3. Pythonプログラムの基本を知ろう

3.1. Pythonの基本

3.1.1. ステートメントとインデント

Pythonのプログラムを実行すると、記述されたコードの命令が、一つずつインタプリタで解釈、処理されていきます。その命令の最小の単位を「ステートメント（statement）」といいます。ステートメントは日本語では「文」という意味ですね。

ステートメントにはその実行したい内容に応じてさまざまな種類が用意されていて、その種類に応じて記述のルール、つまり構文が決まっています。ここでは、ステートメントの書き方についていくつかの例を用いて解説をしていきます。ステートメントの意味については、ここで理解している必要はありませんので、書き方に注目して読み進めてください。

例えば、「Hello Python!」と出力表示するステートメントは、sample03_01.pyのように記述します。このように一行で記述できるステートメントを「単純文」といいます[1]。

sample03_01.py 単純文

```
---------- コード ----------
print('Hello Python!')
-------- ここまで --------
-------- 実行結果 --------
Hello Python!
-------- ここまで --------
```

単純文で、sample03_02.pyのようにかっこ「()」「[]」「{}」で囲まれた範囲であれば、改行をして記述することができます[2]。

sample03_02.py 単純文の改行

```
---------- コード ----------
numbers = [
    1, 2,
    3, 4
```

[1] 単純文は、基本的に一行に一文を記述しますが、複数の単純文をセミコロン(;)で区切って一行に記述することができます。

[2] または、バックスラッシュ「¥」を入れた位置にも改行を挿入することができます。

```
]
print(numbers)
-------- ここまで --------
-------- 実行結果 --------
[1, 2, 3, 4]
-------- ここまで --------
```

この例ではあまり効果的ではありませんが、ステートメントが長くなりすぎる場合などに、改行を入れて「縦に揃える」ことを意識すると読みやすいコードになります。

また、ステートメントとステートメントの間の空行については、自由に入れることができます。sample03_03.pyのようにステートメントの間に複数の空行を入れても問題なく実行をすることができます。

sample03_03.py 空行

```
---------- コード ----------
print('Hello Python!')

print('Hello World!')
-------- ここまで --------
-------- 実行結果 --------
Hello Python!
Hello World!
-------- ここまで --------
```

適宜空行を入れることで、処理のまとまりを作ったり、見やすくしたりすることができます。うまく活用するとよいでしょう。

一方で、一部のステートメントは、他のステートメントを含む形で複数行で構成します。これを「複合文」といいます。sample03_04.pyの「if」から始まる行から最後までは、ひとつのステートメントで複合文です。

sample03_04.py 複合文

```
---------- コード ----------
x=10
y=5
```

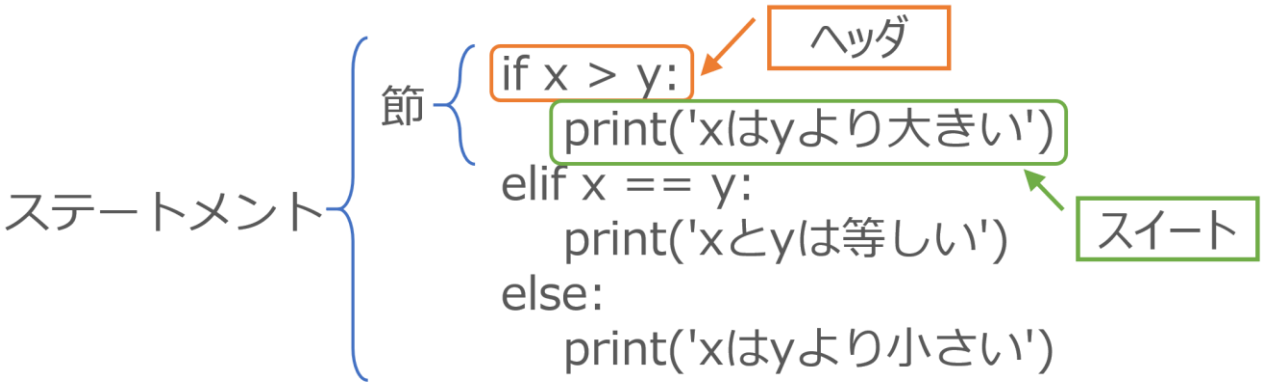
```
if x > y:
    print('xはyより大きい')
elif x == y:
    print('xとyは等しい')
else:
    print('xはyより小さい')
-------- ここまで --------
-------- 実行結果 --------
xはyより大きい
-------- ここまで --------
```

複合文は一つ以上の「節（clause）」で構成されます。sample03_04.pyであればif節、elif節、else節の3つの節で構成されています。

それぞれの節は「ヘッダ（header）」と「スイート（suite）」で構成されます。ヘッダは節の先頭の行で、コロン(:)で終わります。スイートは節に含まれる他のステートメントの集まりです。日本語では「組」「ひと揃い」などの意味がありますね。

図3-1 複合文のつくり

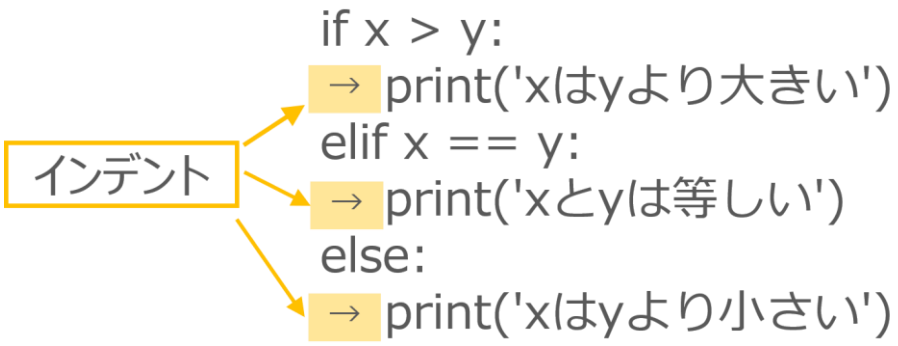
(fig03-01.pngはいる)



Pythonでは、スイートはヘッダから一段字下げをして記述するルールとなっています。この字下げを「インデント（indent）」といいます。

図3-2 インデント

(fig03-02.pngはいる)



インデントには［Tab］キーを、インデントを戻すには［Shift］+［Tab］を使います。VS Codeでは複数行を選択すれば、まとめてインデントをしたり、戻したりすることできます。

Pythonではインデントは重要な役割を果たします。例えば、sample03_05.pyのように、正しいインデントをしていないコードを実行してみましょう。

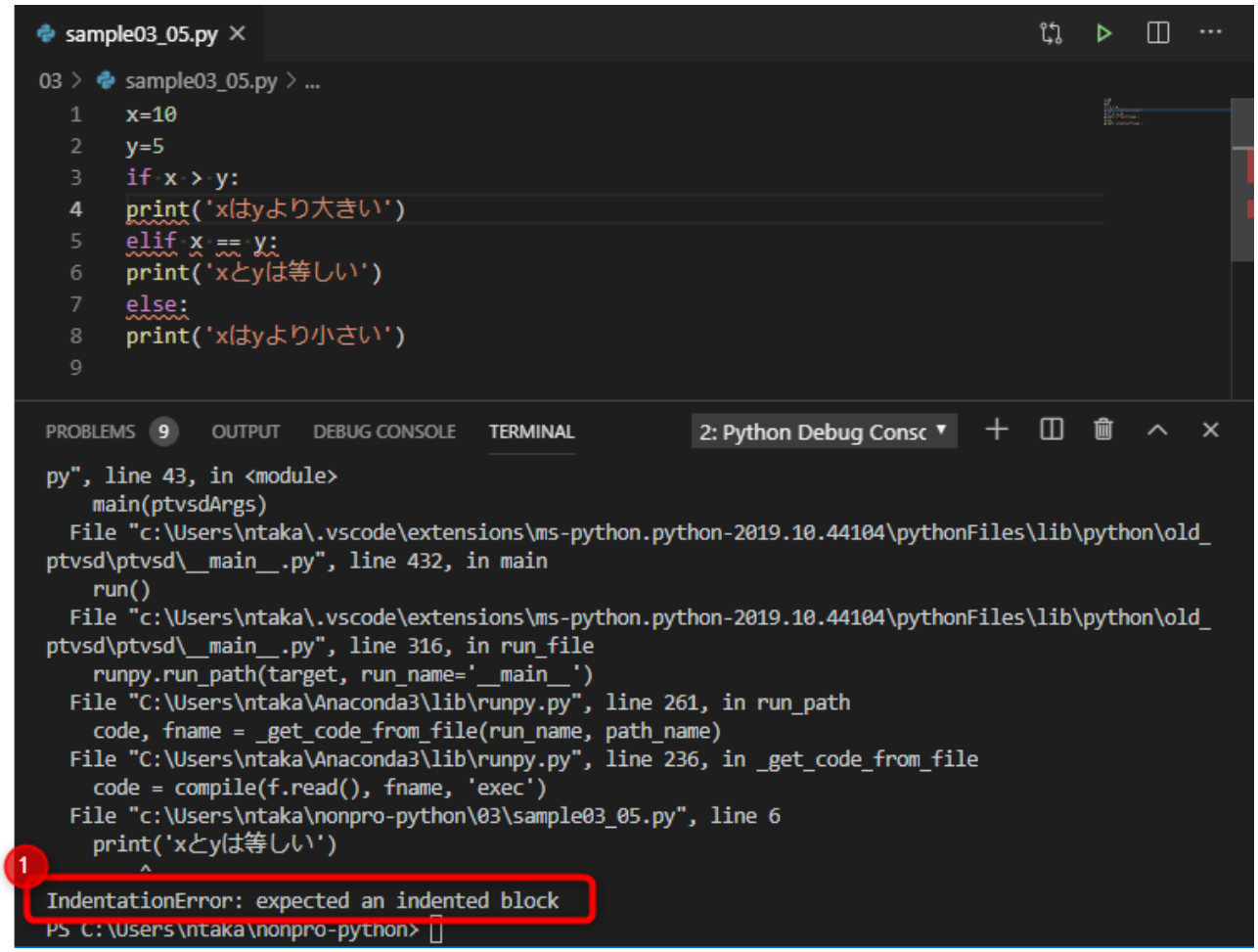
sample03_05.py

```
---------- コード ----------
x=10
y=5
if x > y:
print('xはyより大きい')
elif x == y:
print('xとyは等しい')
else:
print('xはyより小さい')
-------- ここまで --------
```

図3-3にも示している通り、ターミナルの出力表示に以下のような一文を見つけることができるはずです。

```
-------- 実行結果 --------
IndentationError: expected an indented block
-------- ここまで --------
```

図3-3 エラーメッセージ「IndentationError」
(fig03-03.pngはいる)



Cap1 エラーメッセージとその内容

これは、エラーメッセージで、プログラムが正しく動作しなかったこと伝えるものです。「～Error」と表現されているほうがエラーの種類を表し、他方がその内容を表します。これを確認することで、どのようなエラーが発生したかを判定することができます。

この例では「IndentationError」つまりインデントのエラーであり、メッセージは「インデントされたブロックが必要です」という意味ですね。

このように、Pythonではインデントのルールが厳密に定められています。これにより、そのコードは誰が書いても同じようなフォルムになり、読みやすいコードになるのです。