

PROPLA

BETA COURSE

Swift

プログラミング入門

Appleプラットフォームのアプリ開発で使われる、安全性を重視した言語です。

27 LESSONS + PRACTICE

プロプラ

このコースについて

Swiftとは

Swiftは、iPhoneやMacなどのAppleプラットフォーム向けアプリ開発で使われるプログラミング言語です。実行前に型の問題を見つけやすく、安全性と実行性能を重視して設計されています。

よく使われる分野	iPhone、iPad、Macなどのアプリや、Appleプラットフォーム向けのソフトウェア。
書き方の特徴	値の型を確認しながらコードを組み立てます。型を省略できる場面でも、変数の型は明確に決まります。
試せる環境	Swift 6. Xcode Playground、Swift Playgrounds、またはターミナルで試せます。

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/swift/>

目次

01 はじめの一步	5
02 式と文字列	10
03 四則演算	13
04 計算の順序	17
05 値の比較	20
06 変数	23
07 変数名と複数の変数	27
08 変数に保存できる値	31
09 再代入	36
10 条件分岐	40
11 elseとelse if	44
12 ifのネスト	48
13 ループ	53
14 配列	57
15 配列とwhile	61
16 配列とfor	66
17 レンジ	70
18 多重ループ	74
19 関数を使う	78
20 関数を作る	81

21 第一級関数	88
22 高階関数	92
23 mapメソッド	97
24 クロージャ	101
25 クラス	105
26 メソッドと初期化	111
27 クラスとコレクション	117

LESSON 01 / はじめに

01 はじめの一步

[Web版で読む](#) ➤

これは、プログラミングの初心者がSwift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

```
SWIFT  
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。

```
SWIFT  
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。

```
SWIFT  
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

```
error: expected ')' in expression list
```

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

```
SWIFT  
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

```
SWIFT  
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

```
SWIFT  
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

SWIFT

```
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。Swiftでは（Swift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

SWIFT

```
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。Swift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。

SWIFT

```
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。

```
SWIFT

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。

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

```
SWIFT  
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

```
SWIFT  
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

```
SWIFT  
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

SWIFT

```
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

SWIFT

```
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

SWIFT

```
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

SWIFT

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

割り算には、`÷`ではなく`/`（スラッシュ）を使います。

Swiftでは、今の6と2はどちらも整数なので、整数の割り算が行われて3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

```
SWIFT  
  
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

```
SWIFT  
  
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。この割り算の結果は4と表示されます。

問題3

模範解答

SWIFT

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

それぞれの場면을、掛け算、引き算、割り算の式で表します。結果は順に600、400、9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

SWIFT

```
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

SWIFT

```
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。

SWIFT

```
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。

SWIFT

```
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、Swift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

```
SWIFT  
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

```
SWIFT  
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

```
SWIFT  
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

SWIFT

```
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。

SWIFT

```
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

SWIFT

```
print(2 == 2)
```

Swift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。

SWIFT

```
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

SWIFT

```
print(!(2 < 3))
```

`2 < 3`の結果は`true`（真）です。その前に`!`を付けると、真と偽が反対になり、`false`（偽）が出力されます。`!`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

```
SWIFT  
  
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

```
SWIFT  
  
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

```
SWIFT  
  
print(!(10 > 20))
```

10 > 20は偽ですが、!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

SWIFT

```
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

SWIFT

```
var 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。`var 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。

Swiftでは、後から値を変更できる変数を作るときに `var` を書きます。

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

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

SWIFT

```
var 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

SWIFT

```
var 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

SWIFT

```
var 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

SWIFT

```
var 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。

SWIFT

```
var 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

SWIFT

```
var r = 5.0
var base = r * r * 3.14
var h = 8.0
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。

Swiftでは、整数と小数をそのまま掛けることはできません。ここでは`5.0`や`8.0`と書き、すべて小数を表す`Double`という型にそろえています。

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

SWIFT

```
var radius = 4.0
var base = radius * radius * 3.14
var height = 6.0
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

SWIFT

```
var width = 8
var height = 5
var 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

SWIFT

```
var japanese = 72
var math = 88
var english = 80
var 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

```
SWIFT

var 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

```
SWIFT

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

比較の結果（trueまたはfalse）も代入できます。

```
SWIFT

var 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	Swift
整数	Int
小数	Double
文字列	String
ブール値	Bool

Swiftでは、最初に代入した値から変数の型が決まります。たとえば`var a = "Hello"`と書くと、`a`はString型になります。その後で同じ`a`に整数やブール値を代入することはできません。

型を自分で書く場合は、次のように変数名の後へ、型名を付けます。

SWIFT

```
var message: String = "Hello"
var age: Int = 12
var rate: Double = 3.14
var passed: Bool = true
```

この先のSwiftの関数やクラスでは、パラメータやデータの型をこの方法で書きます。

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

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。それぞれの変数に型も明記してください。

ANSWERS

模範解答と解説

問題1

模範解答

SWIFT

```
var 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

SWIFT

```
var 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

SWIFT

```
var name: String = "Kai"  
var age: Int = 14  
var height: Double = 158.5  
var passed: Bool = 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。String、Int、Double、Boolと型を明記しています。

LESSON 09 / 変数

09 再代入

[Web版で読む](#)

値を上書きする

SWIFT

```
var 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

SWIFT

```
var 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

```
SWIFT
```

```
var 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

```
SWIFT  
  
var 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

```
SWIFT  
  
var 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

SWIFT

```
var 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

SWIFT

```
var 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。

実行する処理を{ }の中に書きます。{ }の中はインデント（字下げ）すると、範囲が分かりやすくなります。

複数の処理をまとめる

SWIFT

```
var 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")
}
```

ブロックの中にコードを複数行書くと、その範囲全体がifの対象となります。

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

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

SWIFT

```
var 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

SWIFT

```
var 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

SWIFT

```
var 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else if

[Web版で読む](#) ➤

条件を満たさない場合

SWIFT

```
var 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

SWIFT

```
var age = 12
if age < 0 {
    print("エラーです")
} else 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、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else ifを使います。else if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、else ifのブロックが実行されます。

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。

SWIFT

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

else if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

```
SWIFT

var 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

```
SWIFT

var score = 75
if score > 80 {
    print("A")
} else if 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

SWIFT

```
var age = 14
if age < 6 {
    print(0)
} else if 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

```
SWIFT

var gender = "male"
var 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

SWIFT

```
var weather = "rainy"
var 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

SWIFT

```
var member = "yes"
var 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

SWIFT

```
var light = "green"
var 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

SWIFT

```
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。

SWIFT

```
var 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

SWIFT

```
var 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

```
SWIFT

var 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

```
SWIFT

var 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

```
SWIFT

var number = 1
var total = 0

while number < 11 {
    total += number
    number += 1
}

print(total)
```

numberを1ずつ増やししながら、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 / 配列

14 配列

[Web版で読む](#)

複数の変数で管理する問題

SWIFT

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

var 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

SWIFT

```
var 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。**配列 (array)** 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]` のようにして配列を作ることができます。`a` には `65`, `90`, `85` の3個の値を持つ配列が格納されます。

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、配列では値を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、それで得られる配列の一つ一つの値を**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を一つの配列にまとめて変数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"を一つの配列にまとめ、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を一つの配列にまとめ、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

SWIFT

```
var 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

SWIFT

```
var 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

SWIFT

```
var scores = [72, 88, 80]
var 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 / 配列

15 配列とwhile

[Web版で読む](#) ➤

要素を更新する

```
SWIFT

var 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

```
SWIFT

var a = [65, 90, 85]

var i = 0
while i < 3 {
    print(a[i])
    i += 1
}
```

配列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

SWIFT

```
var a = [65, 90, 85]
print(a.count)
```

`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。`count`を使えば配列の長さ（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。

SWIFT

```
var a = [65, 90, 85]
var n = a.count

var i = 0
while i < n {
    print(a[i])
    i += 1
}
```

要素数を使えば、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。

SWIFT

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

var i = 0
while i < a.count {
    print(a[i])
    i += 1
}
```

要素数を変数に代入せず、条件式に直接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

点数65、90、85をまとめた配列の最初の要素を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をまとめた配列があります。`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

SWIFT

```
var 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

SWIFT

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

var i = 0
while i < numbers.count {
    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。`numbers.count`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

SWIFT

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

var i = 0
while i < scores.count {
    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 / 配列

16 配列とfor

[Web版で読む](#) ➤

forで要素を取り出す

SWIFT

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

for b in a {
    print(b)
}
```

whileとループ変数を使って配列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、配列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。

SWIFT

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

for score in scores {
    print(score)
}
```

TIP

配列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配列なのか、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

SWIFT

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

var total = 0
for score in scores {
    total += score
}

print(total / scores.count)
```

`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、ループと配列を使えば、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をまとめた配列から、`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

```
SWIFT

var 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

```
SWIFT

var numbers = [3, 5, 7, 9]
var 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

SWIFT

```
var scores = [45, 72, 88, 59]
var 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。

```
SWIFT

var a = 0
while a < 10000 {
    print(a)
    a += 1
}
```

0から9999までの値をすべて配列に書くのは大変です。

レンジを使う

```
SWIFT

let numbers = 0..<10000

for number in numbers {
    print(number)
}
```

Swiftでは、`0..<10000`とすると、0から9999までの値を表す**レンジ（Range）**を作ることができます。レンジからも、配列と同じようにforで値を一つずつ取り出してループできます。

```
SWIFT

for a in 0..<10000 {
    print(a)
}
```

`in`の右側に直接`0..<10000`と書くこともできます。

始まりを指定する

```
SWIFT

for a in 10..<20 {
    print(a)
}
```

10..<20のようにして、レンジの始まりと終わりを指定することもできます。

範囲の扱い方

```
SWIFT

let r = 10..<20
print(r.lowerBound)
// rの値を変更することはできない
```

Swiftのレンジから始まりの値を取り出すにはlowerBoundを使えます。レンジは範囲を表す値なので、配列のように途中の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

```
SWIFT

for number in 0..<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

```
SWIFT

for number in 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

SWIFT

```
var total = 0

for number in 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

```
SWIFT

for year in 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

```
SWIFT

for year in 2000..<2200 {
    print("\(year)年")
}
```

文字列の中に\ () を書くと、値を埋め込みます。たとえば、yearの値が2000のとき、"\(year)年"は"2000年"という文字列になります。

ループを重ねる

```
SWIFT

for year in 2000..<2200 {
    for month in 1..<13 {
        print("\(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

```
SWIFT

for year in 2025..<2027 {
    for month in 1..<4 {
        print("\(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

```
SWIFT

for row in 1..<3 {
    for column in 1..<4 {
        print("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

SWIFT

```
for x in 1..<4 {  
    for y in 1..<4 {  
        print("\(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みます。

LESSON 19 / 関数

19 関数を使う

[Web版で読む](#) ➤

これまで使った関数

SWIFT

```
var a = [2, 3, 5]
var b = a.count
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。

これまでもすでにいくつかの関数を使ったことがあります

Swiftでは、配列の長さはcountという値から得られます。printは受け取った値を出力（表示）する関数です。

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

SWIFT

```
var a = -2
var 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

```
SWIFT  
  
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

```
SWIFT  
  
var numbers = [2, 4, 6, 8]  
print(numbers.count)
```

countから要素数の4が得られます。

問題3

模範解答

```
SWIFT  
  
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

```
SWIFT

func square(_ x: Int) -> Int {
    return x * x
}

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

`func`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、`{ }`の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（もどり値, 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。

Swiftでは、パラメータ`x`の後に: `Int`を書いて整数を受け取ることを表し、`-> Int`で整数を返すことを表します。`x`の前の`_`は、呼び出すときの引数ラベルを省略し、`square(x: 3)`ではなく`square(3)`と書くための記号です。

累乗を計算する

```
SWIFT

var 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

SWIFT

```
var 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

SWIFT

```
func power(_ x: Int, _ y: Int) -> Int {
    var result = 1
    for i in 0..
```

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

SWIFT

```
func power(_ x: Int, _ y: Int) -> Int {  
    var result = 1  
    for _ in 0..        result *= x  
    }  
    return result  
}  
  
var a = power(2, 5)  
print(a)
```

TIP

元の`power`のコードでは、`for`で取り出した値をどこでも使っていませんでした。値を使わない場合は、`_`（ワイルドカードパターン）を書き、値を名前に割り当てずに繰り返すことができます。

組み込みの関数を使う

SWIFT

```
import Foundation  
  
var a = pow(2.0, 5.0)  
print(a)
```

実は、SwiftではFoundationに`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。

SWIFT

```
import Foundation  
  
var a = pow(2.0, 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

```
SWIFT

func square(_ x: Int) -> Int {
    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

```
SWIFT

func rectangleArea(_ width: Int, _ height: Int) -> Int {
    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

```
SWIFT

func power(_ x: Int, _ y: Int) -> Int {
    var result = 1
    for _ in 0..
```

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

```
SWIFT

import Foundation

print(pow(5.0, 3.0))
```

Foundationのpowに底と指数を渡します。結果は125.0です。

問題5

模範解答

SWIFT

```
func factorial(_ n: Int) -> Int {  
    var result = 1  
    for number in 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

SWIFT

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

前に絶対値を計算する関数`abs`を使いました。

SWIFT

```
let a: (Int) -> Int = abs
let b = a(-2)
print(b)
```

Swift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。

`(Int) -> Int`は、「整数を一つ受け取り、整数を一つ返す関数」という関数の型です。`abs`には複数の数値型に対応する形があるため、ここではどの型を使うか明示しています。

SWIFT

```
func square(_ x: Int) -> Int {
    return x * x
}

let a = square
let 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、Swift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`addOne`を定義してください。二つの関数を一つの配列にまとめ、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

```
SWIFT

func square(_ x: Int) -> Int {
    return x * x
}

let 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

```
SWIFT

let f: (Int) -> Int = 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

SWIFT

```
func double(_ x: Int) -> Int {  
    return x * 2  
}
```

```
func addOne(_ x: Int) -> Int {  
    return x + 1  
}
```

```
let functions: [(Int) -> Int] = [double, addOne]  
print(functions[0](5))  
print(functions[1](5))
```

関数は値として扱えるため、配列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

```
SWIFT

func square(_ x: Int) -> Int {
    return x * x
}

let 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

SWIFT

```
func twice(_ f: (Int) -> Int, _ x: Int) -> Int {  
    let y = f(x)  
    let z = f(y)  
    return z  
}  
  
func square(_ x: Int) -> Int {  
    return x * x  
}  
  
let a = twice(square, 3)  
print(a)
```

関数`twice`はこのように実装できます。

SWIFT

```
func twice(_ f: (Int) -> Int, _ x: Int) -> Int {  
    return f(f(x))  
}  
  
func square(_ x: Int) -> Int {  
    return x * x  
}  
  
let 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

```
SWIFT

func twice(_ f: (Int) -> Int, _ x: Int) -> Int {
    return f(f(x))
}

func square(_ x: Int) -> Int {
    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

SWIFT

```
func twice(_ f: (Int) -> Int, _ x: Int) -> Int {  
    return f(f(x))  
}  
  
func addOne(_ x: Int) -> Int {  
    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

SWIFT

```
func thrice(_ f: (Int) -> Int, _ x: Int) -> Int {  
    return f(f(f(x)))  
}  
  
func addOne(_ x: Int) -> Int {  
    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メソッド

[Web版で読む](#) ➤

mapで各要素を変換する

SWIFT

```
func square(_ x: Int) -> Int {  
    return x * x  
}  
  
let a = [2, 3, 5]  
let b = a.map(square)  
for x in b {  
    print(x)  
}
```

mapは代表的な高階メソッドです。配列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、元の配列のすべての要素を2乗した結果を返します。

SWIFT

```
func square(_ x: Int) -> Int {  
    return x * x  
}  
  
let a = [2, 3, 5]  
let b = a.map(square)  
print(b)
```

NOTE

mapが返すのは、各要素を変換した新しい配列です。

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

[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addOne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点を加えた新しい配列を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

SWIFT

```
func square(_ x: Int) -> Int {  
    return x * x  
}  
  
let numbers = [2, 3, 5]  
let result = numbers.map(square)  
print(result)
```

mapは各要素にsquareを適用します。結果は[4, 9, 25]です。

問題2

模範解答

SWIFT

```
func addOne(_ x: Int) -> Int {  
    return x + 1  
}  
  
let numbers = [1, 4, 7]  
let result = numbers.map(addOne)  
print(result)
```

mapへ関数addOne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

SWIFT

```
func adjustScore(_ score: Int) -> Int {  
    return score + 5  
}  
  
let scores = [55, 70, 88]  
let adjusted = scores.map(adjustScore)  
  
var 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 クロージャ

[Web版で読む](#) ➤

クロージャで関数を作る

SWIFT

```
let square = { (x: Int) -> Int in x * x }
```

```
let a = square(3)
```

```
print(a)
```

クロージャ (closure) 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、パラメータ `x` を2乗して返すクロージャを作り、それを `square` に代入しています。

mapにクロージャを渡す

`map` のためにわざわざ関数を定義するのは面倒です。

SWIFT

```
let a = [2, 3, 5]
```

```
let b = a.map { x in x * x }
```

```
print(b)
```

クロージャを使うと、`map` で行いたい処理を直接渡すことができます。

SWIFT

```
let a = [2, 3, 5]
```

```
let b = a.map { x in x + 1 }
```

```
print(b)
```

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、クロージャで作って変数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

クロージャを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点を加えた新しい配列を、クロージャを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

SWIFT

```
let square = { (x: Int) -> Int in 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

SWIFT

```
let numbers = [1, 4, 7]  
let result = numbers.map { x in x + 1 }  
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

SWIFT

```
let scores = [55, 70, 88]
let adjusted = scores.map { score in score + 5 }

var 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

SWIFT

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

for i in 0..
```

複数人の名前とテストの点数を扱うには、二つの配列が必要になります。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

SWIFT

```
class Student {
    var name: String = ""
    var score: Int = 0
}

let 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。 **Student** クラスは **name** と **score** の二つの **ストアドプロパティ (stored property)** を持っています。

`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` や `student.score` のように、オブジェクトを格納した変数名とプロパティ名を、でつなぐことで、そのプロパティにアクセスすることができます。

Swift では、 `name` が `String` 型、 `score` が `Int` 型であることを明記しています。

SWIFT

```
class Student {  
    var name: String = ""  
    var score: Int = 0  
}  
  
let student = Student()  
student.name = "Mary"  
student.score = 65  
  
print("\(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("\(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

SWIFT

```
class Student {  
    var name: String = ""  
    var score: Int = 0  
}
```

```
let 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

SWIFT

```
class Book {  
    var title: String = ""  
    var pages: Int = 0  
}  
  
let 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

SWIFT

```
class Rectangle {  
    var width: Int = 0  
    var height: Int = 0  
}  
  
let first = Rectangle()  
first.width = 8  
first.height = 5  
  
let 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

```
SWIFT

class Student {
    var name: String = ""
    var score: Int = 0

    func show() {
        print("\(self.name): \(self.score)点")
    }
}

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

student.show()
```

クラスにはスタッドプロパティだけでなく、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。メソッドの中では`self`が自分自身を指します。省略して`name`や`score`と書くこともできます。

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、`student`に格納されている`Student`オブジェクトに対して`show`が呼び出されます。

作成と同時に初期化する

SWIFT

```
class Student {  
    var name: String  
    var score: Int  
  
    init(_ name: String, _ score: Int) {  
        self.name = name  
        self.score = score  
    }  
  
    func show() {  
        print("\(self.name): \(self.score)点")  
    }  
}  
  
let student = Student("Mary", 65)  
student.show()
```

`init`は特別なメソッドのようなもので、`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ストアプロパティを初期化できます。

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`のことを**イニシャライザ (initializer)**と言います。

NOTE

正確には、Swiftの`init`はメソッドではなくイニシャライザです。書き方はメソッドに似ていますが、新しいインスタンスを初期化するための専用の仕組みです。

Swiftでは、初期値を持たないストアプロパティを作るときにも型が必要です。ここでは`name: String`と`score: Int`を宣言し、`init`の中で必ず初期値を代入しています。

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

```
SWIFT

class Student {
    var name: String = ""
    var score: Int = 0

    func show() {
        print("\(self.name): \(self.score)点")
    }
}

let 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

SWIFT

```
class Book {  
    var title: String  
    var pages: Int  
  
    init(_ title: String, _ pages: Int) {  
        self.title = title  
        self.pages = pages  
    }  
  
    func show() {  
        print("\(self.title): \(self.pages)ページ")  
    }  
}  
  
let 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

SWIFT

```
class Rectangle {  
    var width: Int  
    var height: Int  
  
    init(_ width: Int, _ height: Int) {  
        self.width = width  
        self.height = height  
    }  
  
    func area() -> Int {  
        return self.width * self.height  
    }  
}  
  
let 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](#) ➤

オブジェクトを配列で管理する

```
SWIFT

class Student {
    var name: String
    var score: Int

    init(_ name: String, _ score: Int) {
        self.name = name
        self.score = score
    }

    func show() {
        print("\(self.name): \(self.score)点")
    }
}

let 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配列で管理することができます。

配列のメソッド

SWIFT

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

a.append(3)
print(a)

a.removeAll { value in value == 1 }
print(a)

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

Swiftの配列はクラスではなく構造体ですが、同じようにメソッドを持っています。

要素を追加したり、指定した要素を削除したり、小さい順に並べ替えたりすることができます。使うメソッド名は言語によって異なります。

文字列のメソッド

SWIFT

```
let a = "2,3,5,7"
let b = a.split(separator: ",")
let c = b.compactMap { value in Int(value) }
print(c)
```

Swiftの文字列も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、分割して得られた配列の要素は`["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`を使って整数に変換しています。

NOTE

正確には、Swiftの`split`が返す配列の要素は`String`ではなく`Substring`です。`Substring`は文字列の一部を表す型で、この例では`String`と同じように`Int(value)`へ渡せます。

Swiftの`Int(value)`は、変換できない文字列を受け取る可能性があるため、`Int?`というOptionalの値を返します。`compactMap`を使うと、変換に成功した整数だけを新しい配列にまとめられます。これがSwiftでこの例を表す際に

追加で必要になる型の説明です。

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

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配列にまとめ、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

```
SWIFT

class Student {
    var name: String
    var score: Int

    init(_ name: String, _ score: Int) {
        self.name = name
        self.score = score
    }

    func show() {
        print("\(self.name): \(self.score)点")
    }
}

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

オブジェクトも配列の要素にできます。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

SWIFT

```
var numbers = [4, 1, 7, 3]
numbers.append(5)
numbers.removeAll { value in value == 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

SWIFT

```
let text = "10,20,30,40"
let parts = text.split(separator: ",")
let numbers = parts.compactMap { value in Int(value) }

var 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。