

◆やさしいPython◆

2020 年数理物理学研究室 村上作成

『やさしいPython』 著：高橋麻奈 をまとめてtex打ちしたもの。誤植も多いと思うが悪しからず。

目次

第 1 章	はじめの一步	4
1.1	入力方法	4
1.2	蛇足-Git-	4
第 2 章	Python の基本	5
2.1	コードの内容	5
2.1.1	新しいコードを入力する	5
2.1.2	コメントを記述する	5
2.1.3	1 文ずつ処理する	5
2.2	文字列と数値	6
2.2.1	文字列リテラル	6
2.2.2	数値リテラル	6
2.2.3	n 進数を使う	6
2.2.4	エスケープシーケンスを使う	7
第 3 章	変数と式	9
3.1	変数	9
3.1.1	変数の仕組み	9
3.1.2	変数の名前を決める	9
3.1.3	変数に値を代入する	9
3.1.4	変数を利用する	9
3.1.5	変数の値を変更する	10
3.1.6	その他の型を格納する	10
3.2	演算子の基本	10
3.2.1	式の仕組みを知る	10
3.2.2	式の値を出力する	11
3.2.3	変数を演算する	11
3.3	演算子の種類	12
3.3.1	いろいろな演算子	12
3.3.2	文字列の操作を行う演算子	12
3.3.3	代入演算子	12
3.4	演算子の優先順位	12

3.4.1	演算子の優先順位とは	12
3.4.2	同じ優先順位の演算子を使う	12
3.5	キーボードからの入力	12
3.5.1	キーボードから入力する	12
3.5.2	数値を入力させるには	12
3.5.3	数値を正しく計算する	12
第 4 章	様々な処理	13
4.1	if 文	13
4.1.1	状況に応じた処理をする	13
4.1.2	様々な状況をあらわす条件	13
4.1.3	条件を記述する	13
4.1.4	True と False を知る	13
4.1.5	比較演算子を使って条件を記述する	13
4.2	if elif else	13
4.2.1	if elif else のしくみを知る	13
第 5 章	リスト	14
第 6 章	コレクション	15
第 7 章	関数	16
第 8 章	クラス	17
第 9 章	文字列と正規表現	18
第 10 章	ファイルと例外処理	19
第 11 章	データベースとネットワーク	20
第 12 章	機械学習の基礎	21
第 13 章	機械学習の応用	22
第 14 章 ◆その他資料◆		23
14.1	変数	23
14.1.1	変数入力法	23
14.2	メソッド	23
14.2.1	メソッド入力法	24
14.2.2	メソッド集	24
14.2.3	メソッド入力例	24

第1章 はじめの一步

1.1 入力方法

Atom を利用。追加パッケージは大体以下の通り。

- atom-beautify
- atom-ide-ui
- atom-runner
- atom-terminal
- autocomplete-python
- highlight-colum
- linter-python-pep8

デフォルトで日本語を表示させようとするので文字化けするので、左上の File タブ→起動スクリプト→init.coffee のファイルに
`process.env.PYTHONIOENCODING = "utf-8";`
を追加して保存→再起動させると、文字化けが無くなる。

Atom の Atom-runner パッケージでは入力できないので、他のターミナルでコンパイルするほうが良い。

1.2 蛇足-Git-

\TeX 打ちをしながら本を読んでいたら、誤って上書き保存してしまい、やる気が無くなった

第2章 Pythonの基本

2.1 コードの内容

```
#画面に出力する
print("Hello World")
print("Let's start Python")
```

```
Hello World
Let's start Python
```

2.1.1 新しいコードを入力する

画面に表示させることを”出力する”ともいう。出力するには、`print(...)`というコードを記述する。

2.1.2 コメントを記述する

プログラムの実行とは直接関係のない自分の言葉をメモできる。(コメント) チームでプログラムを作成するときや、自分で見直す際にわかりやすい。
#の後の一文はコメントとして無視される。

2.1.3 1文ずつ処理する

Python では1つの小さな処理を文 (Statement) と呼ぶ。この”文”が
(原則として) 先頭から順番に1文ずつ処理される
ことになっている。

2.2 文字列と数値

2.2.1 文字列リテラル

文字の並びを**文字列リテラル (string literal)** という。文字列は「`'`」または「`"`」でくくって記述する（どちらでも良い）

Python では、次のように3つの「`'`」、`"`」でくくると、改行を含めた文字列を作ることができる。

```
#画面に出力する
print("""Hello World""")
print("'Let's start Python'')
```

Hello world! Let's start Python!

2.2.2 数値リテラル

数値には次のような種類がある。

- **整数リテラル (integer)** 1、3、100 など
- **浮動小数点数リテラル (floating)** 2.1、3.14、5.0 など
- **虚数リテラル (imaginary)** 数字に `j` をつけたもの

数字のリテラルは「`"`」や「`'`」でくくらないことに注意

2.2.3 n進数を使う

プログラムの世界では、2進数、8進数、16進数がよく使われる。Python でこれらの数値を表す場合には、数値の先頭に次の表記を付けて表す。

- **2進数** 数値の先頭に `0b` を付ける
- **8進数** 数値の先頭に `0o` を付ける
- **16進数** 数値の先頭に `0x` を付ける

#n 進法

```
print("10 進数の 10 は 10 進数でいうと", 10, "です")
print("2 進数の 10 は 10 進数でいうと", 0b10, "です")
print("8 進数の 10 は 10 進数でいうと", 0o10, "です")
print("16 進数の 10 は 10 進数でいうと", 0x10, "です")
print("16 進数の F は 10 進数でいうと", 0xF, "です")
```

10 進数の 10 は 10 進数でいうと 10 です
 2 進数の 10 は 10 進数でいうと 2 です
 8 進数の 10 は 10 進数でいうと 8 です
 16 進数の 10 は 10 進数でいうと 16 です
 16 進数の F は 10 進数でいうと 15 です

2.2.4 エスケープシーケンスを使う

文字の中には、1 文字で表せない特殊な文字がある。このような文字を `print()` によって出力する際には、`\` との組み合わせで 1 文字を表す。これを **エスケープシーケンス (escape sequence)** という。

エスケープシーケンス	意味している文字
<code>\t</code>	水平タブ
<code>\v</code>	垂直タブ
<code>\n</code>	改行
<code>\r</code>	復帰
<code>\a</code>	警戒音
<code>\b</code>	バックスペース
<code>\f</code>	改ページ
<code>\'</code>	'
<code>\"</code>	"
<code>\\</code>	\
<code>\○○○</code>	8 進数○○○の文字コードを持つ数字
<code>\xhh</code>	16 進数 hh の文字コードを持つ文字

また、文字列の前に `r` または `R` をつけると、`\` をエスケープシーケンスとして扱わない文字列を作成することができる。（**raw 文字列**）

```
#raw 文字列
print("バックスラッシュを表示：\\")
print(r"バックスラッシュを表示：\\")
```

バックスラッシュを表示：
バックスラッシュを表示：\\

第3章 変数と式

3.1 変数

3.1.1 変数の仕組み

変数：コンピューター内部にあるいろいろな値を記録しておくための**メモリ (memory)** を利用して、値を記録する仕組み。変数というハコの中に値を入れるように、特定の値を入れることができる。

3.1.2 変数の名前を決める

- 英字・数字・アンダースコア (`_`) のいずれかを用いる。
- 数字では始められない
- 大文字と小文字は区別する
- コード上意味を持つ言葉を使うことができない

このような規則に従う名前を、**識別子 (identifier)** と呼ぶ。

3.1.3 変数に値を代入する

変数に特定の値を記憶させることを、**値を代入する**という。「`=`」という記号は、値を記憶させる機能を持っている。Python の変数は、初めて値を代入したときにメモリ上に作成される。

3.1.4 変数を利用する

変数を `print` すると、変数の中に入っている値が出力される。

```
#変数への代入と利用
#変数名 = 式
sale = 10
print("売り上げは", sale, "万円です。")
```

売り上げは 10 万円です。

3.1.5 変数の値を変更する

変数の値は上書きすることができる。

3.1.6 その他の型を格納する

変数には数字の他、様々な値を格納することができる。

型

変数に格納できる値の種類を**型 (type)**と呼ぶ。Python の主な型には、下表のようなものがある。

数値	整数 (int) 真偽値 (bool) 浮動小数点数 (float) 複素数 (complex)
シーケンス	リスト (list) タプル (tuple) 文字列 (str) バイト列 (bytes)
セット	セット (set)
マッピング	ディクショナリ (dict)
クラス	
インスタンス	
例外	

3.2 演算子の基本

3.2.1 式の仕組みを知る

Python の「式」の多くは、

- 演算子 (演算する物:operator)
- オペランド (演算の対象:operand)

を組み合わせで作られる。例えば、「1+2」の場合は、「+」が演算子、「1」と「2」がオペランドにあたる。これが「評価」されることによって「式の値」が得られる。

3.2.2 式の値を出力する

簡単な計算を試みよう。

```
#式の値を出力する print("1+2 は", 1+2, "です。")
```

1+2 は 3 です。

3.2.3 変数を演算する

もう少し複雑な計算を試みよう。

```
#式の値を出力する
price = 50
num = 10
total = price * num
print("単価は", price, "円です。")
print("売上個数は", num, "個です。")
print("合計金額は", total, "円です。")
total = total - 100
print("値引きすると", total, "円です。")
```

単価は 50 円です。
売上個数は 10 個です。
合計金額は 500 円です。
値引きすると 400 円です。

3.3 演算子の種類

主な演算子の種類を次の表に示す。

+	加算	=	代入
-	減算	>	より大きい
*	乗算	>=	以上
**	べき乗	<	未満
/	除算	<=	以下
//	切捨除算	==	等価
@	行列演算	!=	非等価
%	剰余	and	論理積
+	単項+	or	論理和
-	単項-	not	論理否定
~	補数 (ビット否定)	if else	条件
	ビット論理和	in	帰属検査
&	ビット論理積	not in	非帰属検査
あ	ビット排他的論理和	is	同一性検査
<<	左シフト	is not	非同一性検査
>>	右シフト	lambda	ラムダ

3.3.1 いろいろな演算子

3.3.2 文字列の操作を行う演算子

3.3.3 代入演算子

3.4 演算子の優先順位

3.4.1 演算子の優先順位とは

3.4.2 同じ優先順位の演算子を使う

3.5 キーボードからの入力

3.5.1 キーボードから入力する

3.5.2 数値を入力させるには

3.5.3 数値を正しく計算する

第4章 様々な処理

4.1 if文

- 4.1.1 状況に応じた処理をする
 - 4.1.2 様々な状況をあらわす条件
 - 4.1.3 条件を記述する
 - 4.1.4 True と False を知る
 - 4.1.5 比較演算子を使って条件を記述する
- if文のしくみを知る

4.2 if elif else

- 4.2.1 if elif else のしくみを知る

第5章 リスト

第6章 コレクション

第7章 関数

第8章 クラス

第9章 文字列と正規表現

第10章 ファイルと例外処理

第11章 データベースとネットワーク

第12章 機械学習の基礎

第13章 機械学習の応用

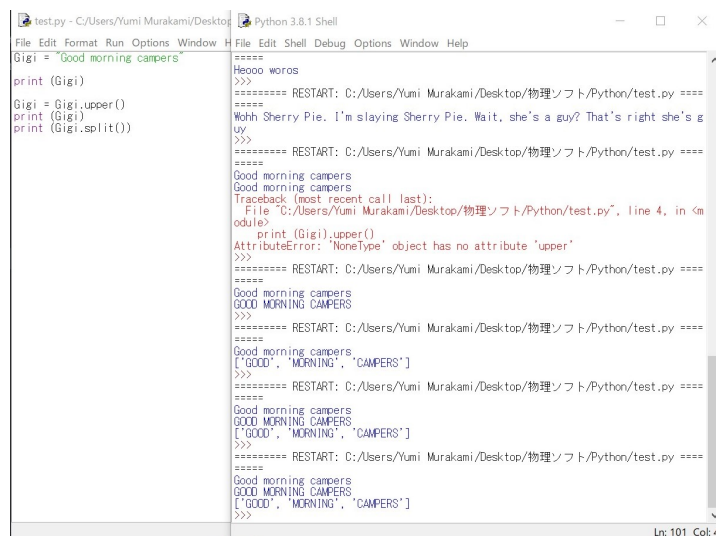
第14章 ◆その他資料◆

14.1 変数

x = "Hello world"

: x という入れ物に Hello world という文字列を保管する
変数の間にスペースは入れられない
数値をはじめにつけられない

14.1.1 変数入力法



```
test.py - C:/Users/Yumi Murakami/Desktop Python 3.8.1 Shell
File Edit Format Run Options Window Help File Edit Shell Debug Options Window Help
Gigi = "Good morning campers"
print (Gigi)
Gigi = Gigi.upper()
print (Gigi)
print (Gigi.split())

====
Heooo woros
>>>
===== RESTART: C:/Users/Yumi Murakami/Desktop/物理ソフト/Python/test.py ====
====
Wohh Sherry Pie. I'm slaying Sherry Pie. Wait, she's a guy? That's right she's a
uy
>>>
===== RESTART: C:/Users/Yumi Murakami/Desktop/物理ソフト/Python/test.py ====
====
Good morning campers
Good morning campers
Traceback (most recent call last):
  File "C:/Users/Yumi Murakami/Desktop/物理ソフト/Python/test.py", line 4, in <mo
odule>
    print (Gigi).upper()
AttributeError: 'NoneType' object has no attribute 'upper'
>>>
===== RESTART: C:/Users/Yumi Murakami/Desktop/物理ソフト/Python/test.py ====
====
Good morning campers
GOOD MORNING CAMPERS
>>>
===== RESTART: C:/Users/Yumi Murakami/Desktop/物理ソフト/Python/test.py ====
====
Good morning campers
["GOOD", "MORNING", "CAMPERS"]
>>>
===== RESTART: C:/Users/Yumi Murakami/Desktop/物理ソフト/Python/test.py ====
Good morning campers
GOOD MORNING CAMPERS
["GOOD", "MORNING", "CAMPERS"]
>>>
===== RESTART: C:/Users/Yumi Murakami/Desktop/物理ソフト/Python/test.py ====
Good morning campers
GOOD MORNING CAMPERS
["GOOD", "MORNING", "CAMPERS"]
>>>
```

図 14.1: how to use variables

14.2 メソッド

メソッド：お尻につけると便利な働きをしてくれるコマンド

14.2.1 メソッド入力法

入力画面：print ("Hello world". コマンド)

→出力画面に出力

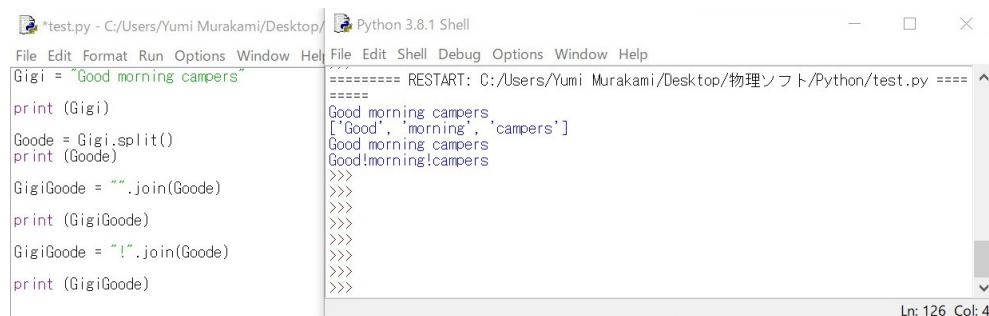
メソッドは組み合わせ可能

14.2.2 メソッド集

split("A")	A を境に分割
replace("A","B")	A を B に置き換え
upper()	小文字を大文字に置き換え
lower()	大文字を小文字に置き換え
join(変数)	ばらばらの文字列をつなげる
find("文字列")	特定の文字列が何番目かを教える
count("文字列")	特定の文字列が登場する個数を教える

14.2.3 メソッド入力例

join



```

*test.py - C:/Users/Yumi Murakami/Desktop/Python 3.8.1 Shell
File Edit Format Run Options Window Help
Gigi = "Good morning campers"
print (Gigi)
Goode = Gigi.split()
print (Goode)
GigiGoode = " ".join(Goode)
print (GigiGoode)
GigiGoode = "!".join(Goode)
print (GigiGoode)

===== RESTART: C:/Users/Yumi Murakami/Desktop/物理ソフト/Python/test.py =====
Good morning campers
['Good', 'morning', 'campers']
Good morning campers
Good!morning!campers
>>>
>>>
>>>
>>>
>>>
>>>
>>>
Ln: 126 Col: 4

```

図 14.2: how to use join