PROPLA

PROGRAMMING COURSE

Python

プログラミング入門

はじめての一歩に向いた、シンプルで読みやすい言語です。

27 LESSONS + PRACTICE

プロプラ

このコースについて

Pythonとは

Pythonは、読みやすく簡潔な文法を重視した、幅広い用途で使われるプログラミング言語です。 小さな自動化からWeb開発、データ分析、AIまで、同じ基本文法を使って段階的に取り組めます。

よく使われる分野	業務の自動化、Web開発、データ分析、AI・機械学習など。
書き方の特徴	インデントで処理のまとまりを表します。記号が比較的少なく、コードを上から読みやすいのが特徴です。
試せる環境	Python 3。ターミナル、IDLEなど、Pythonのコードを実行できる環境で試せます。

PDF版について

このPDFには、Web版と同じ全27レッスンと演習問題・模範解答を収録しています。コードはコピーでき、目次と各レッスンのWeb版へのリンクも利用できます。内容はWeb版の更新に合わせて生成されます。

https://propla.site/learn/python/

目次

LESSON 01 / はじめに

01 はじめの一歩

Web版で読む ↗

これは、プログラミングの初心者がPythonを用いてプログラミングの基礎を学ぶためのドキュメントです。実際に自分でコードを打ち込んで実行しながら、順を追って学習を進めてください。

数字を表示する

```python
print(2)
```

上のコードを実行すると、画面に`2`と表示されます。`print`の後の`( )`の中に書いた内容が出力（画面に表示）されます。

`print`（プリント）は「印刷する」という意味の英単語です。日本語でも「プリントする」というので分かりやすいと思います。ここでは、画面に表示することを「プリント」と呼んでいます。

```python
print(3)
```

`( )`の中を`3`に変えると、出力される内容も`3`に変わります。他にも、`( )`の中を色々な数値に変更して実行してみましょう。

間違えるとどうなるか

数字の代わりに英語のメッセージが表示されたときは、コードを見直してみてください。例えば、最後の`)`を書き忘れると、プログラムの実行は失敗します。

```python
print(2
```

実行すると、次のような**エラーメッセージ**が表示されます。

```
TEXT

SyntaxError: unexpected EOF while parsing
```

エラーメッセージは、プログラムに間違いがあることを教えてくれるメッセージです。

> **NOTE**
>
> エラーメッセージの読み方は、後のレッスンで説明します。

`print(2)`を大文字や全角文字で書いた場合も、正しく動きません。プログラムは、一文字間違えただけで正しく動かないことも多いです。

`)`を書き直し、もう一度実行して、正しく数字が表示されることを確認してください。

自分で色々試してみよう

プログラミングを学ぶときには、コードに自分で色々と変更を加えてみて、実行結果がどう変わるのかを試すことが大切です。そのような変更と実行を通して、プログラムがどのように動作するのかを感覚的に理解することができます。

特に、自分が疑問を感じた点や、「ここを変更するとどうなるんだろう？」と感じた箇所では、必ず変更と実行を通して理解を深めてください。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

数字の`3`を画面に表示してください。

問題2

数字の`10`を画面に表示してください。

問題3

`1`、`5`、`9`を、この順に1行ずつ表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
print(3)
```

`( )`の中に書いた`3`が、そのまま画面に表示されます。

問題2

模範解答

```python
print(10)
```

問題1と同じように、`( )`の中を表示したい数字に変更します。

問題3

模範解答

```python
print(1)
print(5)
print(9)
```

表示したい数字ごとに、出力するコードを1行ずつ書きます。コードは上から順に実行されます。

LESSON 02 / はじめに

02 式と文字列

Web版で読む ↗

式を計算する

```python
print(2 + 3)
```

`( )`の中を`2 + 3`に変更してみると、`2 + 3`ではなく計算結果である`5`が出力されます。Pythonでは（Pythonに限らず普通のプログラミング言語では）式は最初に計算が実行され、その結果が`print`されます。

文字列として出力する

```python
print("2 + 3")
```

もし`2 + 3`という式を`print`したいときには、ダブルクォーテーション（`"`）で囲むと、計算せずにそのまま`2 + 3`と出力されます。`"`は日本語の「」に当たります。Pythonでは（また、多くの他のプログラミング言語でも）`"`で囲まれた部分はその文字の並びそのものを表します。`print("2 + 3")`は、「`2 + 3`という文字の並びを`print`しろ」という意味になり、そのため、`2 + 3`がそのまま出力されます。

`"2 + 3"`のような文字の並びを**文字列（もじれつ, character string）**と呼びます。

```python
print("Hello")
```

`"`で囲まれた部分には自由に文字列を書くことができます。たとえば、`print("Hello")`を実行すると`Hello`と出力されます。

NOTE

文字列の中に"を書きたい場合は、"の前に\（バックスラッシュ）を付けて\"と書きます。例えば、次のように書きます。

```python
print("\"")
```

このように、\を使って特別な文字を表す書き方を**エスケープ（escape）**と呼びます。\"のほかにも、たとえば改行は\n、タブは\tと書いて表します。上のコードを実行すると、"と出力されます。

NOTE

Pythonでは、'Hello'のように'（シングルクォーテーション）で囲んでも文字列を書くことができます。文字列の中に"を書きたい場合は'で囲み、'を書きたい場合は"で囲むと、エスケープを減らせます。

例えば、print('"Hello"')やprint("It's easy")のように書けます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

4 + 6を計算し、その結果を表示してください。

問題2

計算せずに、文字列の4 + 6をそのまま表示してください。

問題3

次のように2行で表示してください。文字列の中の"と改行には、エスケープを使います。

```text
He said "Hello".
Good morning
```

ANSWERS

模範解答と解説

問題1

模範解答

```python
print(4 + 6)
```

式が先に計算されるため、`4 + 6`ではなく、計算結果の`10`が表示されます。

問題2

模範解答

```python
print("4 + 6")
```

`4 + 6`を`"`で囲むと文字列になるため、計算されずにそのまま表示されます。

問題3

模範解答

```python
print("He said \"Hello\".\nGood morning")
```

文字列の中の`"`は`\"`、改行は`\n`と書きます。どちらも、`\`を使ったエスケープです。

LESSON 03 / はじめに

03 四則演算

Web版で読む ↗

足し算

```python
print(2 + 3)
```

これまで見てきたように、`+`で足し算の計算ができます。

引き算

```python
print(6 - 2)
```

足し算の他にも、もちろん引き算をすることもできます。

掛け算

```python
print(2 * 3)
```

掛け算には、×ではなく`*`（アスタリスク）を使います。

割り算

```python
print(6 / 2)
```

割り算には、÷ではなく`/`（スラッシュ）を使います。

Pythonでは、この計算結果は`3.0`と表示されます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

`8 + 5`を計算し、その結果を表示してください。

問題2

`9 - 4`、`7 * 3`、`20 / 5`の結果を、それぞれ1行ずつ表示してください。

問題3

次の三つを計算し、結果を1行ずつ表示してください。

- 1冊150円のノートを4冊買ったときの合計金額
- 1000円から600円を使ったときの残り
- 18個のお菓子を2人で同じ数ずつ分けたときの1人分

ANSWERS

模範解答と解説

問題1

模範解答

```python
print(8 + 5)
```

足し算には`+`を使います。計算結果の`13`が表示されます。

問題2

模範解答

```python
print(9 - 4)
print(7 * 3)
print(20 / 5)
```

引き算には`-`、掛け算には`*`、割り算には`/`を使います。割り算の結果は`4.0`と表示されます。

問題3

模範解答

```python
print(150 * 4)
print(1000 - 600)
print(18 / 2)
```

それぞれの場面を、掛け算、引き算、割り算の式で表します。結果は順に`600`、`400`、`9.0`です。

LESSON 04 / はじめに

04 計算の順序

Web版で読む ↗

複数の数を計算する

```
PYTHON

print(2 + 3 + 5)
```

3個以上の数を使った式を計算することもできます。

計算の順序

```
PYTHON

print(2 + 3 * 5)
```

算数や数学と同じように、掛け算は足し算より先に計算されます。`2 + 3 * 5`では、まず`3 * 5`が計算されて`15`になり、その後`2 + 15`が計算されるので`17`が出力されます。

```
PYTHON

print((2 + 3) * 5)
```

算数や数学と同じように、`( )`を使って計算の順序を変えられます。`(2 + 3) * 5`では、まず`2 + 3`が計算されて`5`になり、その後`5 * 5`が計算されるので`25`が出力されます。

```
PYTHON

print(((2 + 3) * 5 +7) * 11)
```

`( )`を複数重ねることもできます。数学では、括弧（かっこ）を重ねる場合、中括弧`{ }`や大括弧`[ ]`を使いますが、Pythonでは（他の通常のプログラミング言語でも）`( )`を重ねて使います。

数学と同じように、括弧は内側から計算されます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

`2 + 4 * 3`を計算し、その結果を表示してください。

問題2

`2 + 4`を先に計算してから、その結果に`3`を掛けて表示してください。

問題3

1個150円の商品を2個、1個80円の商品を3個買います。合計金額を一つの式で計算して表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
print(2 + 4 * 3)
```

掛け算が足し算より先に計算されるため、結果は`14`になります。

問題2

模範解答

```python
print((2 + 4) * 3)
```

先に計算したい`2 + 4`を`( )`で囲みます。結果は`18`です。

問題3

模範解答

```python
print(150 * 2 + 80 * 3)
```

二つの掛け算が先に計算され、その結果が足されます。合計金額は`540`円です。

LESSON 05 / はじめに

05 値の比較

Web版で読む ↗

数の大小を比べる

```
PYTHON

print(2 < 3)
```

四則演算だけでなく、数の大小を比べる式も実行できます。`2 < 3`（2は3より小さい）は正しいので、`True`（真）が出力されます。

```
PYTHON

print(2 > 3)
```

`2 > 3`（2は3より大きい）は正しくないので、`False`（偽）が出力されます。

同じ値か比べる

```
PYTHON

print(2 == 2)
```

Pythonでは、二つの値が同じかどうかを`==`で比べます。`2`と`2`は同じなので、`True`（真）が出力されます。

```
PYTHON

print(2 != 3)
```

反対に、二つの値が同じでないかどうかは、`!=`で比べます。`2`と`3`は同じでないので、`True`（真）が出力されます。

また、`==`や`!=`を使って、文字列が同じかどうかも比べられます。

`=`を一つだけ書いたものは、値が同じかどうかを比べる記号ではありません。`=`については、次の「変数」のレッスンで説明します。

真と偽を反対にする

```python
print(not 2 < 3)
```

`2 < 3`の結果は`True`（真）です。その前に`not`を付けると、真と偽が反対になり、`False`（偽）が出力されます。Pythonでは、真と偽を反対にするときに`!`ではなく`not`を使います。`not`は、「〜ではない」という否定を表します。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

`8`が`12`より小さいかを比べ、その結果を表示してください。

問題2

文字列の`"cat"`と`"cat"`が同じか、文字列の`"cat"`と`"dog"`が同じでないかを比べ、結果を1行ずつ表示してください。

問題3

`10`が`20`より大きい、という比較の結果を反対にして表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
print(8 < 12)
```

`8 < 12`は正しいため、`True`が表示されます。

問題2

模範解答

```python
print("cat" == "cat")
print("cat" != "dog")
```

同じかどうかと、同じでないかどうかは文字列についても比べられます。どちらの結果も`True`です。

問題3

模範解答

```python
print(not 10 > 20)
```

`10 > 20`は偽ですが、`not`で結果を反対にするため、`True`が表示されます。

LESSON 06 / 変数

06 変数

Web版で読む ↗

同じ計算を繰り返す問題

```python
print(5 * 5 * 3.14)
print(5 * 5 * 3.14 * 8)
```

半径が5、高さが8の円柱の底面積と体積を計算しましょう。

底面積は半径5の円の面積なので`5 * 5 * 3.14`で計算できます。また、体積は底面積に8を掛ければいいので`5 * 5 * 3.14 * 8`で計算できます。

しかし、このとき、`5 * 5 * 3.14`という計算を2回もしており、無駄です。

変数を使う

```python
a = 5 * 5 * 3.14
print(a)
print(a * 8)
```

変数（へんすう, variable）を使うと、計算結果に名前を付けて保存しておき、後で使うことができます。

ここでは、`5 * 5 * 3.14`の計算結果に`a`という名前を付けて保存しています。`a = 5 * 5 * 3.14`で、`5 * 5 * 3.14`の結果が`a`に保存されます。この`a`が変数です。また、`a`に値を保存することを**代入（だいにゅう, assignment）**と言います。

`print(a)`では、`a`に代入（保存）された`5 * 5 * 3.14`の結果が出力されます。

また、`a * 8`では`a`に代入された`5 * 5 * 3.14`の結果に`8`を掛けるので、`print(a * 8)`を実行すると体積が出力されます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

半径が`4`、高さが`7`の円柱について、底面積を変数`a`に保存し、底面積と体積を表示してください。円周率は`3.14`とします。

問題2

`12 * 4`の計算結果を変数`a`に保存し、その値を表示してください。

問題3

一辺が`7`の正方形の面積を変数`a`に保存してください。その面積と、同じ正方形5個分の面積を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
a = 4 * 4 * 3.14
print(a)
print(a * 7)
```

底面積を一度だけ計算して変数`a`に保存します。体積の計算では、保存した値を`a * 7`として再利用できます。

問題2

模範解答

```python
a = 12 * 4
print(a)
```

`12 * 4`の結果である`48`が変数`a`に保存され、変数を出力すると`48`が表示されます。

問題3

模範解答

```python
a = 7 * 7
print(a)
print(a * 5)
```

一度求めた面積を変数aに保存しておけば、同じ計算を繰り返さずに5個分の面積を求められます。

LESSON 07 / 変数

07 変数名と複数の変数

Web版で読む ↗

分かりやすい名前を付ける

```python
teimenseki = 5 * 5 * 3.14
print(teimenseki)
print(teimenseki * 8)
```

変数名（変数の名前）は自由に付けられます。分かりやすい名前を付けておくと、後でその変数が何を表しているのか理解しやすくなります。ここでは、`teimenseki`（底面積）という名前を付けています。

```python
base = 5 * 5 * 3.14
print(base)
print(base * 8)
```

変数名には英語を使うことも多いので、`base`（底面）のような変数名も良いでしょう。

複数の変数を使う

```python
r = 5
base = r * r * 3.14
h = 8
print(base)
print(base * h)
```

複数の変数を使うこともできます。ここでは、`r`（半径を表すradiusの頭文字）、`h`（高さを表すheightの頭文字）と`base`の3個の変数を使っています。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

半径が`4`、高さが`6`の円柱について、`radius`、`base`、`height`という変数を使って底面積と体積を表示してください。円周率は`3.14`とします。

問題2

幅`8`、高さ`5`の長方形について、`width`、`height`、`area`という三つの変数を使って面積を計算し、表示してください。

問題3

国語、数学、英語の点数がそれぞれ`72`、`88`、`80`です。`japanese`、`math`、`english`、`total`という変数を使って合計点を求め、さらに平均点を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
radius = 4
base = radius * radius * 3.14
height = 6
print(base)
print(base * height)
```

値の意味が分かる変数名にすると、どの計算をしているコードなのか読み取りやすくなります。

問題2

模範解答

```python
width = 8
height = 5
area = width * height
print(area)
```

複数の値を別々の変数に保存し、その変数を使って計算できます。`area`には`40`が保存されます。

問題3

模範解答

```python
japanese = 72
math = 88
english = 80
total = japanese + math + english
print(total / 3)
```

三つの点数を`total`にまとめてから科目数の`3`で割ります。意味を表す変数名を使うことで、計算の内容が分かりやすくなります。

LESSON 08 / 変数

08 変数に保存できる値

Web版で読む ↗

文字列を代入する

```python
a = "Hello"
print(a)
```

変数には数値だけでなく、文字列を代入することもできます。

ブール値を代入する

```python
a = 2 < 3
print(a)
```

比較の結果（`True`または`False`）も代入できます。

```python
a = True
print(a)
```

`True`や`False`を直接代入することもできます。`True`および`False`の値を**ブール値（ブールち, boolean value）**と言います。

値の型

数値、文字列、ブール値のような値の種類を**型（かた, type）**と呼びます。

値の種類	Python
整数	`int`
小数	`float`
文字列	`str`
ブール値	`bool`

Pythonでは、変数に別の型の値を代入することもできます。値がどの型かは、プログラムを実行しているときに調べられます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

文字列の`"Hello"`を変数`message`に代入し、その値を表示してください。

問題2

`5`が`8`より小さいかを比べた結果を変数`isSmall`に代入し、その値を表示してください。

問題3

名前`"Kai"`、年齢`14`、身長`158.5`、合格していることを表すブール値を、それぞれ`name`、`age`、`height`、`passed`に代入して表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
message = "Hello"
print(message)
```

変数には数値だけでなく、文字列も保存できます。`message`には文字列の値が代入されています。

問題2

模範解答

```python
isSmall = 5 < 8
print(isSmall)
```

比較の結果はブール値です。比較が正しいため、`True`が変数に保存されます。

問題3

模範解答

```python
name = "Kai"
age = 14
height = 158.5
passed = True

print(name)
print(age)
print(height)
print(passed)
```

文字列、整数、小数、ブール値は、それぞれ異なる型の値です。

LESSON 09 / 変数

09 再代入

Web版で読む ↗

値を上書きする

```python
a = 2
a = 3
print(a)
```

変数の値はあとから変更できます。最初に`a = 2`で`2`を保存し、次に`a = 3`で`3`に上書きしています。そのため、`print(a)`すると`3`が出力されます。

今の値を使って更新する

```python
age = 12
print(age)

age = age + 1
print(age)
```

変数に格納された値を使って計算し、結果を同じ変数に代入することもできます。`age`は年齢を表す変数ですが、誕生日が来て年齢を1増やす処理は、`age = age + 1`というように書けます。

`age = age + 1`という式は、数学では`=`が成り立たず不正な式ですが、プログラミング言語では`=`は代入を表すので、「`=`の右側の値を、`=`の左側の変数に代入する」という意味になります。`age ← age + 1`（`age + 1`の結果を`age`に入れる）のように考えると分かりやすいかもしれません。

省略して書く

```python
age = 12
print(age)

age += 1
print(age)
```

`age = age + 1`などのコードはよく書くので、省略形が用意されています。`age += 1`と書くと、`age = age + 1`という意味になります。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

変数`a`に最初は`2`を代入し、その後`5`を再代入してから、`a`を表示してください。

問題2

変数`age`に`14`を代入し、今の値を使って`1`増やしてから表示してください。`+=`を使って書いてみましょう。

問題3

在庫数を表す変数`stock`を`20`で始めます。商品が`8`個入荷したので増やし、その後`5`個売れたので減らして、最後の在庫数を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
a = 2
a = 5
print(a)
```

後から代入した`5`で元の値が上書きされるため、`5`が表示されます。

問題2

模範解答

```python
age = 14
age += 1
print(age)
```

`age += 1`は`age = age + 1`と同じ意味です。更新後の`15`が表示されます。

問題3

模範解答

```python
stock = 20
stock += 8
stock = stock - 5
print(stock)
```

今の在庫数を使って、入荷分を足し、販売分を引いています。最後の在庫数は`23`です。

LESSON 10 / 条件分岐

10 条件分岐

Web版で読む ↗

条件を満たしたときだけ実行する

```python
age = 12
if age < 20:
    print("お酒を飲めません")
```

20歳未満の場合はお酒を飲めません。`age`が`20`未満の場合だけ「お酒を飲めません」と表示するには`if`を使います。`if`の後には条件（この場合は`age < 20`）を書き、それが正しい（`True`の）場合だけ実行する処理を書くことができます。

条件の後には`:`を書き、次の行からインデント（字下げ）してコードを書きます。インデントはスペースキーで行ってもいいですが、Tabキーを押して行うのが一般的です。

複数の処理をまとめる

```python
age = 12
if age < 20:
    print("お酒を飲めません")
    print("ジュースでも飲んでね")
```

インデントされたコードを複数行書くと、そのインデントされた範囲全体が`if`の対象となります。このインデントされた範囲を**ブロック（block）**と言います。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

年齢を表す変数`age`が`20`より小さいときだけ、`"20歳未満です"`と表示してください。`age`は`16`とします。

問題2

点数を表す変数`score`が`79`より大きいときだけ、`"合格です"`と`"よくできました"`を1行ずつ表示してください。`score`は`85`とします。

問題3

気温を表す変数`temperature`を`7`とします。気温が`10`より小さいときは`"上着を着よう"`、気温が`30`より大きいときは`"水分をとろう"`と表示する二つの`if`を書いてください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
age = 16
if age < 20:
    print("20歳未満です")
```

`age < 20`が真のときだけ、`if`のブロックが実行されます。

問題2

模範解答

```python
score = 85
if score > 79:
    print("合格です")
    print("よくできました")
```

二つの出力を同じブロックに書くと、条件を満たしたときに両方が実行されます。

問題3

模範解答

```python
temperature = 7

if temperature < 10:
    print("上着を着よう")

if temperature > 30:
    print("水分をとろう")
```

二つの条件を別々の`if`で調べます。今回は最初の条件だけが真なので、`上着を着よう`だけが表示されます。

LESSON 11 / 条件分岐

11 elseとelif

Web版で読む ↗

条件を満たさない場合

PYTHON

```
age = 12
if age < 20:
    print("お酒を飲めません")
else:
    print("お酒を飲めます")
```

「もし〜ならA、そうでなければB」のように、「そうでなければ」実行される処理を書くには`else:`を使います。`else:`の後のブロックは、条件を**満たさなかった**場合にだけ実行されます。

三つ以上に分岐する

PYTHON

```
age = 12
if age < 0:
    print("エラーです")
elif age < 20:
    print("お酒を飲めません")
else:
    print("お酒を飲めます")
```

「もし〜ならA、もし…ならB、そうでなければC」のように三つ以上に分岐したい場合には`elif`を使います。`elif`の後には`if`と同じように条件を書くことができ、その条件を満たした場合にだけ、`elif`のブロックが実行されます。`elif`はelse ifの略で、他のプログラミング言語では`else if`と書くものも多いです。

条件は上から順にチェックされ、最初に`True`になった部分が実行されます。一度条件を満たして分岐されたコードに突入すると、他の分岐が実行されることはありません。

```python
age = 12
if age < 0:
    print("エラーです")
elif age < 20:
    print("お酒を飲めません")
elif age < 70:
    print("お酒を飲めます")
else:
    print("お酒を飲まない方がいいですね")
```

`elif`は複数記述することができます。

`if`は最初に一つ、`else`は最後に一つだけです（`if`は必ず必要、`else`はなくても良い）。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

年齢を表す変数`age`が`20`より小さければ`"20歳未満です"`、そうでなければ`"20歳以上です"`と表示してください。`age`は`22`とします。

問題2

点数`score`が`80`より大きければ`"A"`、`60`より大きければ`"B"`、それ以外なら`"C"`と表示してください。`score`は`75`とします。

問題3

年齢`age`が`6`より小さければ入場料を`0`円、`18`より小さければ`500`円、それ以外なら`1000`円と表示してください。`age`は`14`とします。

ANSWERS

模範解答と解説

問題1

模範解答

```python
age = 22
if age < 20:
    print("20歳未満です")
else:
    print("20歳以上です")
```

`age < 20`は偽なので、`else`のブロックが実行されます。

問題2

模範解答

```python
score = 75
if score > 80:
    print("A")
elif score > 60:
    print("B")
else:
    print("C")
```

条件は上から調べられます。最初の条件は偽で、次の条件が真になるため、`B`が表示されます。

問題3

模範解答

```PYTHON
age = 14
if age < 6:
    print(0)
elif age < 18:
    print(500)
else:
    print(1000)
```

`age`は`6`より小さくはありませんが、`18`より小さいため、二つ目の分岐で`500`が表示されます。

LESSON 12 / 条件分岐

12 ifのネスト

Web版で読む ↗

条件分岐を重ねる

PYTHON

```
gender = "male"
age = 12
if gender == "male":
    if age < 18:
        print("結婚できません")
    else:
        print("結婚できます")
else:
    if age < 16:
        print("結婚できません")
    else:
        print("結婚できます")
```

2022年までの日本の法律では、男性は18歳以上・女性は16歳以上になると結婚することができました。この条件で`gender`（性別）と`age`（年齢）から結婚できるか判定するコードは上記のようになります。

このコードでは、`if`で分岐した後でさらに`if`で分岐しています。これを`if`の**ネスト（nest）**と言います。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

天気`weather`が`"rainy"`なら、さらに気温`temperature`を調べます。気温が`10`より小さければ`"傘と上着"`、そうでなければ`"傘"`と表示してください。雨でなければ`"傘は不要"`と表示します。天気は`"rainy"`、気温は`8`とします。

問題2

会員かどうかを表す`member`が`"yes"`のときだけ購入金額`total`を調べ、`total`が`5000`より大きければ`"割引あり"`、そうでなければ`"通常価格"`と表示してください。会員でなければ`"会員登録が必要"`と表示します。`member`は`"yes"`、`total`は`6200`とします。

問題3

信号`light`が`"green"`のとき、さらに車`car`が`"none"`かを調べます。車がいなければ`"渡る"`、いれば`"待つ"`と表示してください。信号が青でなければ`"止まる"`と表示します。`light`は`"green"`、`car`は`"coming"`とします。

ANSWERS

模範解答と解説

問題1

模範解答

PYTHON

```
weather = "rainy"
temperature = 8

if weather == "rainy":
    if temperature < 10:
        print("傘と上着")
    else:
        print("傘")
else:
    print("傘は不要")
```

最初の`if`で天気を分岐し、そのブロックの中の`if`で気温を分岐しています。今回は`傘と上着`が表示されます。

問題2

模範解答

```PYTHON
member = "yes"
total = 6200

if member == "yes":
    if total > 5000:
        print("割引あり")
    else:
        print("通常価格")
else:
    print("会員登録が必要")
```

会員であることを確認した後で、購入金額についてもう一度分岐しています。今回は`割引あり`が表示されます。

問題3

模範解答

```PYTHON
light = "green"
car = "coming"

if light == "green":
    if car == "none":
        print("渡る")
    else:
        print("待つ")
else:
    print("止まる")
```

外側の条件を満たした場合だけ、内側の条件が調べられます。今回は信号は青ですが車がいるため、`待つ`が表示されます。

LESSON 13 / ループ

13 ループ

Web版で読む ↗

繰り返して書く問題

```python
print(0)
print(1)
print(2)
...
print(9999)
```

0から9999までの数字を順番に出力することを考えます。`print(0)`から`print(9999)`まで書けばもちろんできますが、10000行のコードになってしまい大変です。

```python
a = 0
print(a)
a += 1
print(a)
a += 1
print(a)
...
```

変数を使って書けば`print(a)`と`a += 1`のコピー&ペーストで書くことができますが、それでも大変です。

whileで繰り返す

```python
a = 0
while a < 10000:
    print(a)
    a += 1
```

同じ処理を繰り返すには`while`を使います。`while`は`if`に似ていますが、ブロックの実行が終わるともう一度`a < 10000`の条件がチェックされます。そして、もし条件が正しければ、もう一度`while`のブロックが実行されます。このようにして、条件が正しい間は処理が繰り返されます。

このような繰り返しを**ループ（loop）**と言います。

> WARNING
>
> `a += 1`を書き忘れると、`a < 10000`がいつまでも成り立つため、ループが終わりません。このように処理が終わらないループを**無限ループ**と呼びます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

`while`を使って、`0`から`4`までの数字を順番に表示してください。

問題2

`while`を使って、`5`から`1`までの数字を順番に表示してください。

問題3

`while`を使って、`1`から`10`までの整数の合計を求め、最後に合計だけを表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
a = 0
while a < 5:
    print(a)
    a += 1
```

`a`を`0`から始め、表示するたびに`1`増やします。`a`が`5`になると条件が偽になり、ループが終わります。

問題2

模範解答

```python
a = 5
while a > 0:
    print(a)
    a = a - 1
```

今回はループ変数を`1`ずつ減らします。`a`が`0`になると`a > 0`が偽になり、ループが終わります。

問題3

模範解答

```python
number = 1
total = 0

while number < 11:
    total += number
    number += 1

print(total)
```

`number`を1ずつ増やしながら、その値を`total`に足します。ループが終わると`total`は`55`になります。

LESSON 14 / リスト

14 リスト

Web版で読む ↗

複数の変数で管理する問題

```python
a1 = 65
a2 = 90
a3 = 85

total = a1 + a2 + a3
print(total / 3)
```

クラス全員のテストの点数から平均点を計算するには、すべての人の点数を足してから人数で割る必要があります。

今、クラスには3人しかいないものとして、変数`a1`, `a2`, `a3`にそれぞれの点数が格納されているものとします。このとき、合計点は`total = a1 + a2 + a3`で計算でき、これを`total / 3`のように人数で割ることで平均点を求めることができます。

複数の値を一つにまとめる

```python
a = [65, 90, 85]

print(a[0])
print(a[1])
print(a[2])
```

`a1`, `a2`, `a3`のように変数を増やしていくと管理が大変です。**リスト（list）**を使えば、複数の値を一つの変数で管理することができます。

`[65, 90, 85]`のようにしてリストを作ることができます。`a = [65, 90, 85]`とすると、`a`には`65`, `90`, `85`の3個の値を持つリストが格納されます。

このとき、`a[0]`で最初の値`65`に、`a[1]`で2番目の値`90`に、`a[2]`で3番目の値`85`にアクセスすることができます。このように、リストでは値を0から数えます。0番目、1番目、2番目と言うこともあります。

`[ ]`の中に書き、何番目かを表す値を**インデックス（index）**と言います。また、それで得られるリストの一つ一つの値を**要素（ようそ, element）**と言います。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

`10`、`20`、`30`を一つのリストにまとめて変数`numbers`に代入し、三つの要素を順番に表示してください。

問題2

`"red"`、`"green"`、`"blue"`、`"yellow"`を一つのリストにまとめ、2番目と4番目の要素を表示してください。

問題3

三人の点数`72`、`88`、`80`を一つのリストにまとめ、インデックスを使って平均点を計算し、表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
numbers = [10, 20, 30]

print(numbers[0])
print(numbers[1])
print(numbers[2])
```

最初の要素のインデックスは`0`です。`0`、`1`、`2`を指定すると、三つの要素へ順番にアクセスできます。

問題2

模範解答

```python
colors = ["red", "green", "blue", "yellow"]

print(colors[1])
print(colors[3])
```

インデックスは`0`から始まるため、2番目には`1`、4番目には`3`を指定します。

問題3

模範解答

```python
scores = [72, 88, 80]
total = scores[0] + scores[1] + scores[2]
print(total / 3)
```

三つの要素をインデックスで取り出して合計し、人数の`3`で割ります。平均点は`80`です。

LESSON 15 / リスト

15 リストとwhile

Web版で読む ↗

要素を更新する

```python
a = [65, 90, 85]

print(a[0])
print(a[1])
print(a[2])

a[0] = 67

print(a[0])
```

`a[0] = 67`のように、インデックスを指定して特定の要素だけ更新することもできます。

ループですべての要素を処理する

```python
a = [65, 90, 85]

i = 0
while i < 3:
    print(a[i])
    i += 1
```

リストを使えばループと組み合わせて処理することができます。`a[i]`のように`[ ]`の中に変数を使うことができるので、ループで変数の値を1ずつ増やしながら処理することができるからです。

この`i`のように、ループの制御に使う変数を**ループ変数**と呼ぶことがあります。ループ変数にはindexの頭文字を取ってよく`i`が使われます。

要素数に合わせて繰り返す

```python
a = [65, 90, 85]
print(len(a))
```

`i < 3`のように、人数をコード中にベタ書き（ハードコード）してしまうと、人数が増えたときにコードも修正しなければなりません。`len`を使えばリストの長さ（要素数）を得ることができます。この例では`3`が出力されます。

```python
a = [65, 90, 85]
n = len(a)

i = 0
while i < n:
    print(a[i])
    i += 1
```

`len`を使えば、さっきのループのコードはこのように書き換えられます。

```python
a = [65, 90, 85]

i = 0
while i < len(a):
    print(a[i])
    i += 1
```

`len`の結果を変数に代入せず、条件式に直接`len(a)`を使うこともできます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

点数`65`、`90`、`85`をまとめたリストの最初の要素を`70`に更新し、更新後の値を表示してください。

問題2

`[4, 8, 12, 16]`のすべての要素を、`while`を使って順番に表示してください。繰り返す回数は要素数から求めてください。

問題3

点数`60`、`75`、`90`をまとめたリストがあります。`while`を使って、すべての点数を`5`ずつ増やし、更新後の値を順番に表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
scores = [65, 90, 85]
scores[0] = 70
print(scores[0])
```

インデックス`0`を指定して代入すると、最初の要素だけを更新できます。

問題2

模範解答

```python
numbers = [4, 8, 12, 16]

i = 0
while i < len(numbers):
    print(numbers[i])
    i += 1
```

ループ変数`i`をインデックスとして使います。`len(numbers)`まで繰り返すため、要素数を変えても同じコードで処理できます。

問題3

模範解答

```python
scores = [60, 75, 90]

i = 0
while i < len(scores):
    scores[i] += 5
    print(scores[i])
    i += 1
```

`scores[i]`を使うと、ループのたびに異なる要素を更新できます。表示される値は`65`、`80`、`95`です。

LESSON 16 / リスト

16 リストとfor

Web版で読む ↗

forで要素を取り出す

```python
a = [65, 90, 85]

for b in a:
    print(b)
```

whileとループ変数を使ってリストの各要素にアクセスすることもできますが、これは頻繁に行われる処理なので、もっと便利な手段が用意されています。それがforです。

forを使うと、リストの各要素を順番に取り出して処理することができます。この場合、aの要素が一つずつ順番に取り出されbに格納されてからブロックの内容が繰り返し実行されます。

```python
scores = [65, 90, 85]

for score in scores:
    print(score)
```

> **TIP**
>
> リストには英語の名詞の複数形の変数名が、取り出された要素を格納するには単数形の変数名が用いられることが多いです。こうすれば、変数に格納されているのがリストなのか、単体の値なのかを区別しやすくなります。

平均点を計算する

```python
scores = [65, 90, 85]

total = 0
for score in scores:
    total += score

print(total / len(scores))
```

`for`を使うと、平均点を計算する処理は上記のように書けます。

このように、ループとリストを使えば、たとえ10000人の平均点を計算するコードでも簡単に書くことができます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

点数`65`、`90`、`85`をまとめたリストから、`for`ですべての要素を順番に表示してください。

問題2

`[3, 5, 7, 9]`のすべての要素を足し、合計を表示してください。

問題3

点数`45`、`72`、`88`、`59`のうち、`59`より大きい点数がいくつあるか数えて表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
scores = [65, 90, 85]

for score in scores:
    print(score)
```

要素が一つずつ`score`に入り、そのたびにブロックが実行されます。インデックスを自分で管理する必要はありません。

問題2

模範解答

```python
numbers = [3, 5, 7, 9]
total = 0

for number in numbers:
    total += number

print(total)
```

取り出した要素を順番に`total`へ足します。ループの後では、合計の`24`が保存されています。

問題3

模範解答

PYTHON

```
scores = [45, 72, 88, 59]
passed = 0

for score in scores:
    if score > 59:
        passed += 1

print(passed)
```

各要素を`if`で調べ、条件を満たすたびに`passed`を`1`増やします。該当する点数は`72`と`88`なので、`2`が表示されます。

LESSON 17 / ループ

17 レンジ

Web版で読む ↗

連続する数を扱う

前に、0から9999まで順番に表示するコードを書きました。

PYTHON

```
a = 0
while a < 10000:
    print(a)
    a += 1
```

0から9999までの値をすべてリストに書くのは大変です。

PYTHON

```
numbers = [0, 1, 2, ..., 9999]

for number in numbers:
    print(number)
```

これもリストで書きたいところですが、10000個の要素を書くのは大変です。

レンジを使う

PYTHON

```
numbers = range(10000)

for number in numbers:
    print(number)
```

レンジ（range）を使えば簡単にこれを実現できます。`range(10000)`とすると、`0`から`9999`までの範囲の値を持つレンジオブジェクトを作ることができます。レンジからも、リストと同じように`for`で値を一つずつ取り出してループすることができます。

```python
for a in range(10000):
    print(a)
```

`in`の右側に直接`range(10000)`のように書くこともできます。

始まりを指定する

```python
for a in range(10, 20):
    print(a)
```

`range(10, 20)`のようにして、レンジの始まりを指定することもできます。

範囲の扱い方

```python
r = range(10, 20)
print(r[0])
r[0] = 999  # エラー
```

レンジもリストと同じように`r[0]`のようにして要素にアクセスすることができますが、リストと違って要素を変更することはできません。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

レンジと`for`を使って、`0`から`4`までの数字を順番に表示してください。

問題2

始まりと終わりを指定したレンジを使って、`10`から`14`までの数字を順番に表示してください。

問題3

レンジと`for`を使って、`1`から`100`までの整数の合計を求め、表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
for number in range(5):
    print(number)
```

終わりとして指定した`5`は含まれないため、`0`から`4`までが表示されます。

問題2

模範解答

```python
for number in range(10, 15):
    print(number)
```

始まりを`10`、終わりを`15`にします。終わりの`15`は含まれません。

問題3

模範解答

```python
total = 0

for number in range(1, 101):
    total += number

print(total)
```

`100`まで含めるため、終わりには`101`を指定します。合計は`5050`です。

LESSON 18 / ループ

18 多重ループ

Web版で読む ↗

年を順番に表示する

PYTHON

```
for year in range(2000, 2200):
    print(year)
```

レンジと`for`を使って2000年から2199年までを表示します。

文字列に値を埋め込む

PYTHON

```
for year in range(2000, 2200):
    print(f"{year}年")
```

`f" "`を使えば、文字列の中に値を埋め込めます。たとえば、`year`の値が`2000`のとき、`f"{year}年"`は`"2000年"`という文字列になります。

ループを重ねる

PYTHON

```
for year in range(2000, 2200):
    for month in range(1, 13):
        print(f"{year}年{month}月")
```

`if`だけでなく、`while`や`for`もネストすることができます。

外側に`year`（年）のループ、内側に`month`（月）のループを書けば、`2000年1月`, `2000年2月`, …, `2199年12月`と順番にすべての年と月をループすることができます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

外側のループで`2025`年から`2026`年、内側のループで`1`月から`3`月を処理し、`2025年1月`のような形ですべての組み合わせを表示してください。

問題2

外側のループで`1`から`2`、内側のループで`1`から`3`を処理し、`行1-列1`のような形ですべての組み合わせを表示してください。

問題3

`1`から`3`までの掛け算表を作ります。二つのループを重ね、`2 * 3 = 6`のような形ですべての組み合わせを表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
for year in range(2025, 2027):
    for month in range(1, 4):
        print(f"{year}年{month}月")
```

年を一つ取り出すたびに、内側の月のループが最初から最後まで実行されます。全部で`2 * 3`個の組み合わせが表示されます。

問題2

模範解答

```python
for row in range(1, 3):
    for column in range(1, 4):
        print(f"行{row}-列{column}")
```

二つのループ変数を一つの文字列に埋め込みます。行と列の組み合わせが6通り表示されます。

問題3

模範解答

PYTHON

```
for x in range(1, 4):
    for y in range(1, 4):
        print(f"{x} * {y} = {x * y}")
```

外側の数ごとに、内側の`1`から`3`をすべて掛けます。計算式と結果は、どちらも文字列へ埋め込めます。

LESSON 19 / 関数

19 関数を使う

Web版で読む ↗

これまで使った関数

```python
a = [2, 3, 5]
b = len(a)
print(b)
```

まとまった処理に名前を付けて、それを実行することができます。このような機能を**関数（かんすう, function）**と言います。

これまでにもすでにいくつかの関数を使ったことがあります

`len`はリストを受け取り、その長さ（要素数）を返す関数です。`len(a)`のように`len`にリスト`a`を渡すと、そのリストの要素数である`3`が得られます。

`print`は受け取った値を出力（表示）する関数です。

絶対値を求める

```python
a = -2
b = abs(a)
print(b)
```

`abs`は絶対値を返す関数です。`-2`の絶対値は`2`です。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

`abs`を使って、`-7`の絶対値を表示してください。

問題2

`[2, 4, 6, 8]`の要素数を求めて表示してください。

問題3

数直線上の`-4`と`7`の距離を、`abs`を使って求め、表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
print(abs(-7))
```

絶対値を求めると、`-7`から符号を取り除いた`7`が得られます。

問題2

模範解答

```python
numbers = [2, 4, 6, 8]
print(len(numbers))
```

`len`関数から要素数の`4`が得られます。

問題3

模範解答

```python
print(abs(-4 - 7))
```

二つの位置の差は`-4 - 7`で`-11`です。その絶対値を求めると、距離の`11`が得られます。

LESSON 20 / 関数

20 関数を作る

Web版で読む ↗

関数を定義する

```python
def square(x):
    return x * x

a = square(3)
print(a)
```

`def`を使って、自分で関数を定義することができます。関数が行う処理は、インデントされたブロックに記述します。

受け取った数の2乗を返す関数`square`は上記のように書くことができます。受け取る数を`x`で表し（`x`でなくても好きな名前を付けられます）、`return`で結果を返します。ここでは`x * x`で2乗した数を返しています。

関数を定義するときに、受け取る値に付ける名前を**パラメータ（parameter）**と言います。この関数では`x`がパラメータです。関数を呼び出すときに渡す値を**引数（ひきすう, argument）**、関数が返す値を**戻り値（もどりち, return value）**と言います。また、関数に引数を渡して実行することを、関数を**呼び出す（よびだす, call）**と言います。

累乗を計算する

```python
a = 2 * 2 * 2 * 2 * 2
print(a)
```

2の5乗を計算していますが、掛け算を何度も書くのは大変です。

関数の形を考える

```python
a = power(2, 5)
print(a)
```

これを計算する関数`power`を作ってみます。`power`には二つのパラメータが必要です。関数には好きな個数のパラメータを定義できます。パラメータが0個の関数を作ることもできます。

> **NOTE**
>
> この時点ではまだ`power`を作っていないため、上のコードだけを実行するとエラーになります。

power関数を実装する

```python
def power(x, y):
    result = 1
    for i in range(y):
        result *= x
    return result

a = power(2, 5)
print(a)
```

関数`power`はこのように書くことができます。

使わないループ変数

```python
def power(x, y):
    result = 1
    for _ in range(y):
        result *= x
    return result

a = power(2, 5)
print(a)
```

> **TIP**
>
> 元の`power`のコードではループ変数`i`がどこでも使われていませんでした。使われない変数は`_`（アンダースコア）で書くのが一般的です。

組み込みの関数を使う

```python
a = pow(2, 5)
print(a)
```

実は、Pythonには元々`pow()`という関数が用意されているので、自分で作る必要はありません。

```python
a = pow(2, 2.5)
print(a)
```

この`pow`はさっき自分で書いた`power`よりも汎用的な実装がされており、2.5乗のような小数乗を計算することもできます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

整数を一つ受け取り、その数を2乗して返す関数`square`を定義してください。`square(4)`の結果を表示してください。

問題2

長方形の幅と高さを整数で受け取り、面積を返す関数`rectangleArea`を定義してください。幅`8`、高さ`5`の面積を表示してください。

問題3

整数`x`の`y`乗を返す関数`power`を定義してください。`power(3, 4)`の結果を表示してください。

問題4

言語に用意されている累乗の関数を使って、`5`の3乗を表示してください。

問題5

`1`から受け取った整数`n`までをすべて掛けた値を返す関数`factorial`を定義してください。`factorial(5)`の結果を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
def square(x):
    return x * x

print(square(4))
```

パラメータ`x`を`x * x`で2乗し、`return`で結果を返します。`square(4)`の戻り値は`16`です。

問題2

模範解答

```python
def rectangleArea(width, height):
    return width * height

print(rectangleArea(8, 5))
```

幅と高さをパラメータとして受け取り、その積を戻り値にします。同じ関数へ別の幅と高さを渡せば、ほかの長方形にも使えます。

問題3

模範解答

```python
def power(x, y):
    result = 1
    for _ in range(y):
        result *= x
    return result

print(power(3, 4))
```

`result`を`1`から始め、`x`を`y`回掛けます。`3`を4回掛けた結果の`81`が返されます。

問題4

模範解答

```python
print(pow(5, 3))
```

`pow`に底と指数を渡します。結果は`125`です。

問題5

模範解答

```python
def factorial(n):
    result = 1
    for number in range(1, n + 1):
        result *= number
    return result

print(factorial(5))
```

累乗する関数と同じように、ループで`result`へ値を掛けていきます。`1 * 2 * 3 * 4 * 5`の結果である`120`が返されます。

LESSON 21 / 関数

21 第一級関数

Web版で読む ↗

関数を変数に代入する

PYTHON

```
a = abs(-2)
print(a)
```

前に絶対値を計算する関数`abs`を使いました。

PYTHON

```
a = abs
b = a(-2)
print(b)
```

Pythonでは、`abs(-2)`のように関数に値を渡して呼び出すだけでなく、`abs`という関数そのものを変数に代入することができます。変数に代入された関数は`( )`で値を渡して呼び出すことができます。

ここでは関数`abs`を変数`a`に代入し、`a(-2)`で呼び出しています。

PYTHON

```
def square(x):
    return x * x

a = square
b = a(3)
print(b)
```

自分で定義した関数も、同じように変数に代入できます。

このように、Pythonでは関数を数値や文字列と同じように値として扱えます。このような性質を持つ関数を**第一級関数（だいいっきゅうかんすう, first-class function）**と呼びます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

整数を2乗する関数`square`を定義し、その関数自体を変数`f`に代入してください。`f(5)`の結果を表示してください。

問題2

絶対値を求める関数`abs`を変数`f`に代入し、`f(-8)`の結果を表示してください。

問題3

整数を2倍する関数`double`と、整数を1増やす関数`add_one`を定義してください。二つの関数を一つのリストにまとめ、それぞれに`5`を渡した結果を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
def square(x):
    return x * x

f = square
print(f(5))
```

関数を呼び出す`( )`を付けずに`square`と書くと、関数そのものを変数へ代入できます。`f`からも同じ関数を呼び出せます。

問題2

模範解答

```python
f = abs
print(f(-8))
```

組み込みの関数も変数へ代入できます。変数`f`に`-8`を渡して呼び出すと、絶対値の`8`が返されます。

問題3

模範解答

```python
def double(x):
    return x * 2

def add_one(x):
    return x + 1

functions = [double, add_one]
print(functions[0](5))
print(functions[1](5))
```

関数は値として扱えるため、リストの要素にもできます。結果は`10`と`6`です。

LESSON 22 / 関数

22 高階関数

Web版で読む ↗

関数を引数に渡す

PYTHON

```
def square(x):
    return x * x

a = twice(square, 3)
print(a)
```

関数の引数に関数を渡すこともできます。

たとえば、同じ関数を2回繰り返して実行する関数`twice`を考えます。そうすると、`twice`に繰り返したい関数`square`を渡したくなるはずです。これが、関数に関数を渡すという意味です。

上記の例では、まず`square(3)`で`3 * 3`が実行されて`9`が得られ、さらに`square(9)`で`9 * 9`が実行されて`81`が得られます。

このように、関数を引数として受け取る、または関数を戻り値として返す関数を**高階関数（こうかいかんすう, higher-order function）**と言います。

> NOTE
>
> この時点ではまだ`twice`を作っていないため、上のコードだけを実行するとエラーになります。

高階関数を実装する

```python
def twice(f, x):
    y = f(x)
    z = f(y)
    return z

def square(x):
    return x * x

a = twice(square, 3)
print(a)
```

関数`twice`はこのように実装できます。

```python
def twice(f, x):
    return f(f(x))

def square(x):
    return x * x

a = twice(square, 3)
print(a)
```

より簡潔に書くこともできます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

受け取った関数を2回適用する高階関数`twice`と、整数を2乗する関数`square`を定義してください。`twice(square, 3)`の結果を表示してください。

問題2

受け取った整数を`1`増やす関数`addOne`と、受け取った関数を2回適用する高階関数`twice`を定義してください。

`twice(addOne, 3)`の結果を表示してください。

問題3

受け取った関数を3回適用する高階関数`thrice`を定義してください。`addOne`を渡して、`thrice(addOne, 10)`の結果を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
def twice(f, x):
    return f(f(x))

def square(x):
    return x * x

print(twice(square, 3))
```

`twice`は関数`f`を引数として受け取り、その関数を2回呼び出します。`3`を2乗した`9`がもう一度2乗されるため、`81`が表示されます。

問題2

模範解答

```python
def twice(f, x):
    return f(f(x))

def addOne(x):
    return x + 1

print(twice(addOne, 3))
```

最初の`addOne`で`3`が`4`になり、次の`addOne`で`5`になります。関数を引数として渡すことで、`twice`は異なる関数にも使えます。

問題3

模範解答

```python
def thrice(f, x):
    return f(f(f(x)))

def addOne(x):
    return x + 1

print(thrice(addOne, 10))
```

`f`の戻り値を次の`f`へ渡す処理を3回重ねます。`10`に`1`が3回足され、`13`が表示されます。

LESSON 23 / 関数

23 map関数

Web版で読む ↗

mapで各要素を変換する

PYTHON

```
def square(x):
    return x * x

a = [2, 3, 5]
b = map(square, a)
for x in b:
    print(x)
```

`map`は代表的な高階関数です。リストの各要素に関数を適用した結果を返します。

たとえば、`map`に`square`を渡すと、第2引数に渡したリストのすべての要素を2乗した結果を返します。

PYTHON

```
def square(x):
    return x * x

a = [2, 3, 5]
b = map(square, a)
print(b)
```

> NOTE
>
> `map`が返すのはリストではありません。レンジと同じように特別な`map`オブジェクトです。

```python
def square(x):
    return x * x

a = [2, 3, 5]
b = map(square, a)
c = list(b)
print(c)
```

`list`関数を使って`map`オブジェクトをリストに変換することができます。リストは`print`で中身を表示することができます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

`[2, 3, 5]`の各要素を2乗する関数`square`を定義し、`map`ですべての要素を変換した結果を表示してください。

問題2

整数を`1`増やす関数`add_one`を定義し、`map`を使って`[1, 4, 7]`の各要素に適用した結果を表示してください。

問題3

点数`55`、`70`、`88`のすべてに`5`点を加えた新しいリストを`map`で作ってください。さらに、変換後の点数をすべて足し、その合計を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
def square(x):
    return x * x

numbers = [2, 3, 5]
result = list(map(square, numbers))
print(result)
```

`map`は各要素に`square`を適用します。結果は`[4, 9, 25]`です。`map`オブジェクトは`list`でリストに変換しています。

問題2

模範解答

```python
def add_one(x):
    return x + 1

numbers = [1, 4, 7]
result = list(map(add_one, numbers))
print(result)
```

`map`へ関数`add_one`を渡すと、すべての要素に同じ処理が適用されます。結果は`[2, 5, 8]`です。

問題3

模範解答

```python
def adjust_score(score):
    return score + 5

scores = [55, 70, 88]
adjusted = list(map(adjust_score, scores))

total = 0
for score in adjusted:
    total += score

print(total)
```

`map`で各点数を変換してから、これまでに学んだループで合計します。変換後は`60`、`75`、`93`なので、合計は`228`です。

LESSON 24 / 関数

24 ラムダ式

Web版で読む ↗

ラムダ式で関数を作る

```python
square = lambda x: x * x

a = square(3)
print(a)
```

ラムダ式（ラムダしき, lambda expression）を使えば、その場で関数を作ることができます。このコードでは、`lambda`でパラメータ`x`を2乗して返す関数を作り、それを`square`に代入しています。

mapにラムダ式を渡す

`map`のためにわざわざ関数を定義するのは面倒です。

```python
a = [2, 3, 5]
b = map(lambda x: x * x, a)
c = list(b)
print(c)
```

ラムダ式を使うと、`map`で行いたい処理を直接渡すことができます。

```python
a = [2, 3, 5]
b = map(lambda x: x + 1, a)
c = list(b)
print(c)
```

2乗ではなく、1足したいならこのようになります。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

整数を2乗して返す関数を、ラムダ式で作って変数`square`に代入してください。`square(4)`の結果を表示してください。

問題2

ラムダ式を`map`へ直接渡し、`[1, 4, 7]`の各要素に`1`を足した結果を表示してください。

問題3

点数`55`、`70`、`88`のすべてに`5`点を加えた新しいリストを、ラムダ式を`map`へ直接渡して作ってください。さらに、変換後の点数をすべて足し、その合計を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

```python
square = lambda x: x * x
print(square(4))
```

名前を付けて関数を定義する代わりに、その場で関数を作って変数へ代入できます。`square(4)`の結果は`16`です。

問題2

模範解答

```python
numbers = [1, 4, 7]
result = list(map(lambda x: x + 1, numbers))
print(result)
```

短い変換処理なら、別に関数を定義せず、その場で関数を作って`map`へ渡せます。結果は`[2, 5, 8]`です。

問題3

模範解答

```python
scores = [55, 70, 88]
adjusted = list(map(lambda score: score + 5, scores))

total = 0
for score in adjusted:
    total += score

print(total)
```

`map`へ短い変換処理を直接渡してから、これまでに学んだループで合計します。変換後は`60`、`75`、`93`なので、合計は`228`です。

LESSON 25 / クラス

25 クラス

Web版で読む ↗

情報がばらばらになる問題

```python
names = ["Mary", "Mike", "Tom"]
scores = [65, 90, 85]

for i in range(len(names)):
    print(names[i], scores[i])
```

複数人の名前とテストの点数を扱うには、二つのリストが必要になります。しかし、名前と点数をばらばらに扱うことになりわかりづらいです。

クラスで情報をまとめる

```python
class Student:
    name = ""
    score = 0

student = Student()
student.name = "Mary"
student.score = 65

print(student.name)
print(student.score)
```

クラス（class）を使えば、名前と点数のような、複数種類の値を一つにまとめて扱うことができます。

ここでは`Student`（生徒）クラスを作ります。クラスの中に書いた`name`と`score`は、`Student`クラスが持つ**クラス属性（class attribute）**です。このように値を表す属性を**データ属性（データぞくせい, data attribute）**と言います。

`Student()`で`Student`のオブジェクトを作って変数`student`に代入します。`student.name = "Mary"`のように代入すると、`student`オブジェクト自身が持つ**インスタンス属性（instance attribute）**が作られます。`student.name`や`student.score`のように、オブジェクトを格納した変数名と属性名を`.`でつなぐことで、その属性にアクセスできます。

```python
class Student:
    name = ""
    score = 0

student = Student()
student.name = "Mary"
student.score = 65

print(f"{student.name}: {student.score}点")
```

`Mary: 65点`のような形式で名前と点数を出力するには`print(f"{student.name}: {student.score}点")`と書けます。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

名前と点数を持つ`Student`クラスを作ってください。オブジェクトを一つ作り、名前に`"Mary"`、点数に`65`を代入して、両方を表示してください。

問題2

本の題名とページ数を持つ`Book`クラスを作ってください。オブジェクトを一つ作り、題名に`"Python Guide"`、ページ数に`240`を代入して表示してください。

問題3

幅と高さを持つ`Rectangle`クラスを作ってください。幅`8`・高さ`5`のオブジェクトと、幅`3`・高さ`9`のオブジェクトを作り、それぞれの面積を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

PYTHON

```
class Student:
    name = ""
    score = 0

student = Student()
student.name = "Mary"
student.score = 65

print(student.name)
print(student.score)
```

クラスに名前と点数をまとめ、作成したオブジェクトのそれぞれの値へ`.`を使ってアクセスします。

問題2

模範解答

```python
class Book:
    title = ""
    pages = 0

book = Book()
book.title = "Python Guide"
book.pages = 240

print(book.title)
print(book.pages)
```

関連する題名とページ数を一つの`Book`オブジェクトにまとめています。

問題3

模範解答

```python
class Rectangle:
    width = 0
    height = 0

first = Rectangle()
first.width = 8
first.height = 5

second = Rectangle()
second.width = 3
second.height = 9

print(first.width * first.height)
print(second.width * second.height)
```

同じクラスから複数のオブジェクトを作ると、それぞれが別の幅と高さを持てます。面積は順に`40`と`27`です。

LESSON 26 / クラス

26 メソッドと初期化

Web版で読む ↗

メソッドを定義する

PYTHON

```
class Student:
    name = ""
    score = 0

    def show(self):
        print(f"{self.name}: {self.score}点")

student = Student()
student.name = "Mary"
student.score = 65

student.show()
```

クラスにはデータ属性だけでなく、クラス専用の関数である**メソッド（method）**を定義することもできます。

インスタンスメソッドの第1パラメータには`self`を使い、自分自身を表します。

> NOTE
>
> 第1パラメータを`self`と名付けるのはPythonの慣習です。別の名前でも動作しますが、通常は`self`を使います。

ここでは、`Mary: 65点`のような形式で名前と点数を出力するメソッド`show`を実装しています。

`student.show()`とすると、`show`の第1パラメータ`self`に`student`に格納されている`Student`オブジェクトが自動的に渡されます。

作成と同時に初期化する

```python
class Student:
    name = ""
    score = 0

    def __init__(self, name, score):
        self.name = name
        self.score = score

    def show(self):
        print(f"{self.name}: {self.score}点")

student = Student("Mary", 65)
student.show()
```

`__init__`は特別なメソッドで、`Student( )`で`Student`オブジェクトを作成するときに呼ばれます。`__init__`にパラメータを定義すると、`Student("Mary", 65)`のように引数を渡し、オブジェクト作成と同時に指定された値でデータ属性を初期化できます。

そうすれば、後から`student.name = "Mary"`のようにして値をセットしなくても、`Student("Mary", 65)`のようにして初期化できます。

`__init__`メソッドのことを**イニシャライザ（initializer）**とも言います。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

名前と点数を持つ`Student`クラスに、`名前: 点数点`の形で表示する`show`メソッドを定義してください。名前が`"Mary"`、点数が`65`のオブジェクトで`show`を呼び出してください。

問題2

題名とページ数を作成時に受け取る`Book`クラスを定義してください。また、`題名: ページ数ページ`の形で表示する`show`メソッドも定義し、`"Python Guide"`、`240`で作ったオブジェクトから呼び出してください。

問題3

幅と高さを作成時に受け取る`Rectangle`クラスを定義してください。面積を返す`area`メソッドを作り、幅`8`・高さ`5`の長方形の面積を表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

PYTHON

```
class Student:
    name = ""
    score = 0

    def show(self):
        print(f"{self.name}: {self.score}点")

student = Student()
student.name = "Mary"
student.score = 65
student.show()
```

メソッドの中では、`self`を使って、そのメソッドを呼び出したオブジェクト自身の値へアクセスします。

問題2

模範解答

```python
class Book:
    title = ""
    pages = 0

    def __init__(self, title, pages):
        self.title = title
        self.pages = pages

    def show(self):
        print(f"{self.title}: {self.pages}ページ")

book = Book("Python Guide", 240)
book.show()
```

`__init__`で作成時の引数を受け取り、それぞれの値を初期化しています。

問題3

模範解答

```python
class Rectangle:
    width = 0
    height = 0

    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

rectangle = Rectangle(8, 5)
print(rectangle.area())
```

初期化した幅と高さを`area`メソッドで掛け、結果を戻り値にします。`rectangle.area()`は`40`を返します。

LESSON 27 / クラス

27 クラスとコレクション

Web版で読む ↗

オブジェクトをリストで管理する

PYTHON

```
class Student:
    name = ""
    score = 0

    def __init__(self, name, score):
        self.name = name
        self.score = score

    def show(self):
        print(f"{self.name}: {self.score}点")

students = [Student("Mary", 65), Student("Mike", 90), Student("Tom", 85)]
for student in students:
    student.show()
```

クラスを使うことで、複数の生徒の情報を一つのリストで管理することができます。

リストのメソッド

```python
a = [2, 1, 7, 5]
print(a)

a.append(3)
print(a)

a.remove(1)
print(a)

a.sort()
print(a)
```

リストもクラスなのでメソッドを持っています。

たとえば、`append`メソッドを使えば要素を追加することが、`remove`メソッドを使えば要素を削除することができます。また、`sort`メソッドを使えば、要素を小さい順に並べ替えることもできます。

文字列のメソッド

```python
a = "2,3,5,7"
b = a.split(",")
c = list(map(int, b))
print(c)
```

文字列もクラスなのでメソッドを持っています。

たとえば、`split`メソッドを使えば文字列を分割することができます。

ここでは、`"2,3,5,7"`という文字列を`,`で分割しています。しかし、分割して得られたリストの要素は`["2", "3", "5", "7"]`のように文字列になっているので、これをさらに`int`関数を使って整数に変換しています。

PRACTICE

演習問題

コードを自分で書いて実行してから、模範解答と解説を確認しましょう。

問題1

名前と点数を作成時に受け取り、`show`メソッドで表示できる`Student`クラスを定義してください。二人分のオブジェクトを一つのリストにまとめ、すべての生徒の`show`を呼び出してください。

問題2

`[4, 1, 7, 3]`に`5`を追加し、`1`を削除してから、小さい順に並べ替えて表示してください。

問題3

文字列`"10,20,30,40"`を`,`で分割し、すべての要素を整数に変換してください。変換した整数の合計を求めて表示してください。

ANSWERS

模範解答と解説

問題1

模範解答

PYTHON

```
class Student:
    name = ""
    score = 0

    def __init__(self, name, score):
        self.name = name
        self.score = score

    def show(self):
        print(f"{self.name}: {self.score}点")

students = [Student("Mary", 65), Student("Mike", 90)]
for student in students:
    student.show()
```

オブジェクトもリストの要素にできます。ループで一つずつ取り出すと、それぞれのオブジェクトに対してメソッドを呼び出せます。

問題2

模範解答

```python
numbers = [4, 1, 7, 3]
numbers.append(5)
numbers.remove(1)
numbers.sort()
print(numbers)
```

追加、削除、並べ替えを順番に行います。最後に表示される内容は`[3, 4, 5, 7]`です。

問題3

模範解答

```python
text = "10,20,30,40"
parts = text.split(",")
numbers = list(map(int, parts))

total = 0
for number in numbers:
    total += number

print(total)
```

`split`の結果は文字列なので、整数へ変換してから足します。4個の整数の合計である`100`が表示されます。