

Python

基礎構文

変数

Pythonでは、変数を作ることができる。C言語やJavaのように、int(整数型)、string(文字列)などの型指定が不要。もちろんJavaScriptやPHPのようなvarも\$も不要。

```
message = "Hello World!"  
print(message)
```

出力結果：

```
>Hello World!
```

このように変数を扱い、上のコードでは変数messageのなかのHello World!がprintでコンソール上（画面上）に出力される。

代入演算子

ここでは=が登場しているが、数学的なイコールではない。プログラミングにおいて、=は、_変数名と値を紐づける、対応付ける_という意味を持つ。

以下、代入演算子を示す。

a = b	# a に b を代入する
a += b	# a = a + b に同じ
a -= b	# a = a - b に同じ
a *= b	# a = a * b に同じ
a /= b	# a = a / b に同じ
a %= b	# a = a % b に同じ
a **= b	# a = a ** b に同じ
a //= b	# a = a // b に同じ
a &= b	# a = a & b に同じ
a = b	# a = a b に同じ
a ^= b	# a = a ^ b に同じ
a <<= b	# a = a << b に同じ
a >>= b	# a = a >> b に同じ

代数演算子

Pythonでは、単項演算子が存在しない。単項演算子とは、C言語やJavaでいう、変数iの値に1を加算するときのi++のような演算子である。これがPythonにはないので、i+=1のような表記に置き換える必要がある。

簡単な計算を試みる。

```
plus = 2 + 3
print(plus)
```

出力結果:

>5

このような時に使う足し算や、引き算などの演算子を代数演算子と言う。

以下、代数演算子を示す。

+a	# 正数
-a	# 負数
a + b	# 加算
a - b	# 減算
a * b	# 乗算
a / b	# 除算
a % b	# a を b で割った余り
a ** b	# a の b 乗
a // b	# 切り捨て除算

その他にもビット演算などもある。

~a	# ビット反転
a & b	# AND: 論理積 (aもbも1のビットが1)
a b	# OR: 論理和 (aまたはbが1のビットが1)
a ^ b	# XOR: 排他的論理和 (aまたはbが1のビットが1)
a << b	# b ビット左シフト
a >> b	# b ビット右シフト

小数を交えた計算は、

```
plus = 4 + 3.0
print(plus)
```

出力結果:

>7.0

小数が含まれると、答えも小数になる。

変数では数字だけでなく文字列も扱うことが可能である。

文字列

変数に代入することが可能。代入については、変数 = '文字列' や、変数 = "文字列" のように、'シングルクォーテーション' か "ダブルクォーテーション" で囲む。

また、複数行にわたる文字列を定義したい場合、文字列の中に改行を表す文字を入力する必要がある。このような特殊な文字を表すのに使用するのがエスケープシーケンスという。`\`バックスラッシュと文字の組み合わせでエスケープシーケンスを表す。

以下、エスケープシーケンスを示す。

```
a = "Hello \nWorld!!"
```

出力結果:

```
>Hello
>World!
```

まだ書き込む

文字列<-->数値

文字列と数値を変えたい時は以下のように記述する。

```
string = "123"
print(string)
print(int(string))
```

出力結果:

```
>"123"
>123
```

数値にするなら`int`、文字列にするなら`str`を使います。

リスト(配列)

似たようなデータを一つの変数で管理する変数のようなものをリストと言う。他言語では配列とも呼ばれる。

```
number_list = ["one", "two", "three"]
print(number_list)
print(number_list[0])
```

出力結果:

```
>["one", "two", "three"]
```

>one このように数字の文字列を管理する`number_list`というリストを作る。中身を取り出すには`number_list[x]`として、`x`には添字と呼ばれる数字を記述する。リスト、配列の添字は1からではなく、`0, 1, 2...`というように0から数えるので、`number_list[0]`とすると、先頭の`one`を取得する。

以下、リストが扱える便利なメソッドをいくつか示す。

```
number_list = ["one", "two", "three"]

number_list.reverse()
```

```
print(number_list)
number_list.sort()
print(number_list)
```

出力結果:

```
>["three", "two", "one"]
```

```
>["one", "three", "two"]
```

このようにソート(並び替え)や逆順が可能。

タプル

タプルも配列だが、後から変更できないという点で大きく異なる。つまり、定数の配列と言える。以下、定義の仕方を示す。

Flask

1.Flaskの基礎構文は以下の通り

2.

PyMySQL

Json

JavaScript

要素の生成

HTML

Jinja2

CSS

ボタンのデザイン

Terminal

cd

ls

touch

Flaskの起動

git

SSH接続