

Pythonの教科書

by 大家 慧士(20入)

これが章

①はじめに

1.1 進め方

- この資料は教科書のようなものだと考えてください。
- プログラミングの勉強はなるべく **目標をもって**、絶対に **手を動かして**進めてください。
- 章ごとに読み進めてください、章を読んだらその章で得た知識で **なにかコードを書いて**みてください。
- わからないことがあったら、
「**ネットで調べる→先輩に聞く**」
って感じで進めてください。
- この資料を見て、改善してほしいところや文句がありましたら、大家に言ってください。

1 はじめに

1.2 Pythonとは

Pythonとはプログラミング言語のひとつです。

Pythonの特徴には

- 計算が得意(AI, 最適化などにも使われている)
- 書くのが簡単(簡単すぎてPythonを勉強してからCを勉強しようとするとうと拒絶反応が出るので注意)
- ライブラリ(便利な道具)が充実している
- 情報がネットにたくさん載っている
- 大家が使っている

などがあります。

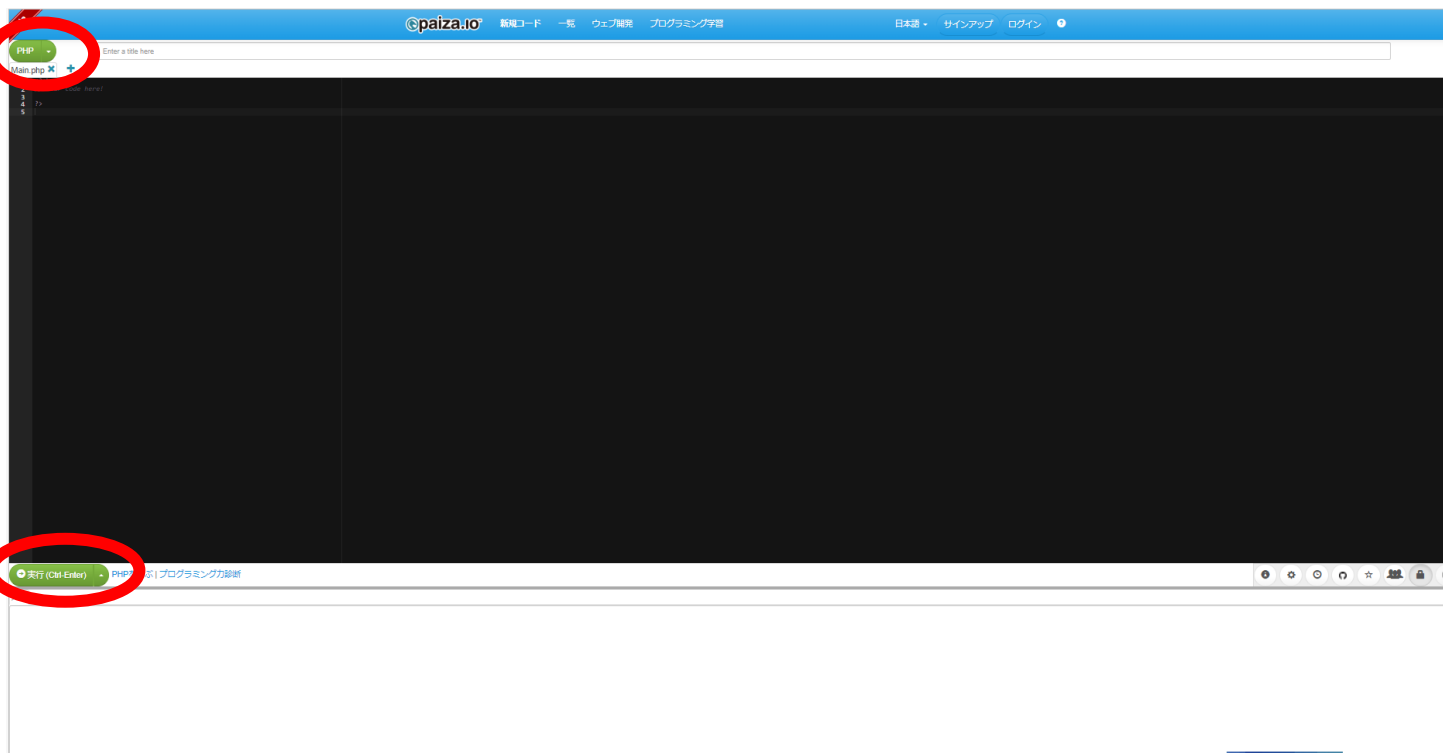
2 環境準備

2.1 とりあえず動かしたい人

↓のサイトを開いてください

[Online PHP Editor | ブラウザでプログラミング・実行ができる「オンライン実行環境」 | paiza.IO](https://paiza.io)

ここを**Python3**にする



ここで**実行**できる

2 環境準備

2.2 自分のPCで開きたい場合(個人的にはこっちがオススメ)

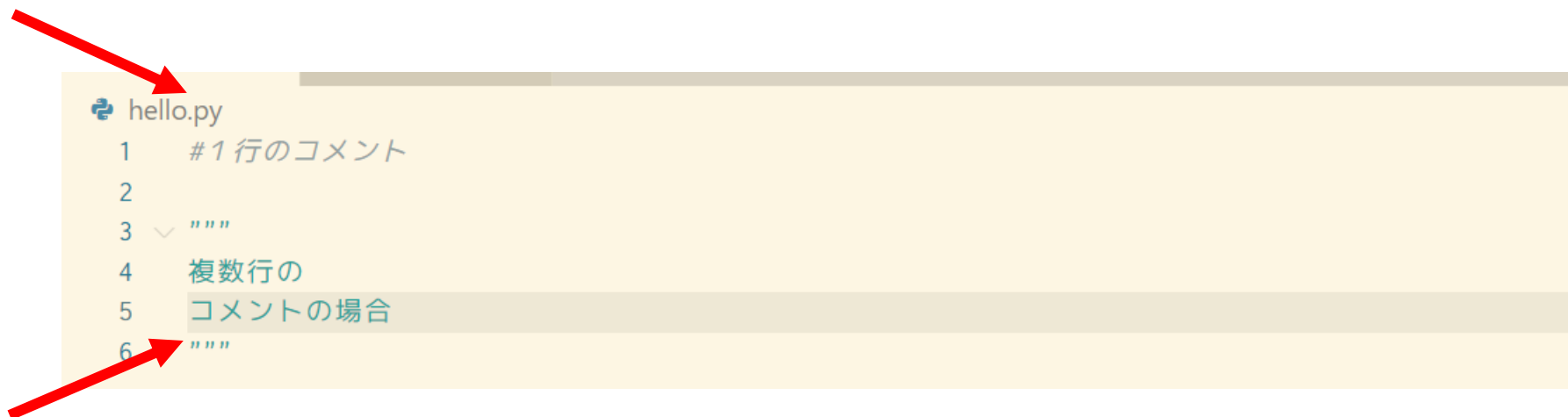
↓のサイトを参考にPythonとVSCodeをインストールして下さい

- [Pythonのインストール: Visual Studio Code でPython入門【Windows編】 - python.jp](#)
- [Visual Studio Codeのインストール: Visual Studio Code でPython入門【Windows編】 - python.jp](#)

3 コメント

コメントとはプログラミングで用いるメモのようなものです。書いても書かなくてもプログラムの挙動に変化はありません。※書くと他の人がコードを読みやすくなるので書くことをオススメします。

#だと、その行がコメントになります



""" と """ に挟まれ場所がコメントとなります

4 出力

何かを出力したいときは`print`を使います。

構文

```
print(入力したい文字)
```

ここに出力内容
が表示されてい
ます。

```
hello.py
1  #文字を表示
2  print('Hello')
3
4  #数字を表示
5  print(3)
```

ターミナル 出力 デバッグ コンソール 問題 Python + - □ □ へ ×

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved

新しいクロスプラットフォームの PowerShell をお試しください
https://aka.ms/pscore6

PS C:\MyFolder\DTK> & "C:/Program Files (x86)/Microsoft
Visual Studio/Shared/Python37_64/python.exe" c:/MyFolder
/DTK/hello.py
Hello
3
PS C:\MyFolder\DTK> □
```

5 型・変数

型の種類は3つあります。

型によって使い方が異なってくるので気を付けましょう。

```
hello.py
1  #型の種類
2
3  #文字(str)
4  'Hello'
5
6  #整数(int)
7  2
8
9  #小数点数(float)
10 2.1
```

↓ 数字(int)を文字(str)に変化させています(ほかの変換もできます)

```
hello.py
1  #数字を文字に変化
2  str(3)
```


5 型・変数

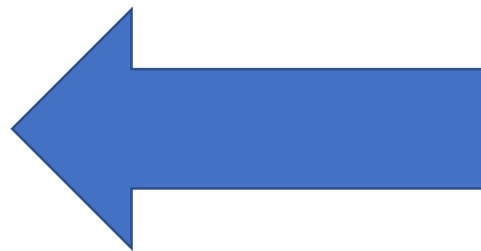
変数とは名前の付いた箱のようなものです。
箱の中に文字や数字を入れるには下のよう
にします。

構文(変数の定義)

変数名 = 入れたいもの

```
hello.py > ...  
1  #文字をxという箱の中に入れる  
2  x = 'Hello'  
3  
4  #数字をyという箱の中に入れる  
5  y = x
```

イメージ



3

6 入力

ここでは入力することができます。
入力することで出力を変化させることがで

```
hello.py > ...
1  #入力結果をxに入れる
2  x = input('ここに入力してください:')
3
4  #xの中身を表示させます
5  print(x)
```

ターミナル 出力 デバッグ コンソール 問題 Python + - □ □ □

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
.

新しいクロスプラットフォームの PowerShell をお試しください
https://aka.ms/pscore6

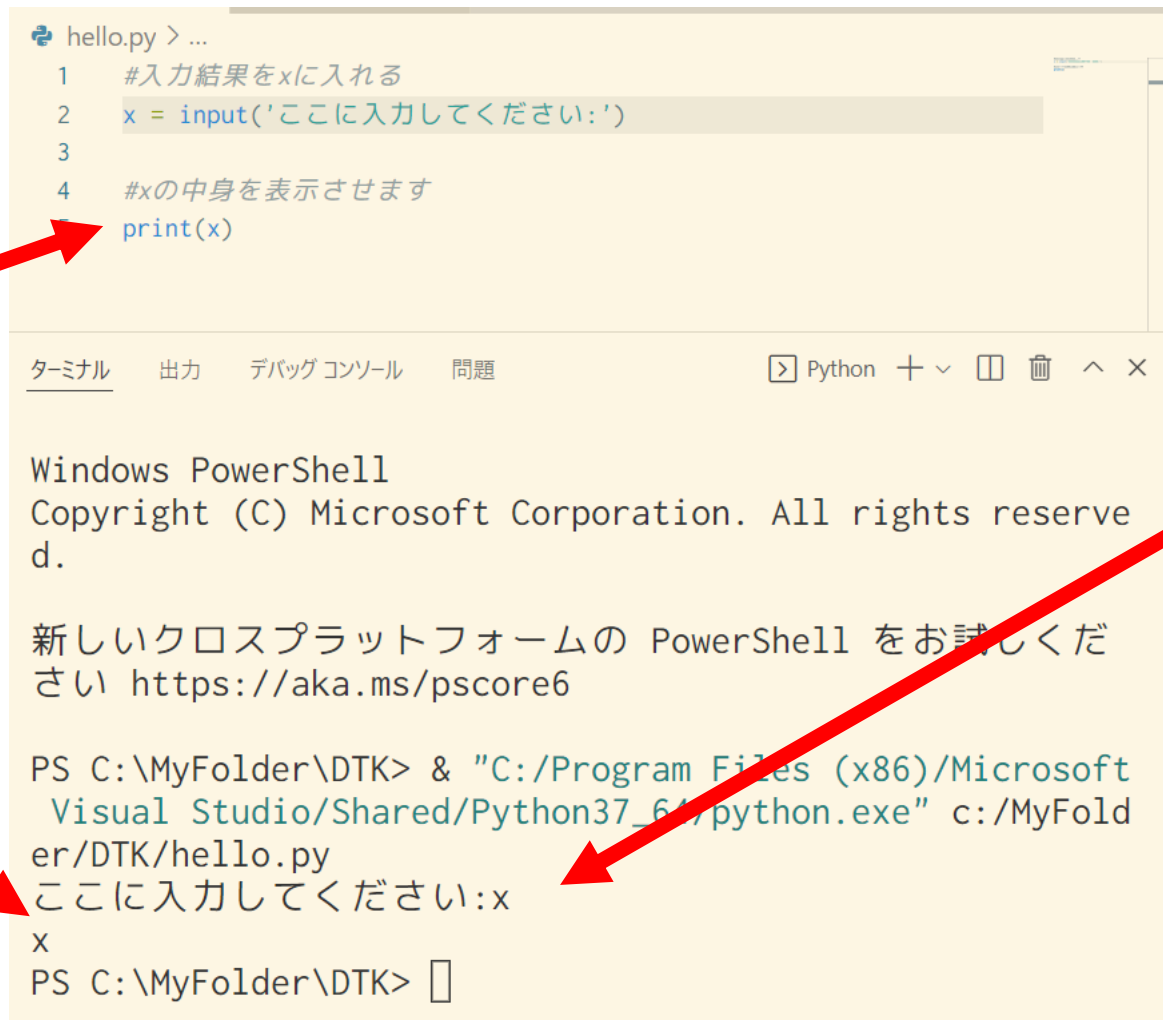
PS C:\MyFolder\DTK> & "C:/Program Files (x86)/Microsoft
Visual Studio/Shared/Python37_64/python.exe" c:/MyFolder
/DTK/hello.py
ここに入力してください:

inputを使うとターミナル
に入力する場所が現れます。

6 入力

先ほどの入力の場所に `x` を入れてみましょう。

入力結果を出力する
ようにしていたので、
`x`が出力されています。



```
hello.py > ...
1  #入力結果をxに入れる
2  x = input('ここに入力してください:')
3
4  #xの中身を表示させます
   print(x)

ターミナル  出力  デバッグ コンソール  問題  Python + - [] へ ×

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新しいクロスプラットフォームの PowerShell をお試しください https://aka.ms/pscore6

PS C:\MyFolder\DTK> & "C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/python.exe" c:/MyFolder/DTK/hello.py
ここに入力してください:x
x
PS C:\MyFolder\DTK> 
```

`x`を入力
しました。

7 四則演算

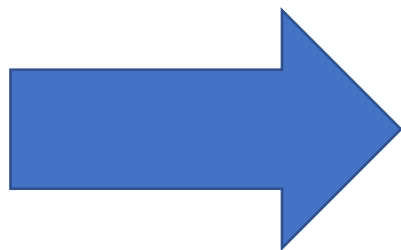
ここでは計算します、下は計算の種類です。

記号	役割	記号	役割
+	足し算	**	累乗
-	引き算	//	割り算(切り捨て)
*	掛け算	%	割り算のあまり
/	割り算		

7 四則演算

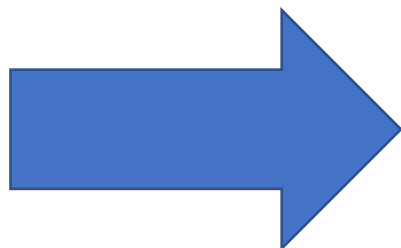
プログラムで書くと下のようになります。

コード



```
hello.py
1  #1+2を出力します
2  print(1 + 2)
3
4  #2-1を出力します
5  print(2 - 1)
6
7  #3x5を出力します
8  print(3 * 5)
9
10 #6÷2を出力します
11 print(6 / 2)
12
13 #5÷2の余りを出力します
14 print(5 % 2)
```

出力



```
3
1
15
3.0
1
```

8 条件分岐(if文)

if文とは条件式がTrueかFalseのときで実行内容が変わります。
下の構文だと「条件式がTrue(正しい)なら「条件に合ったとき」の方が実行されます。
逆に条件式がFalse(間違い)なら「条件に合わなかったとき」の方が実行されます。

構文

```
if 条件式:  
    条件式が合っているとき  
else:  
    条件式が合っていないとき
```



8 条件分岐(if文)

また、条件式には下の様なものがあります。

式	内容	式	内容
$A == B$	AとBが同じ	$A \leq B$	AはB以下
$A != B$	AとBが違う	$A \geq B$	AはB以上
$A < B$	AはBより低い		
$A > B$	AはBより高い		

8 条件分岐(if文)

プログラムで書くと下のようになります。

コード

```
C: > MyFolder > DTK > hello.py > ...
```

```
1  a = 10
2  b = 20
3
4  #AとBが同じかを判定する
5  if a == b:
6      | print('AとBは同じです')
7  else:
8      | print('AとBは違います')
```

出力

```
PS C:\Users\USER> & C:/Users/USER/anaconda3/python.exe c:/MyFolder/DTK/hello.py
AとBは違います
```


8 条件分岐(if文)

elifを使うと条件式を複数使えます。
また、**elif**は何回も使うことができます。

構文

if 条件式1:
 条件式1が合っているとき
elif 条件式2:
 条件式1が合っていないくて、
 条件式2が合っているとき
else:
 条件式がどちらも合っていないとき



9 データ構造

9.1 list型(配列)

list型はデータを複数扱うことができます。

構文(listの定義)

変数名 = [保存したいもの,保存したいもの,保存したいもの]

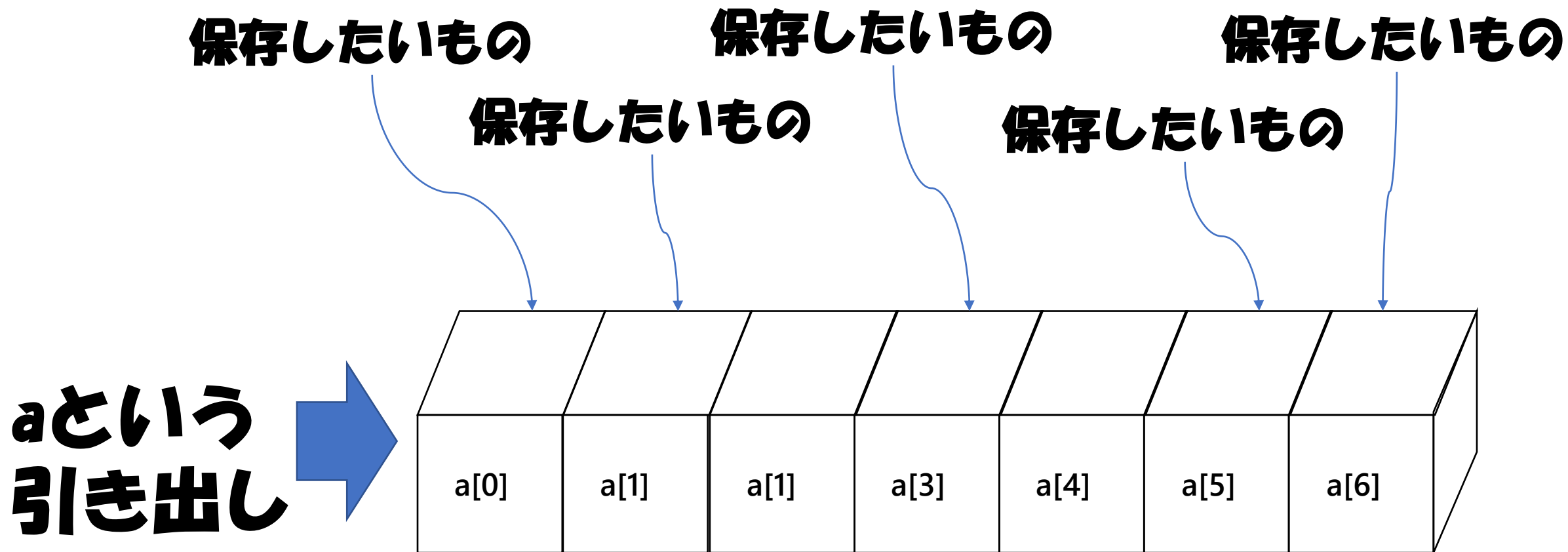
データを取り出すときは**何番目**を取り出したいかを宣言します。

構文(listからデータを取り出し)

変数名[取り出したい番号]

9 データ構造

イメージでいうとa(引き出し)のそれぞれの場所に保存したいものを入れることができる。



9 データ構造

プログラムで書くと下のようになります。

コード

C: > MyFolder > DTK > hello.py > ...

```
1  #listの定義
2  a=[2,3,4,1]
3
4  #aの0個目を表示
5  print(a[0])
6  #aの2個目を表示
7  print(a[3])
```

出力

```
2
1
```

9 データ構造

9.2 dict型

dict型はデータを辞書のように扱うことができます。

構文 (dictの定義)

変数名 = {キーA:保存したいものA, キーB:保存したいものB}

↖ 対応

↖ 対応

データを取り出すときは取り出したいものに**対応する**
キーを宣言します。

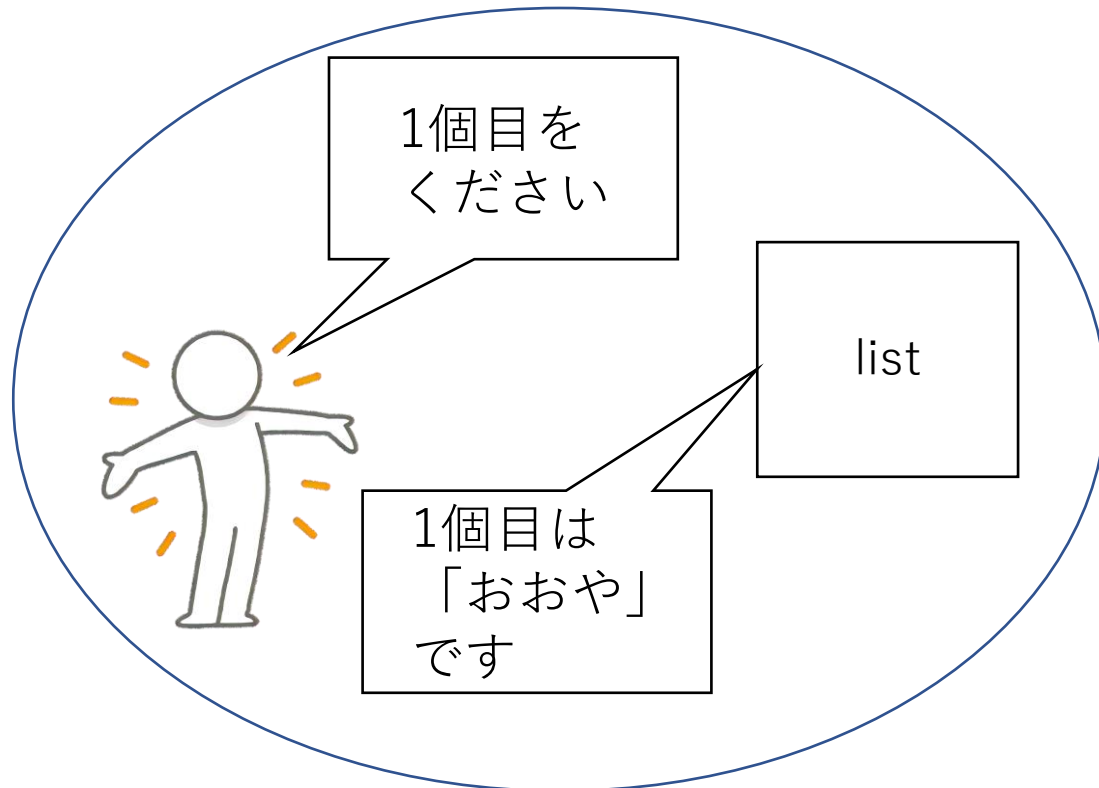
構文 (dictからデータを取り出し)

変数名[キーA]

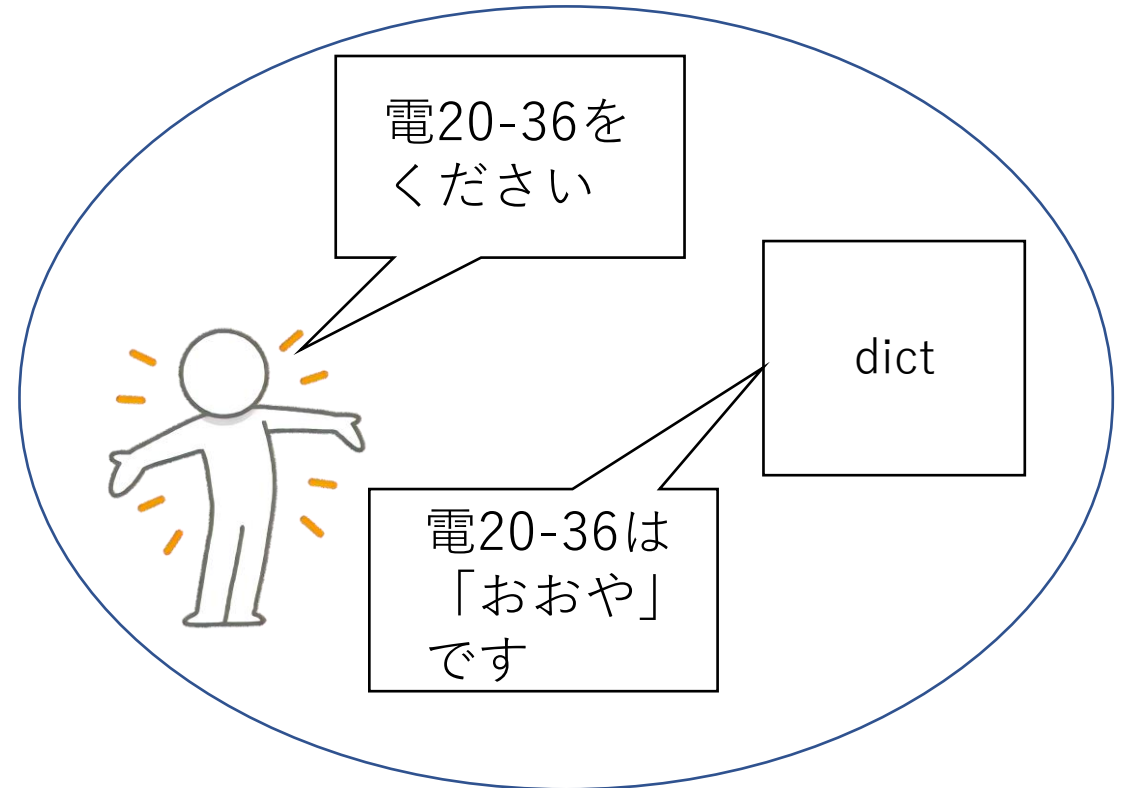
9 データ構造

イメージでいうとリストの呼び出し方をキーに変えたものです。**※リスト内でキーの同じものは一つだけです。**

list型



dict型



9 データ構造

プログラムで書くと下のようになります。

コード

```
C: > MyFolder > DTK > 📄 hello.py > ...
1  #学籍番号と生徒の名前を辞書に追加
2  a = {'電20-36': 'おおや', '電20-35': '田中'}
3
4  #学籍番号で生徒の名前を呼び出す
5  print(a['電20-35'])
6
7  print(a['電20-36'])
```

出力

```
PS C:\Users\USER> & C:/Users/USER/anaconda3/python.exe c:/MyFolder/DTK/hello.py
田中
おおや
```

9 データ構造

9.3 多次元配列

listの中に**list**を入れることができます。

構文(2次元のlistの定義)

```
変数名 = [[保存したいものA, 保存したいものB],  
          [保存したいものC, 保存したいものD]]
```

イメージ

変数名	[0]	[1]
変数名[0]	保存したいものA	保存したいものB
変数名[1]	保存したいものC	保存したいものD

9 データ構造

これは使ってみないとすこし理解が難しいです。

コード

```
C: > MyFolder > DTK > 📄 hello.py > ...
1  #多次元(今回は二次元)配列を定義している
2  a = [['A', 'B'],
3       | | ['C', 'D']]
4
5  #一つ目のリストを呼び出す
6  print(a[0])
7  #Cを呼び出す
8  print(a[1][0])
```

出力

```
PS C:\Users\USER> & C:/Users/USER/anaconda3/python.exe c:/MyFolder/DTK/hello.py
['A', 'B']
C
```

10 ループ

10.1 for文

for文はリストの中身を一つずつ取り出して変数に入れます。
取り出すごとにループの中を実行します。

構文

```
for 変数名 in リスト:  
    ループ内で実行したいこと
```

forを使うときにはよく **range**を使うことがあります。
rangeを使うと特定の回数、ループを実行できます。

10 ループ

プログラムで書くと下のようになります。

コード

```
C: > MyFolder > DTK > hello.py > ...
1  #リストを作成
2  a = ['A', 'B', 'C']
3  #リストの中身をそれぞれiに入れる
4  for i in a:
5      #iを表示
6      print(i)
7
8  #rangeを使った場合(今回は3回ループする)
9  for i in range(3):
10     #Helloを表示
11     print('Hello')
```

出力

```
PS C:\Users\USER> & C:/Users/USER/anaconda3/python
A
B
C
Hello
Hello
Hello
```

`range`の中が3なので`print`が3回実行されます。

10 ループ

10.2 while文

while文は()の中に条件式を入れます。それが成り立っているときだけループが繰り返されます。

構文

```
while(条件式):  
    ループ内で実行したいこと
```

条件式の場所に**True**を入れると無限にループが実行されます。

10 ループ

構文

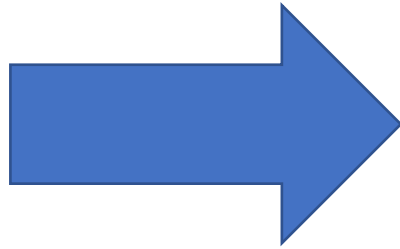
```
while(Ture):  
    ループ内で実行したいこと
```

このままでは無限にループしてしまいます。
そこでループ文内で使うとそのループを強制的に終了
ることができる、`break`を使います。
これは`for`でも`while`でもどちらでも使うことができます。

10 ループ

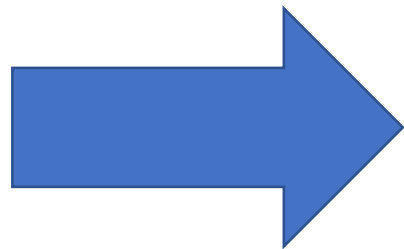
プログラムで書くと下のようになります。

コード



```
C: > MyFolder > DTK > hello.py > ...  
1   i=0  
2   #iが2以下ならループされます  
3   while(i <= 2):  
4       #iに1足す  
5       i +=1  
6  
7   print(i)  
8  
9   #無限ループ  
10  while(True):  
11      #iが5以上でbreakを実行  
12      if i >= 5:  
13          break  
14  
15      i += 1  
16  
17  print(i)
```

出力



```
PS C:\Users\USER> & C:/Users/USER/anaconda3/pyt  
3  
5
```

11 関数

関数とは特定の処理をしてくれる機能のことです。

構文(関数の定義)

```
def 変数名(引数1,引数2):  
    関数内でやりたい処理  
    return 返り値
```

引数:関数に与える情報

返り値:関数を呼び出したときに返ってくる値

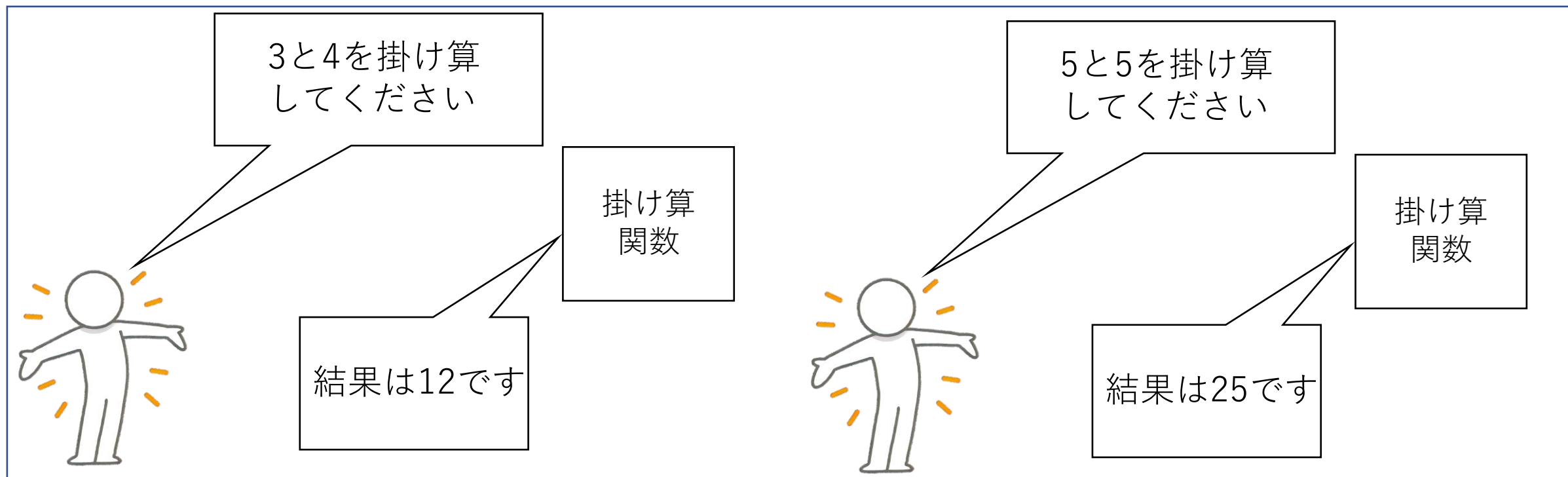
構文(関数の呼び出し)

```
変数名(引数1,引数2)
```

11 関数

例えば2つの値を与えるとその2つの値の積が返ってくる掛け算の関数を実装したとします。するとその関数を呼び出すたびに掛け算をしてくれます。

イメージ



11 関数

プログラムで書くと下のようになります。(コードについては次のページに詳しく書いています。)

コード

```
C: > MyFolder > DTK > hello.py > a
1  #関数の定義
2  def a (x,y):
3      #xとyをかけて、結果をzに入れる
4      z = x * y
5      #zを返す
6      return z
7
8  #関数を呼び出して表示1
9  print(a(3,4))
10 #関数を呼び出して表示2
11 print(a(5,5))
```

出力

```
PS C:\Users\USER> & C:/Users/USER/anaconda3/python.exe c:/MyFolder/DTK/hello.py
12
25
```

11 関数

コードをよく見てみましょう。
書いてある番号は処理の順番です。

```
C: > MyFolder > DTK > hello.py > a
1  #関数の定義
2  def a (x,y):
3      #xとyをかけて、結果をzに入れる
4      z = x * y
5      #zを返す
6      return z
7
8  #関数を呼び出して表示1
9  print(a(3,4))
```

1 呼び出す前に関数を定義しましょう

3 引数のx(3)とy(4)をかけてzに入れて、
returnでzを呼び出された場所に返す。

4 返ってきた値を表示

2 関数を呼び出して引数のxに3、yに4を入れる。

EX 読み終わった方へ

お疲れさまでした。あなたはPythonの基礎を習得しました。あとは下のコードを実行してください。

```
a = input('あなたの今の状況:')  
  
if a == 'Pythonでやりたいことがある':  
    やりたいことについて調べる  
  
elif a == 'Python使い方が不安':  
    atcodaをPythonでやってみる  
  
else:  
    先輩に聞いてみる
```