Python 入門

プログラミングの基礎から応用まで

教師用指導資料

部分サンプル（１章）

東京書籍

はじめに

高校生は日常的にさまざまなアプリを利用している一方で，それらの仕組みに対する意識は希薄です。しかし，プログラミングの学習を進めていくと，仕組みの一部を理解できるようになったり，自分でプログラムを作ろうと考えるようになったりして，生徒の意識が変わっていくことに気づくでしょう。

ただし，プログラミングの学習には，言語の仕様やプログラミング特有の概念の理解，論理的な思考など，これまであまり求められなかった力を必要とするため，生徒にとって「難しい」ものです。

本資料では，これまで筆者らがプログラミングを指導してきた経験から，授業を円滑に進めていくためのノウハウをお伝えします。指導の参考になれば幸いです。

プログラミングの指導に際して

興味を持たせる

プログラミングを学ぶ目的の 1 つは，日常的に利用している情報技術の仕組みを理解することです。数行程度の簡単なゲームプログラムを作成しただけで，普段遊んでいるゲームの仕組みに言及する生徒もいます。「そういうことだったのか」「こうやって作られていたのか」と感じさせる題材で，プログラミングに興味を持たせながら授業を進めるとよいでしょう。

達成感を持たせる

授業では「自分でできた」という経験を多く積ませましょう。そのためには，実行結果を見せるなどして，ヒントを与えながらプログラムを考えさせる方法や，ソースコードの重要な箇所を空欄にしておいて，その部分を考えさせる方法なども有効です。

文字入力の練習をしておく

高校生の多くはキーボードによる文字入力を苦手としています。しかし，入力に手間取っているようでは，プログラミングの本質的な部分を学ぶことは困難です。そのため，プログラミングの授業を行う前には，文字入力の方法の確認やタイピングの練習をしておきましょう。プログラムは 1 文字でも間違えると実行できなかったり，正しく動作しなかったりします。0（ゼロ）と O（オー），l（エル）と i（アイ）など，紛らわしい文字について注意を促したり，：（コロン）やタブなどのキーの位置を確認したりすることが必要です。

ショートカットキーの活用

プログラミングでは，同じ命令を何度も使ったり，特定の文字列を探したりすることがたびたびあります。そのような作業を効率よく行うために，ショートカットキーを活用するように指導するとよいでしょう。特に，下記の操作は役に立つ場面が多く，事前に説明しておくことをおすすめします。

- 範囲選択（Shift+ 矢印）
- コピー（Ctrl+C）
- 貼り付け（Ctrl+V）
- 検索（Ctrl+F）

エラーを出す

「プログラムは，正しく記述されていないと動作しない」ということを体験的に理解させましょう。そのためには，あえてエラーを出させて，その原因を考えてもらうことが有効です。例えば，次のようなプログラムをあえて実行させ，エラーの原因を取り除くことで正しく動作することを体験させるとよいでしょう。

- `plint('abc')` → `print('abc')`
- `for i in range(3)` → `for i in range(3):`
- `if a = 3:` → `if a == 3:`

各章の指導のポイント

第 1 章　Python の基本

第 1 章は，プログラミングおよび Python の基本となる概念や用語の学習が中心です。数行程度の簡単なプログラムを通して，以下の概念・用語を理解させることが目標となります。第 2 章以降のプログラムと比べると派手さはありませんが，この章の内容を理解していないと，以降の学習でつまずいてしまいます。生徒の理解状況を把握しながら，丁寧に進めていきましょう。

	内容	理解すべき概念・用語
1	文字を表示しよう (print 関数)	関数　引数
2	文字を覚えておける入れ物を使おう (変数)	変数　代入　予約語
3	キーボードから文字を入力しよう (input 関数)	input 関数
4	パスワードを判定しよう (if 文)	if 文　条件式　インデント
5	パスワードが正しくないときにメッセージを出そう (else 文)	else 文
6	繰り返して実行しよう (for 文)	for 文
7	for で繰り返す範囲を意識しよう (インデント)	インデント
8	数を計算しよう (算術演算子)	四則演算　算術演算子　文字列型　数値型
9	数を比べよう (比較演算子)	比較演算子
10	条件を組み合わせよう (論理演算子)	論理演算子
11	キーボードから数を入力しよう（文字列と数値の変換)	int 関数
12	偶数だけを表示しよう（for と if の組み合わせ)	

生徒の多くは，文字の誤入力によるエラーの対処に追われます。練習のため，あえてエラーを出させて，その要因を生徒自身で取り除く活動などを取り入れるとよいでしょう。また，指示された活動しか行わない生徒もいるため，変数に代入する値や条件式などを自由に変えさせて，実行結果を確認する活動を促すのもよいでしょう。

4 ページ

第1章 Pythonの基本

この章では，Python言語を使いプログラミングの基礎を学びます。

❶ ① 文字を表示しよう（print関数）

画面に文字を表示してみよう。プログラムの中で，コンピュータにしてほしい動作を伝えるときは関数を使う。関数は「関数名()」という形をしていて，( )の中には引数と呼ばれる情報を書ける。

［プログラム1-1］print関数を使って文字を表示

```
1 print('abc')
2 print('def', 'ghi')
```

実行結果

```
abc
def ghi
```

プログラム1-1のprint()は画面に文字を表示する関数で，表示する文字は，引数として( )の中に「' '」（シングルクォーテーション）で囲んで表現する。複数の文字列を表示したい場合は，引数をカンマで区切って書く。カンマで区切ると，空白を挟んで連続して表示される。

プログラム1-1のようにprint関数を続けて実行すると，print関数は文字列を表示した後に，改行して次の行に移る。

［プログラム1-2］改行を行わないprint関数

```
1 print('abc', end='')
2 print('xyz')
```

実行結果

```
abcxyz
```

まずは手を動かしてやってみることが大事です

▶関数
プログラミングにおいて，あるまとまった処理に名前を付けたもの。プログラムの中ではその名前を用いて命令を指示することにより処理が実行される。

▶引数
「ひきすう」と読む。プログラムにおける関数が外部とデータをやりとりするために用いる値のこと。関数の中で処理するデータを自由に変える役割を持つ。

MEMO
文字列は「"」（ダブルクォーテーション）で囲んで表現することもできる。文字列の中にシングルクォーテーションを使いたい場合は，ダブルクォーテーションで囲むとよい。

MEMO
プログラムは半角で書く。( )や空白は全角にしてしまうとエラーになるが，気がつかないことも多いので全角入力をしていないか注意が必要である。

4

❶

文字には全角と半角があることや，その区別ができていない生徒がいることに気をつける。
print(' パイソン ') のように，表示する文字には全角文字を用いることもできる。

5 ページ

同じ行に表示を続けたい場合は，**プログラム1-2**のようにprint関数の引数としてend=''を指定する。こうすることで，表示した後の改行を行わないようにできる。

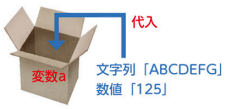
練習問題

print関数を使って，「Hello Python!」と表示するプログラムを作ろう。

② 文字を覚えておける入れ物を使おう（変数）

変数という入れ物に，文字列などを入れて覚えておくことができる（**図1-1**）。

[図1-1] 変数は文字列や数値などを入れておける箱のようなもの



[プログラム1-3] 変数を使って文字を表示

```
1 name = 'Taro'
2 print(name)
```
❶

実行結果

```
Taro
```

プログラム1-3では，nameという変数に'Taro'という文字列を代入している。一度値を入れておくと，プログラムの中でnameという名前でその値を使うことができる。

[プログラム1-4] 変数を使って2回文字を表示

```
1 name = 'Taro'
2 print(name)
3 name = 'Jiro'
4 print(name)
```
❷

▶**変数**
コンピュータが扱うメモリ上の特定の領域を識別するための名前。数値や文字列などの要素（データ）を一時的に保管しておく「入れ物」として使う。

MEMO
変数には,文字列ではなく数値を代入することもできる。例えば「1234」と書かれている場合,それが「千二百三十四」という数値なのか,「いちにいさんよん」という文字列なのかで意味が異なる。文字列を代入する場合には,シングルクォーテーションまたはダブルクォーテーションで囲む。

▶**代入**
変数に文字列や数値を格納することを「代入」と呼ぶ。また,変数に入れる文字列や数値のことを一般的に値（あたい）または要素と呼ぶ。

❶
変数名は生徒自身に考えさせて，「自分で定義できる」ことを体験させてもよい。
「name = Taro」や「name = 'Taro」などとあえて記述・実行させてエラーを出させ，正しく記述しないと動作しないことを理解させるとよい。

❷
3行目が実行されると，1行目で変数 name に代入された 'Taro' は，'Jiro' へと上書きされる。

練習問題の解答例

```
1 print('Hello Python!')
```

実行結果

```
Taro
Jiro
```

プログラム1-4では，nameという変数に'Taro'という文字列を代入し，それを表示した後で，同じ変数に'Jiro'という文字列を代入し，それを表示している。変数には何度も値を代入することができる。代入すると，変数には後から代入した値が入る。

変数名はnameのほかに，xやxy123，x_123など，英字，数字，「_」（アンダースコア）を使った名前を使うことができる。ただし，1文字目に数字を使うことはできない。また，if，forなどPython言語自身に使われている言葉（**予約語**）は使えない。

▶**予約語**
Pythonでは，以下のような言葉は関数名や変数名には利用できない。

False	None	True
and	as	assert
async	await	break
class	continue	def
del	elif	else
except	finally	for
from	global	if
import	in	is
lambda	nonlocal	not
or	pass	raise
return	try	while
with	yield	

③ キーボードから文字を入力しよう（input関数）

input関数を使うと，実行中にキーボードから文字列を入力できる。

半角英数字だけでなく，漢字を入力することもできるね

［プログラム1-5］input関数を使って文字を入力

```
1 name = input()
2 print(name)
```

実行結果（色のついた部分はキーボードからの入力）

```
Taro
Taro
```

プログラム1-5を実行すると，画面に入力欄が表示される。実行結果のようにTaroを入力すると，input関数の結果がTaroになり，その文字列が変数nameに代入される。print関数でnameを引数に指定すると，画面にTaroが表示される。

7ページ

［プログラム1-6］input関数に文字列を指定

```
1 name = input('Name? ')
2 print(name)
```

実行結果（色のついた部分はキーボードからの入力）

```
Name? Taro
Taro
```

入力する画面にメッセージを表示するときは，input関数の引数として文字列を指定する。**プログラム1-6**では，'Name? 'という文字列を指定しているため，実行すると画面にName? が表示される。入力時にメッセージを表示することで，使う人に何を入れてほしいかを伝えることができる。

「Name?」と聞かれて，「Taro」と名前を入力したら，「Taro」と表示された。コンピュータが名前を覚えてくれたみたいですね

練習問題

名前を入力したら，「名前 san, good night!」と表示するプログラムを作ろう（例えば，Taroと入力した場合は「Taro san, good night!」と出力される）。

❶ ④ パスワードを判定しよう（if文）

if文を使うと，指定した条件が成り立つときだけ処理を実行できる。if文は，次の形で記述する。ifの後に空白をあけて，条件式を「pw == in」のように指定する。条件式の右側には「:」（コロン）を書く。処理の部分には，print関数のようにさまざまなプログラムを書ける。

MEMO
アルゴリズムの制御構造のうち，条件によって処理を選択して実行するものを「分岐構造」と言う。分岐構造はこのようにif文などを使って作る。

if文の書き方

```
if 条件式:
    処理
```

［プログラム1-7］if文

```
1 a = 3
2 if␣a == 3:
3 ␣␣␣␣print('aは3です')
```

❷

半角スペースを入れないとエラーになる箇所は␣で示しています

❶
if 文はよく利用するので，「if a=3:」や「if a==3」（コロンの記述忘れ），インデントなしなどのミスが起こりやすいことを確認し，確実に記述できるようにしておくと，今後の学習がスムーズになる。

❷
条件式を「a==4」や「a!=3」などと変更した場合の実行結果を確認をさせるとよい。

練習問題の解答例

```
1 name = input()
2 print(name, 'san, good night!')
```

8 ページ

実行結果

```
aは3です
```

プログラム1-7では，変数aの値が3のときに「aは3です」を画面に表示する。条件が成り立つときに処理されるprint関数を見ると，左側に何文字かの空白があることがわかる。このように，左側に空白文字を書くことを**インデント**[1]と呼ぶ。if文は，インデントされている行を処理する範囲と判断する。インデントは4個の半角スペースを使うことが多い。処理に複数のプログラムを書く場合には，インデントの個数がそろっていないとエラーになるので注意しよう。

MEMO
Pythonでは，インデントに何個の半角スペースを用いるかが非常に重要である。インデントについては12ページも参照。

❷ **[プログラム1-8] if文を使って文字列を判定**

```
1 pw = 'Hello'
2 ipt = input('password? ')
3 if␣pw == ipt:
4 ␣␣␣␣print('login ok')
```

実行結果（色のついた部分はキーボードからの入力）

```
password? Hello
login ok
```

キーボードから入力した文字列が，正しいパスワードであることを調べるプログラムを作ってみよう。**プログラム1-8**では，はじめに正解のパスワードの文字列を変数pwに代入している。このパスワードとキーボードから入力された文字列が等しければ画面に「login ok」を表示する。キーボードからの入力は，input関数を用いて変数iptに代入する。

次にif文を使い，条件式「pw == ipt」で変数pwと変数iptの値が等しいかどうかを比較する。等しかった場合は，プログラムの4行目のインデントされているprint関数を実行して，画面に「login ok」が表示される。

❸ 「==」を使うと2つの値が等しいかどうかを調べることができ，「!=」を使うと2つの値が等しくないかどうかを調べることができる（**表1-1**）。セットで覚えておこう。

❶
インデントは Tab キーで設定することも可能である。
インデントの有無は，実行の可否や動作結果に大きく影響するため，その扱いに気をつけさせる。

❷
プログラミングの学習においては，できるだけ全ての場合の実行結果を確認させるとよい。
このプログラムでは，「login ok」と表示される場合と何も表示されない場合があるが，そのことを体験的に理解させるとよいだろう。

❸
「!=」については，「等しくないかどうか」と教えてもよいが，「==」を「!=」と書き換えた場合のプログラムを生徒に作成させ，その実行結果から「!=」の意味を考えさせることで，より理解が深まるだろう。

9ページ

［表1-1］左右の値が等しいかどうかを調べる演算子

演算子	意味
==	左右の値が等しい
!=	左右の値が等しくない

❶ **練習問題**

if文を使って，「Good morning」と入力された場合に「It's a fine day!」と表示するプログラムを作ろう。

⑤ パスワードが正しくないときにメッセージを出そう（else文）

if文では「条件式が成り立った場合」の処理を書くことができた。今度は，**else文**を使って「条件式が成り立たなかった場合」の処理を書いてみよう。if文は，次の形で書くこともできる。ifの条件式が成り立った場合は処理1が，成り立たなかった場合は処理2が実行される。

❷ if-else文の書き方

```
if 条件式:
    処理1
else:
    処理2
```

［プログラム1-9］if-else文

```
1 pw = 'Hello'
2 ipt = input('password? ')
3 if␣pw == ipt:
4 ␣␣␣␣print('login ok')
5 else:
6 ␣␣␣␣print('wrong password')
```

実行結果（色のついた部分はキーボードからの入力）

```
password? python
wrong password
```

プログラム1-9の前半の4行で「正しいパスワードと入力され

❶
この練習問題では，文中に「It's」とシングルクォーテーションがあるため，表示させるためには文全体をダブルクォーテーションで囲む必要があることを教えておくとよい。

❷
if-else 文は条件式と処理１，処理２の関係の理解が重要である。条件式や処理内容を変更したり，４行の if ブロックを，見本を見ないで入力させるなどの活動を繰り返すとよい。また，８ページと同様，２通りの結果が出ることを確認させ，そのような確認を常に自分自身で行うよう意識させることも，今後の学習を進めるうえで効果的である。

練習問題の解答例

```
1 ipt=input()
2 if ipt=='Good morning':
3     print("It's a fine day!")
```

た文字列が同じかどうか」を調べている。同じ場合には「login ok」が表示される。同じでない場合には，後半5〜6行目のelse文に書かれたprint関数が実行されて「wrong password」が表示される。

プログラムを入力し，実行して動作を確認しよう。正解のパスワードを入力して「login ok」が表示されることと，不正解のパスワードを入力して「wrong password」が表示されることの両方を確認することで，プログラムが正しく書かれていることを確認できる。

練習問題

if文を使って，「How are you?」と入力された場合は「Good!」と表示し，そうでなければ「Hello?」と表示するプログラムを作ろう。

❶ ⑥ 繰り返して実行しよう（for文）

for文を使うと，処理を繰り返して実行できる。for文は，次の形で書くことができる。変数にはi，jなど短い名前がよく使われる。回数は処理を実行する回数である。

MEMO
アルゴリズムの制御構造のうち，条件によって同じ処理を繰り返し実行するものを「反復構造」と言う。反復構造はこのようにfor文などを使って作る。また，反復構造の繰り返し行う処理をループとも呼ぶ。

for文の書き方

```
for 変数 in range(回数):
    処理
```

❷ [プログラム1-10] for文

```
1 for i in range(3):
2     print('Hello')
```

実行結果

```
Hello
Hello
Hello
```

プログラム1-10は，回数として3を指定している。実行すると，2行目のprint関数が3回実行されて，Helloが3回表示さ

10

❶
for 文は初学者には難しく，つまずきやすい概念であるため，丁寧に指導する必要がある。

❷
変数 i を他の文字に変えたり，回数を増やしたりなどさせて，変更ができる箇所とできない箇所があることを理解させる。

練習問題の解答例

```
1 msg=input()
2 if msg=='How are you?'
3     print('Good!')
4 else:
5     print('Hello?')
```

れる。

[プログラム1-11] for文の範囲

```
1 for␣i␣in␣range(2):
2 ␣␣␣␣print('abc')
3 ␣␣␣␣print('def')
```

実行結果

```
abc
def
abc
def
```

for文はインデントされた範囲を繰り返して実行する。**プログラム1-11**は，回数として2を指定している。実行すると，インデントされた2つのprint関数が2回ずつ実行されて，abc，def，abc，defが表示される。

[プログラム1-12] for文で指定する変数の値

```
1 for␣i␣in␣range(3):
2 ␣␣␣␣print(i)
```

実行結果

```
0
1
2
```

for文で指定した変数には，実行を行うたびに値が代入される。**プログラム1-12**では，range(3)は0，1，2という3個の値を作り，それを変数iに1つずつ代入しながら処理が実行される。print関数では変数iの値を表示しているため，画面には0，1，2が表示される。

[プログラム1-13] for文で指定する変数の範囲

```
1 for␣i␣in␣range(1, 4):
2 ␣␣␣␣print(i)
```

MEMO

「for i in range(a):」と書くと，iに0から「a - 1」の値が代入されてa回繰り返す。一方，「for i in range(a, b):」と書くとiにaから「b - 1」の値が代入されて「b - a」回繰り返す。プログラム1-12とプログラム1-13を見比べると，この違いがよくわかる。

実行結果

```
1
2
3
```

range関数に2つの引数を渡すことで，変数iに入れる値の範囲を指定できる。**プログラム1-13**のrange(1, 4)は「1以上，4未満」という意味であり，変数iには1，2，3の値が1つずつ代入され，結果としてprint関数は3回実行される。

⑦ forで繰り返す範囲を意識しよう（インデント）

if文やfor文で実行する処理の範囲は**インデント**で指定する。if文やfor文はインデントの空白文字（半角スペース）の数で実行する範囲を判断しているため，インデントの空白文字数は4個などでそろえることが大切である。

［プログラム1-14］ for文で実行する処理の範囲

```
1 for␣i␣in␣range(2):
2 ␣␣␣␣print('a', end='')
3 ␣␣␣␣print('b', end='')
4 print('c', end='')
```

実行結果

```
ababc
```

プログラム1-14では，for文は2〜3行目のインデントされたaとbを表示するprint関数を繰り返して実行する。4行目のcを表示するprint関数はインデントされていないため，for文で繰り返し実行する範囲に入らない。結果として，aとbは2回ずつ表示され，cは1回だけ表示された。

練習問題

指定された実行結果が表示されるように，プログラム1-14のインデントを変更しよう。
［1］aabcを表示する。

MEMO
空白文字数は2個でも3個でも構わないが，同じ処理をさせるインデントは空白文字数をそろえなければならない。例えば，print('a')の前は空白が4個，print('b')の前は空白が3個のように空白文字の数が違っていると，エラーになる。

練習問題の解答例

［1］

```
1 for i in range(2):
2     print('A', end='')
3 print('B', end='')
4 print('C', end='')
```

［2］

```
1 for i in range(2):
2     print('A', end='')
3     print('B', end='')
4     print('C', end='')
```

［2］abcabcを表示する。

⑧ 数を計算しよう（算術演算子）

プログラムでは，文字列のほかに数値を扱うこともできる。数値の**四則演算**は，「+」「-」「*」「/」という**算術演算子**を使う。乗算には「×」ではなく「*」（アスタリスク）を使い，除算には「÷」ではなく「/」（スラッシュ）を使う。また，「%」（剰余）を使うと，割ったときの余りを求めることができる。**表1-2**に，使うことのできる算術演算子を示す。

また，変数には型と呼ばれる種類があり，**文字列型**や**数値型**がある。例えばa='abc'と書くと自動的に文字列型となり，a=1と書くと自動的に数値型となる。四則演算を行う際は，数値型の変数を使わなければならない。

［表1-2］算術演算子

演算	記号	意味
加算	+	左側の値に右側の値を足す
減算	-	左側の値から右側の値を引く
乗算	*	左側の値に右側の値を掛ける
除算	/	左側の値を右側の値で割る
剰余算	%	左側の値を右側の値で割った余り

❶

［プログラム1-15］数を計算

```
1 a = 11
2 b = 2
3 print('a + b =', a + b)
4 print('a - b =', a - b)
5 print('a * b =', a * b)
6 print('a / b =', a / b)
7 print('a % b =', a % b)
```

実行結果

```
a + b = 13
a - b = 9
a * b = 22
a / b = 5.5
```

次ページへ続く

除算は分数と考えればスラッシュでも違和感はないですね

MEMO
「/」を使った除算で割り切れない場合は，プログラム1-15のように小数点を含んだ商となる。小数点以下を切り捨てたものにしたい場合は「//」を使う。

MEMO
累乗を求める場合は「**」を使う。例えば，2の3乗を求めたい場合は2**3と記述する。

❶
剰余算「%」については，以下のような実行結果の具体例をいくつか見せて，その意味を考えさせるとよい。
print(10%3)
print(10%4)

```
a % b = 1
```
前ページから続く

プログラム1-15では，変数aと変数bに数を入れ，それらの加減乗除と剰余を計算している。見やすいように，計算結果の前に'a + b ='のような文字列を表示した。a，+などの文字は「'」で囲まれた場合は文字列と認識されるが，「'」で囲まない場合は変数や演算子として認識される。このため，'a + b ='はそのまま文字列として表示されるが，a + bは「11 + 2」の意味になり，結果の13が表示される。計算では，a * (b + c)のようにカッコを付けた計算も行える。

練習問題

プログラムを使って，以下の数式の答えを求めよう。割り算で割り切れない場合にはどのような表示になるかを観察しよう。

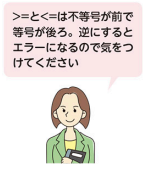
[1] 512 - 128
[2] 32 * 4
[3] 10 / 3
[4] 1303 % 17

⑨ 数を比べよう（比較演算子）

文字列では「等しいか，異なるか」を調べることができた。数値では「等しいか，異なるか」に加え，「大きい」「小さい」「以上」「以下」などを調べることができる。**表1-3**に，使うことのできる**比較演算子**を示す。

[表1-3] 比較演算子

意味	数学での記号	プログラミングでの記号
aはbより大きい	a>b	a > b
aはb以上	a≧b	a >= b
aはb以下	a≦b	a <= b
aはb未満	a<b	a < b
aとbは等しい	a=b	a == b
aとbは等しくない	a≠b	a != b



練習問題の解答例

[1] 384

[2] 128

[3] 3.3333333333333335

[4] 11

15 ページ

［プログラム1-16］数の大小を比較

```
1 x = 5
2 y = 3
3 if␣x > y:
4 ␣␣␣␣print(x)
5 else:
6 ␣␣␣␣print(y)
```

実行結果

```
5
```

プログラム1-16では，x > yで変数xと変数yの大小関係を調べている。結果として，大きいほうのxの値が表示された。

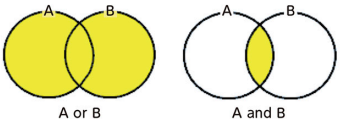
❶ **練習問題**

プログラム1-16の「x > y」の条件式を修正して，いろいろ比較をしてみよう。そのとき，xとyの値を変更して，条件式が成り立つときと成り立たないときの両方の動作を確認しよう。

⑩ 条件を組み合わせよう（論理演算子）

プログラムでは，複数の条件を組み合わせて使うことができる。「A and B」と書くと，条件Aと条件Bの両方が成り立つことを調べられる。一方，「A or B」と書くと，条件Aと条件Bのどちらかが成り立つことを調べられる。andやorは**論理演算子**と呼ばれる（**図1-2**）。

［図1-2］論理演算子で指定される範囲の例



▶**論理演算子**
andは2つ以上の条件を指定して，それら全てを満たすかどうかを調べる。orは2つ以上の条件を指定して，それらのうちいずれかを満たすかどうかを調べる。ほかにも，指定した条件を満たさないことを調べるnotもある。

MEMO
一般的に「A or B」は「AまたはB」と読む。「A and B」は「AかつB」と読む。

❶
問題文のように生徒に条件を変えさせる活動のほか，解答例のようなプログラムを提示して，実行結果を考えさせるクイズ形式の活動を取り入れてもよい。

練習問題の解答例

```
1 x=5
2 y=3
3 if x<=y:
4     print(x)
5 else:
6     print(y)
```

❶ **[プログラム1-17] andを使った論理演算**

```
1 x = 16
2
3 if␣x > 10␣and␣x % 5 == 0:
4 ␣␣␣␣print('OK')
5 else:
6 ␣␣␣␣print('NG')
```

実行結果

```
NG
```

プログラム1-17では，「xの値が10より大きい」と「xが5で割り切れる」が両方とも成り立つときにOKが表示され，そうでないときはNGが表示される。xの値を変えて，両方とも成り立つときやどちらか片方だけが成り立つときに，OKとNGのどちらが表示されるかを確認しよう。

[プログラム1-18] orを使った論理演算

```
1 x = 16
2
3 if␣x > 10␣or␣x % 5 == 0:
4 ␣␣␣␣print('OK')
5 else:
6 ␣␣␣␣print('NG')
```

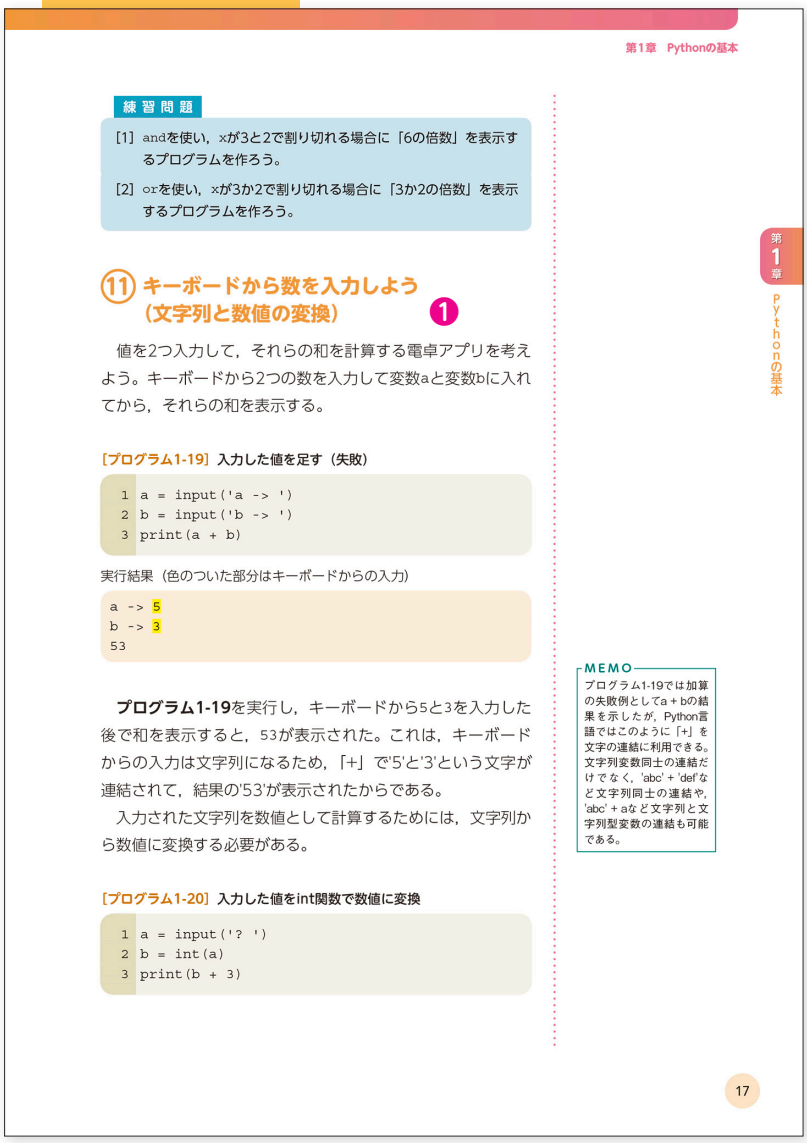
実行結果

```
OK
```

プログラム1-18では，「xの値が10より大きい」か「xが5で割り切れる」の少なくともどちらかが成り立つときにOKが表示され，両方とも成り立たないときはNGが表示される。xの値を変えて，どちらか片方だけが成り立つときや両方とも成り立たないときに，OKとNGのどちらが表示されるかを確認しよう。

❶
if 文の条件に and や or ををうまく使うと，短い行数で可読性の高いプログラムを作ることができる。

17 ページ



練習問題

[1] andを使い，xが3と2で割り切れる場合に「6の倍数」を表示するプログラムを作ろう。

[2] orを使い，xが3か2で割り切れる場合に「3か2の倍数」を表示するプログラムを作ろう。

⑪ キーボードから数を入力しよう（文字列と数値の変換） ❶

値を2つ入力して，それらの和を計算する電卓アプリを考えよう。キーボードから2つの数を入力して変数aと変数bに入れてから，それらの和を表示する。

[プログラム1-19] 入力した値を足す（失敗）

```
1 a = input('a -> ')
2 b = input('b -> ')
3 print(a + b)
```

実行結果（色のついた部分はキーボードからの入力）

```
a -> 5
b -> 3
53
```

プログラム1-19を実行し，キーボードから5と3を入力した後で和を表示すると，53が表示された。これは，キーボードからの入力は文字列になるため，「+」で'5'と'3'という文字が連結されて，結果の'53'が表示されたからである。

入力された文字列を数値として計算するためには，文字列から数値に変換する必要がある。

[プログラム1-20] 入力した値をint関数で数値に変換

```
1 a = input('? ')
2 b = int(a)
3 print(b + 3)
```

MEMO
プログラム1-19では加算の失敗例としてa + bの結果を示したが，Python言語ではこのように「+」を文字の連結に利用できる。文字列変数同士の連結だけでなく，'abc' + 'def'など文字列同士の連結や，'abc' + aなど文字列と文字列型変数の連結も可能である。

❶
「数なのに文字列として扱われる」という認識が不十分であると，これ以降に作成するプログラムが思い通りに動作しない原因となるため，丁寧に指導しておきたい。

練習問題の解答例

[1]

```
1 x=16
2 if x%2==0 and x%3==0:
3     print('6の倍数')

  # x=16の場合
```

[2]

```
1 x=16
2 if x%3==0 or x%2==0:
3     print('3か2の倍数')

  # x=16の場合
```

実行結果（色のついた部分はキーボードからの入力）

```
? 123
126
```

プログラム1-20では，input関数からの入力を変数aに入れた後，**int関数**で文字列から整数の数値に変換している。例えば，input関数に対して文字列123を入力すると，int関数によって文字列123が数値の123に変換されて変数bに入るため，「b + 3」の結果として126が表示される。

MEMO
int関数については67ページも参照。

［プログラム1-21］input関数の結果をすぐに数値に変換

```
1 a = int(input('? '))
2 print(a + 3)
```

❶

int関数を使えば数字の文字列を整数の数値に変換できるんですね。スゴイ！

実行結果（色のついた部分はキーボードからの入力）

```
? 123
126
```

なお，**プログラム1-21**のように「a = int(input('? '))」と書くことで，input関数の結果を数値に変換してから変数aに入れることもできる。

練習問題

［1］変数aと変数bに数値を入力し，「a + b」の計算結果を表示するプログラムを作ろう。

［2］変数aと変数bに数値を入力し，「a - b」の計算結果を表示するプログラムを作ろう。

❷ ⑫ 偶数だけを表示しよう（forとifの組み合わせ）

for文の処理の中でif文を使ってみよう。for文で繰り返して処理するため，if文はインデントして記述し，if文の条件が成り立つときに実行する部分はさらにインデントする必要がある。

❶
関数の入れ子は便利だが，難しく感じる生徒が多い。
a=int(input('? ')) が，次の２行と同じ意味であることを改めて確認させたい。
ipt=input('? ')
a=int(ipt)
また，閉じかっこの過不足からエラーになることも多いため，注意を促す必要がある。

❷
for と if を組み合わせると，一段と難易度が高くなる。組み合わせたプログラムについて考えさせる前に，改めて for 文と if 文の理解度を確認するとよい。

練習問題の解答例

［1］

```
1 a = int(input('a -> '))
2 b = int(input('b -> '))
3 print(a + b)
```

［2］

```
1 a = int(input('a -> '))
2 b = int(input('b -> '))
3 print(a - b)
```

❶ **[プログラム1-22]** for文の中のif文

```
1 for␣i␣in␣range(1, 10):
2 ␣␣␣␣if␣i > 5:
3 ␣␣␣␣␣␣␣␣print(i)
```

実行結果

```
6
7
8
9
```

プログラム1-22では，range(1, 10)は「1以上，10未満の数」を表すため，1，2，3，4，5，6，7，8，9という9つの数値が変数iに代入されてfor文が実行される。for文が実行する処理は，インデントされたif文である。if文の条件が成り立つときは，さらにインデントされたprint関数が実行される。結果として，1から9までの整数のうち，5より大きい6，7，8，9が表示された。

[プログラム1-23] 偶数だけを表示

```
1 for␣i␣in␣range(1, 10):
2 ␣␣␣␣if i % 2 == 0:
3 ␣␣␣␣␣␣␣␣print(i)
```

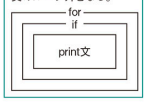
実行結果

```
2
4
6
8
```

プログラム1-23は，「1以上，10未満の数」の中で偶数だけを表示するプログラムである。「i % 2 == 0」で変数iが偶数かどうかを判断する。「i % 2」は変数iの値を2で割ったときの余りであり，その値が0の場合は「2で割り切れる」，つまり変数iの値が偶数であることがわかる。

MEMO

for文とif文を組み合わせるときは，インデントの空白数には十分に注意する必要がある。例えばプログラム1-22やプログラム1-23でprint(i)の下にもう1つ処理がある場合，空白が8個だとif文の条件内，4個だとif文の条件外でfor文のループ内，0個だとfor文のループ外となる。



「2で割り切れる」イコール「偶数」という考え方をプログラムにすると，「i % 2 == 0」になるんですね。つまり「2で割ったら，余りがなくて0になる」。なるほど！

❶
このような複雑な構造のプログラムは，見本を見ながら入力するだけでは本質的な理解につながらないことも多い。入力の前に，プログラムの実行結果を予想させる活動を取り入れるとよいだろう。

20 ページ

練習問題

このプログラムは，1から29までの数値を表示する。2行目のif文の条件式を，指定した表示が得られるように変更しよう。

[1]「23以上の数値」が表示される。
[2]「17より大きい数値」が表示される。
[3]「7未満の数値」が表示される。
[4]「19以下の数値」が表示される。
[5]「13以外の数値」が表示される。
[6]「11のみ」が表示される。
[7]「奇数の数値」が表示される。

```
1 for␣i␣in␣range(1, 30):
2 ␣␣␣␣if␣i >= 1:
3 ␣␣␣␣␣␣␣␣print(i)
```

20

練習問題の解答例

[1]

```
1 for i in range(1,30):
2     if i>=23:
3         print(i)
```

以降，２行目のみ

[2]

```
    if i>17:
```

[3]

```
    if i<7:
```

[4]

```
    if i<=19:
```

[5]

```
    if i!=13:
```

[6]

```
    if i==11:
```

[7]

```
    if i%2=1:
```