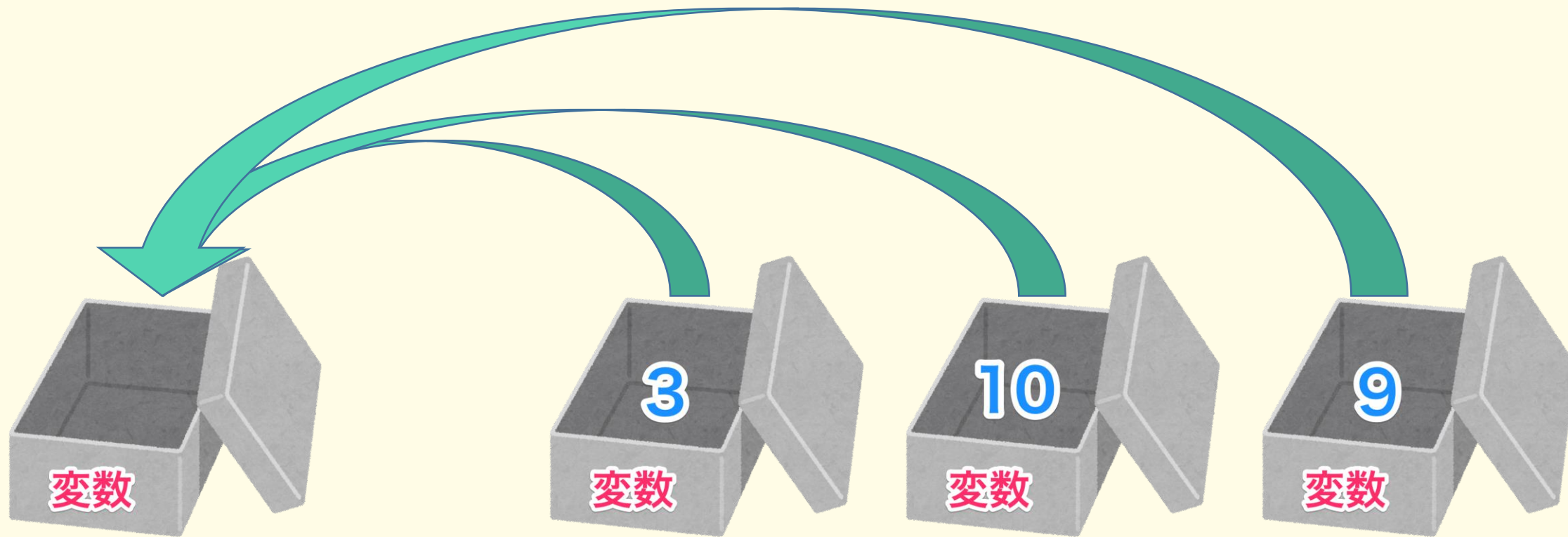
- Table(テーブル)型を使ってみよう
- アイテムを識別してみよう

# ちょっと変わった変数



今日は少し変わった**Table**(テーブル)型と  
言われる変数を使ってみよう

# 変数のなかにいくつかの変数



いくつかの変数を入れることができる変数を  
イメージしてみよう

# Table

```
tbl = {3, 9, 10}
```

変数 **tbl** の中に

**3と9と10が入っている**

さっきの図をプログラムに書くとこんな感じ  
この変数tblが**Table (テーブル) 型**といわれるものだよ

# Tableの作り方

変数名 = { 値1, 値2, ... }

変数を作るときとほとんど一緒に、違うのは右側だね

値の部分には数, 文字, TrueとFalse(真偽値), 変数, テーブルを入れることができるよ

また, 値は「, 」で区切ればいくらかでもTableに入れることができるよ

# Tableの中身を読む

```
tbl = {3, 9, 10}
print( tbl[1] )
```

← 値の番号は左から順に1,2,3,...

← tbl変数1番目の数の  
3が表示される

Table型の中身を読むときは **変数名[番号]** で  
読むことができるよ

# 中身を書きかえる

```
tbl = {3, 9, 10} ← テーブルをつくる
```

```
print( tbl[1] ) ← テーブル内の1番目の値を表示する
```

```
tbl[2] = 100 ← テーブル内の2番目の値を100にする
```

```
print( tbl[2] ) ← テーブル内の2番目の値を表示する
```

読むときと同じように **変数名**[**番号**] = **値** で  
中身の値を書きかえることができるよ



# 課題1 Tableを使ってみよう

```
tbl = {3, 9, 10} ← テーブルをつくる
```

```
print( tbl[1] ) ← テーブル内の1番目の値を表示する
```

```
tbl[2] = 100 ← テーブル内の2番目の値を100にする
```

```
print( tbl[2] ) ← テーブル内の2番目の値を表示する
```

上の画像のコードをそのまま書いて  
実行してみよう。動きについて確認しよう！

# for文のループカウンタ

```
tbl = {10,9,8,7,6}
```

```
for i=1,5 do  
  print(tbl[i])
```

```
end
```

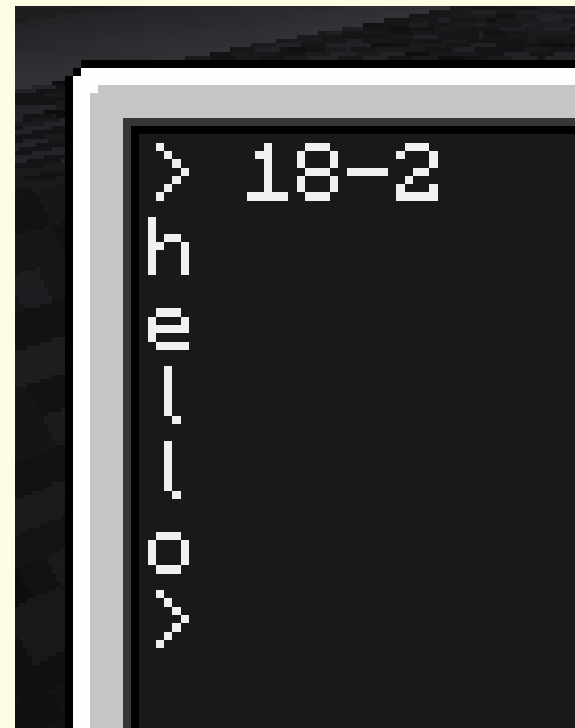
ループカウンタが1から5になるまでくり返す

ループカウンタの値の番号の  
テーブルの値を表示する

ループカウンタをうまく使えばテーブルの中身を  
すべて取り出して使うことができる

## 課題2 for文を使って表示

```
tbl = {"h", "e", "l", "l", "o"}
```



A terminal window with a dark background and light gray text. The prompt is '>'. The first line of output is '18-2'. The second line is 'h'. The third line is 'e'. The fourth line is 'l'. The fifth line is 'l'. The sixth line is 'o'. The prompt '>' is visible at the end of the sixth line.

「h」「e」「l」「l」「o」の文字をTable型に入れて、for文を使ってすべて表示してみよう

# 値に名前を付ける

```
tbl = { tech = 10, chance = 20 }
```

**techに10の値, chanceに20の値が入ったtbl変数を作る**

1番目, 2番目ではなく, 「**tech**」のように  
Table型では値に名前をつけることができるよ

# 名前のついた値の読み書き

```
print( tbl["tech"] )
```

tbl変数のtechという値を  
表示する

```
tbl["chance"] = 100
```

tbl変数のchanceという値の  
中身を100にする

名前のついた値は番号のかわりにその名前を  
使って **変数**["**名前**"] と書けば読み書きできるよ

# 課題3 りんごとみかん

```
> 18-3  
mikan or ringo  
mikan  
50  
>
```

**mikanと入力すれば  
みかんの値段50円を表示**

```
> 18-3  
mikan or ringo  
ringo  
80  
>
```

**ringoと入力すれば  
りんごの値段80円を表示**

**ringo**と入力すればりんごの値段を**mikan**と入力すればみかんの値段を表示しよう。**ただしif文は使わずに!**

# どんなときにTableを使う？

```
murao = {  
  toshi = 20,  
  daigaku = "hiroshima-daigaku",  
  sukinamono = "udon"  
}
```

年齢

通っている大学

好きなもの

自分のプロフィールなどをTable型を使って  
あらわすことが多い

# アイテムを識別しよう



今日やることは「目の前にあるのは土ブロックかどうか」のようにアイテムを識別すること



# アイテムを識別する命令

```
blk, tbl = turtle.inspect()
```

**変数を2つ使うところに注意!!**

`turtle.inspect`命令を使うことで、  
目の前のアイテムを識別できる

# 2つの変数

`turtle.inspect`命令は特別な命令で命令を実行するときに2つの変数が必要になる。  
2つの変数には下の2つが入る。

- 前の変数  
ブロックがあるかどうかの判定(trueかfalse)
- 後ろの変数  
ブロックが何なのかが入ったTable型のデータ

# Table型の変数の中身

後ろの変数は下のようなTable型になっていて  
["name"]でそのブロックがなにかを調べられる

```
{  
  state = {  
    variant = "dirt",  
    snowy = false,  
  },  
  name = "minecraft:dirt",  
  metadata = 0,  
}
```

ブロックの状態

雪が積もっているか

ブロックの種類

メタデータ

## 課題4 ブロックの種類を表示しよう

```
> 18-4  
minecraft:dirt  
>
```

**minecraft:dirtは  
土ブロックという意味**

**turtle.inspect**命令を使って目の前のブロックの  
種類を表示してみよう

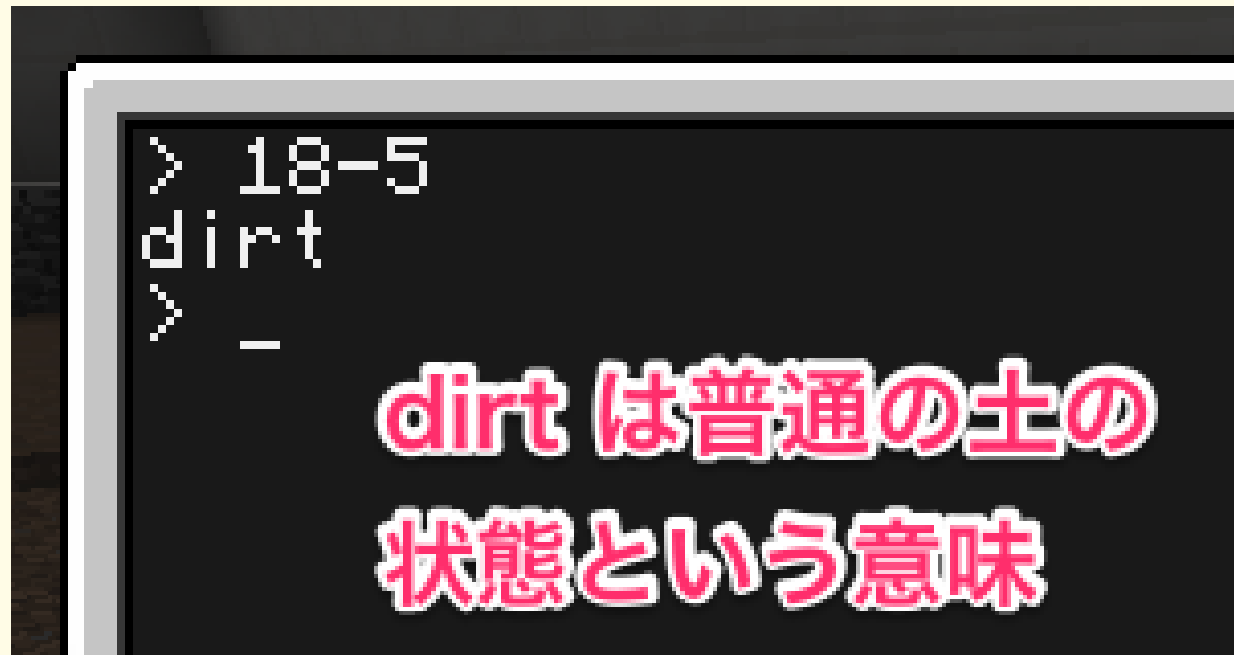
# テーブルの中にテーブルを入れる

```
{  
  state = {  
    variant = "dirt",  
    snowy = false,  
  },  
  name = "minecraft:dirt",  
  metadata = 0,  
}
```

ブロックの状態  
雪が積もっているか  
ブロックの種類  
メタデータ

テーブルの中にテーブルを入れることもできる  
例えば上の画像のstateなど

# 課題5 ブロックの状態を表示しよう



```
> 18-5  
dirt  
> _
```

**dirt は普通の土の  
状態という意味**

`turtle.inspect`命令を使って目の前のブロックの  
状態を表示してみよう

# 課題5ができれば



粗い土をクラフトして課題4と課題5の  
プログラムで調べてみよう。どうなるかな?