

一般教育演習

プログラミングで問題を解く：

集計から人工知能まで

瀧川 一学

工学部 情報理工学コース

残りの2回！

15. 7/26 授業のまとめと振り返り：この後どうする？

→ (解説) 最初の授業の前フリ(プログラムの役割など)を思い出し授業でやらなかったこと、今後のプログラミングとの付き合い方、python以外の言語、などを総括します。

16. 8/2 まとめNotebook作成日

→ (実習)

今日のお題：最終講義

- フォロアーアップQ & A
- この授業の評価について
- 授業のふりかえり：プログラミングとは？
- このあとどうする？
 - 授業でやらなかったこと
 - Python以外の言語
- 授業のまとめ

授業の目標 (シラバスより)

プログラミングとは何だろうか？ある人はそれは「アート」な知的活動だと言い、ある人はそれはコンピュータに魔法をかけるための「呪文」だと言う。今や私たちの社会や生活は、電子機器を動かしたりデータをやり取りする無数のコンピュータプログラムが支えている。プログラミングとはそれらのコンピュータプログラムを作る活動である。この入門授業では、実際にプログラムを作って動かすという実習とフォローアップディスカッションを通してプログラミングとは何かを理解することを目標とする。この「アート」な「呪文」に関心があるなら、文系理系やプログラミングの経験は全く問わない。

到達目標（シラバスより）

1. 「プログラミングとは何か？」について自分なりの答えが持てるようになる。
2. プログラミングに最低限必要な「コンピュータの仕組み」や「アルゴリズム」「データ構造」と言った初歩的知識を身につける。
3. 実際にプログラミング言語を用いて簡単なプログラムを作成するための基本的なスキルを身につけ実践することができる。
4. 今後、プログラミングを自ら習得していくために何をしていけば良いかの理解や知識を得る。
5. コンピュータプログラムが我々の生活社会の基盤にあることを意識し、品質の高い美しいプログラムを作り上げることの価値を議論できるようになる。

授業計画（シラバスより）

1. ガイダンス：授業の進め方を説明
2. プログラミングとは？(1)：自己紹介～コンピュータの仕組みとプログラム、その社会における役割
3. プログラミングとは？(2)：実習環境設定～初めてのプログラミング
4. 基本文法を学ぼう(1)
5. 基本文法を学ぼう(2)+フォローアップ
6. 標準ライブラリを使ってみよう(1)
7. 標準ライブラリを使ってみよう(2)+フォローアップ
8. プログラムの構造と設計(1)
9. プログラムの構造と設計(2)+フォローアップ
10. データを読み解こう(1)
11. データを読み解こう(2)+フォローアップ
12. 品質の高い美しいプログラムを作るには(「アート」な「呪文」への道)+ディスカッション
13. 画像を認識するAIプログラムを作ってみよう(1)
14. 画像を認識するAIプログラムを作ってみよう(2)
15. 授業のまとめと振り返り：この後どうする？

授業

01. 4/12 **ガイダンス**

02. 4/19 プログラミングとは？(1) 自己紹介 + Anaconda設定 + lightbot

03. 4/26 プログラミングとは？(2) コンピュータでプログラムが動く仕組み

04. 5/10 Pythonを始めよう paiza.IO + pythonインタプリタ(python.exe) + 対話モードPEPL

05. 5/17 Pythonに慣れよう Jupyter Notebook + print文・変数・四則演算

06. 5/24 Pythonの基本(1) for文、if文、while文、変数と型

07. 5/31 Pythonの基本(2) 0と1での表現(音・画像・文字)、文字化け、関数、リスト、PEP8

08. 6/07 Pythonの基本(3) プログラミングとライブラリ、辞書型・集合型、標準ライブラリ

09. 6/14 Pythonの基本(4) Pythonここまでのまとめ (FizzBuzzなど)

10. 6/21 Pythonの実践(1) Jupyter Notebook自作、外部ライブラリpandasとmatplotlib

11. 6/28 Pythonの実践(2) データの集計・分析と可視化 (pandas + matplotlib)

12. 7/05 Pythonの実践(3) イチからプログラミングしてみる (High or Low、Hit & Blowなど)

13. 7/12 Pythonの応用(1) AIと画像認識、画像と配列、numpyと配列

14. 7/19 Pythonの応用(2) scikit-learn、手書き数字画像認識

15. 7/26 **まとめ**

16. 8/02 レポート(Jupyter Notebook形式)作成日

ふりかえり

「プログラムの構造と設計」

Zen of Python、PEP8、イチからプログラミング、
ライブラリとモジュール

「品質の高い美しいプログラムを作るには」

当初一回をこれに使い発表してもらおう予定
だったが「これまでのまとめ演習 x 2」に変更

フォローアップ・ディスカッション

個人差が大きかったのでミニレポートQ & Aを
通して質問や感想を受け付け→それを反映で実施

The Zen of Python (by Tim Peters)の抜粋いくつか

>> import this

Beautiful is better than ugly.

醜いより美しいほうがいい。

Explicit is better than implicit.

暗示するより明示するほうがいい。

Simple is better than complex.

複雑であるよりは平易であるほうがいい。

Readability counts.

読みやすいことは善である。

If the implementation is hard to explain, it's a bad idea.

コードの内容を説明するのが難しいのなら、それは悪い実装である。

美学がある！



大事なこと

注意：これは「プログラミング」を教えてもらう受動的座学の授業ではなく、「プログラミング」をどうやって学んでいけば良いかの根本骨格、つまり自分で学んでいくための素養を得るための授業

なぜなら、

「プログラミング」とは「プログラミング言語」を学ぶことではないから
「プログラミング」とは結局自分で実際にやってみないと分からないから

例え：スイカの味を食べたことのない人にいくら説明しても結局食べてみないことには本当のことはわからない！

授業形態と成績について

- 出席2/3(10回)以上を成績評価対象
(演習なので基本的には全ての授業への参加が望ましい)
- **成績**：授業や演習への積極性(30%), レポートや成果物の内容(50%)、ディスカッションへの貢献(20%)を総合的に評価
- **評価**：入門授業であるため、プログラムのクオリティそのものよりも、考察やコメントによる思考過程のプレゼンテーションをより重視する。プログラミングとは何かを理解し、プログラムが作れるようになっていく「学習過程の磨き方そのもの」を最も評価する。

プログラミングって何？

プログラミングとは何でないか

- 「コーディング」(コンピュータにむかって何やらか
ちゃかちゃ打ち込むこと!)とは違う
- 「プログラミング言語」の学習とは違う
- コンピュータを使ってやることというわけでもない
(コンピュータなくともプログラミングはできる)

答えは千差万別(みなさんの中に!)

瀧川の個人的感覚

「プログラミング」

= 論理的思考(設計)そのもの

+ 表現や構造の美学

コンピュータが
実行するから
(工学技術)

人に読まれるため
に書くものだから
(創作活動)

到達目標（シラバスより）

1. 「プログラミングとは何か？」について自分なりの答えが持てるようになる。
2. プログラミングに最低限必要な「コンピュータの仕組み」や「アルゴリズム」「データ構造」と言った初歩的知識を身につける。
3. 実際にプログラミング言語を用いて簡単なプログラムを作成するための基本的なスキルを身につけ実践することができる。
4. 今後、プログラミングを自ら習得していくために何をしていけば良いかの理解や知識を得る。
5. **コンピュータプログラムが我々の生活社会の基盤にあることを意識し、品質の高い美しいプログラムを作り上げることの価値を議論できるようになる。**

2020問題？

2020年度（平成32年度）から小学校でプログラミング教育が必修化



[会見・報道・お知らせ](#)

[政策・審議会](#)

[白書・統計・出版物](#)

[申請・手続き](#)

[文部科学省の紹介](#)

[教育](#)

[科学技術・学術](#)

[スポーツ](#)

[文化](#)

[トップ](#) > [教育](#) > [小学校、中学校、高等学校](#) > [教育の情報化の推進](#) > [教育の情報化に関する取組](#) > [情報教育の推進](#) > [プログラミング教育](#) > [小学校プログラミング教育の手引（第一版）](#)

● 教育の情報化の推進

小学校プログラミング教育の手引（第一版）

小学校プログラミング教育の手引（第一版）について

今般文部科学省では、新小学校学習指導要領におけるプログラミング教育の円滑な実施に向けて、「小学校プログラミング教育の手引（第一版）」を取りまとめました。

本手引は、学習指導要領や同解説で示している小学校段階のプログラミング教育についての基本的な考え方などをわかりやすく解説し、教師がプログラミング教育に対して抱いている不安を解消し、安心して取り組めるようにすることをねらいとしており、小学校プログラミング教育導入の経緯、小学校プログラミング教育で育む力、プログラミング教育のねらいを実現するためのカリキュラム・マネジメントの重要性と取組例などについて解説するとともに、教育課程内における指導例や、企業・団体や地域等との連携の例などを掲載しています。

また、本手引は、今後の各学校における実践の充実を踏まえ、また、「未来の学びコンソーシアム」の取組とも連携を図りながら、適時改訂を重ね、充実させていく予定です。

この手引を参照いただき、小学校段階のプログラミング教育の実施に向けての準備や実践等にお役立てください。

小学校プログラミング教育の手引（第一版）

[小学校プログラミング教育の手引（第一版）](#) (PDF:3578KB) 

[▶ 教育の情報化の推進](#)

[▶ 教育の情報化に関する方針等](#)

[▶ 教育の情報化に関する取組](#)

[▶ 教育の情報化に関する基盤整備](#)

[▶ 社会教育の推進](#)

[▶ メディア教育](#)

[▶ リンク](#)

[▶ サイトマップ](#)

[▶ このウェブサイトについて](#)

2020年度（平成32年度）から小学校でプログラミング教育が必修化

- 皆さんのお子さんはプログラミングを小学校で学ぶ（たぶん）
- 今生まれている子供の保護者たちも戦々恐々（娘2歳11ヶ月）
- 「プログラミング」という教科が作られるわけではない
- プログラミングスクールや早期教育がもてはやされている？
- 一因に「人工知能(AI)技術」の社会浸透？



小学生・中学生・高校生向け プログラミング教室 | STAR Programming SCHOOL (スタープログラミングスクール)

お問い合わせはお電話にどうぞ！
03-6380-4123
info@star-programming-school.com

2020年 プログラミング教育 必修化!!

授業見学 & 相談会 受付中!

一人ひとりの子ども達が、それぞれの未来を切り拓くスターへ。

メニュー: スクールについて | コースのご案内 | 開校教室のご案内 | イベント情報 | 受講料金 | 受講生作品 | 保護者様の声



TECH:CAMP

特設 コース メンター 卒業生 ブログ 参加日程・料金 エントリー

人生を変える1ヶ月
最高のプログラミングスキルを最短で

募集締め切り(通常プラン) 11月17日(木) 24時

次回開催 11/24(木) ~ 12/23(金)

次回以降はこちら

説明会&受検申込 申込受付中!

TECH:CAMPにエントリー>

いいね! 695 シェア

TECH:CAMP
こんにちは! 🍌 アクセスしていただき、ありがとうございます。お申し込みありがとうございます。お申し込みありがとうございます。

子どもを億万長者にしたければプログラミングの基礎を教えなさい



松林弘治
Koji Matsubayashi

Teach your kids to code to turn them into billionaires

「著作権保護コンテンツ」

あなたが10年後に生き残っているために

プログラミングを知らないビジネスパーソンのためのプログラミング講座

A Programming Course for Business People
福嶋紀仁 *Kei-Hito Fukushima*

プログラミングができなくても、「プログラミングはわかる」のが一流のビジネスパーソンだ。

IT時代に不用品扱いされないための「金曜日」プログラミング。これから始めるビジネスパーソンが最初に読んでほしい1冊。
「読むだけでわかるプログラミング」

日経 Kids+ 子どもの未来を拓く! (17歳以下専用)

いよいよ小中学校で必修化!

親子で始めるプログラミング

PROGRAM (Programming) COMPUTER



親子で楽しく作る! プログラムを作ってみよう! 文章でもよくわかる! そもそもプログラムとは? 入会から卒業まで! プログラミングの楽しみ方

プログラミング必修化で子どもたちは何を学ぶのか

Windows パソコン だけで OK!

「すごい!」「なぜ?」「どうして?」子どもといっしょに

コンピュータとプログラミングを学ぶ本

矢沢久雄 著

キー入力 | 計算処理 | 表計算 | 確率化・優等 | 計算誤差
小数点数 | バグ・デバッグ | ゲーム | 画像処理 | 学習環境

日経BP社

親子で楽しみながら考える力、つくる力、伝える力を育もう!

小学生から始める

わくわくプログラミング

Scratch 1.4/2.0



MITメディアラボ レズニック教授 推薦!

これからの社会に不可欠な3つの能力を身につけよう!

創造力 論理的思考力 共感力

子供から大人まで楽しめる

日経パソコン 編

この1冊で、子どもの思考力、創造力、学力が確実にアップ!

小中学生からはじめるプログラミングの本

2018年版



夢中になるゲームの作りかたくさん!

プログラミング教育の必修化で何が何をすべき?

高校生からはじめる

プログラミング

古村毅一郎 著

N高校のプログラミング教育メソッドを大公開!



KADOKAWA

監修 株式会社ドワンゴ
著者 松林弘治

プログラミングは最強のビジネススキルである

KADOKAWA

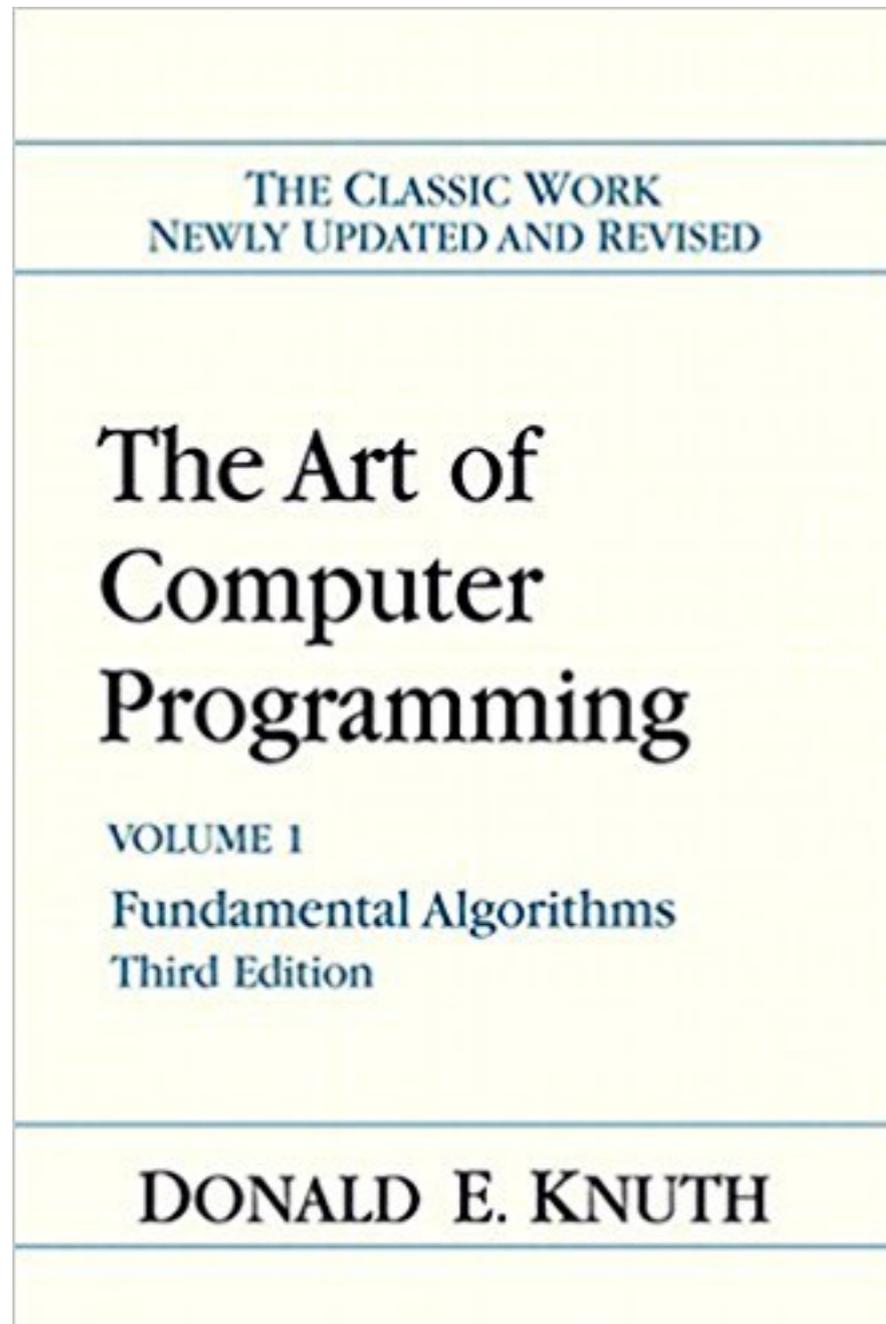
! ?

2020年から始まるプログラミング教育では、
プログラミング言語を記述して、プログラム
を動かすことは行いません (by 文科省)

プログラミングは「アート」 by Don Knuth

バイブル 通称"TAOCP"

1974年ACMチューリング賞 受賞講演



Computer Programming as an Art

by Donald E. Knuth

When *Communications of the ACM* began publication in 1959, the members of ACM's Editorial Board made the following remark as they described the purposes of ACM's periodicals [2]: "If computer programming is to become an important part of computer research and development, a transition of programming from an art to a disciplined science must be effected." Such a goal has been a continually recurring theme during the ensuing years; for example, we read in 1970 of the "first steps toward transforming the art of programming into a science" [26]. Meanwhile we have actually succeeded in making our discipline a science, and in a remarkably simple way: merely by deciding to call it "computer science."

Implicit in these remarks is the notion that there is something undesirable about an area of human activity

Copyright © 1974, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

Author's address, Computer Science Department, Stanford University, Stanford, CA 94305

that is classified as an "art"; it has to be a Science before it has any real stature. On the other hand, I have been working for more than 12 years on a series of books called "The *Art* of Computer Programming." People frequently ask me why I picked such a title; and in fact some people apparently don't believe that I really did so, since I've seen at least one bibliographic reference to some books called "The *Act* of Computer Programming."

In this talk I shall try to explain why I think "Art" is the appropriate word. I will discuss what it means for something to be an art, in contrast to being a science; I will try to examine whether arts are good things or bad things; and I will try to show that a proper viewpoint of the subject will help us all to improve the quality of what we are now doing.

One of the first times I was ever asked about the title of my books was in 1966, during the last previous ACM national meeting held in Southern California. This was before any of the books were published, and I recall having lunch with a friend at the convention hotel. He knew how conceited I was, already at that

プログラミングは「魔法をかける呪文」

バイブル 通称"SICP"

Structure and
Interpretation
of Computer
Programs

Second Edition



Harold Abelson and
Gerald Jay Sussman
with Julie Sussman

We are about to study the idea of a *computational process*. Computational processes are abstract beings that inhabit computers. As they evolve, processes manipulate other abstract things called *data*. The evolution of a process is directed by a pattern of rules called a *program*. People create programs to direct processes. **In effect, we conjure the spirits of the computer with our spells.**

我々の呪文でコンピュータの精霊を呼ぶ

"Why programming is a good medium for expressing poorly understood and sloppily-formulated ideas" (Marvin Minsky)

"人工知能の父"



WHY PROGRAMMING IS A GOOD MEDIUM FOR EXPRESSING POORLY UNDERSTOOD AND SLOPPILY-FORMULATED IDEAS

Marvin Minsky

MIT

This is a slightly revised version of a chapter published in *Design and Planning II -- Computers in Design and Communication*, (Martin Krampen and Peter Seitz, eds.), Visual Committee Books, Hastings House Publishers, New York, 1967.

There is a popular, widespread belief that computers can do only what they are programmed to do. This false belief is based on a confusion between form and content. A rigid grammar need not make for precision in describing processes. The programmer must be very precise in following the computer grammar, but the content he wants to be expressed remains free. The grammar is rigid because of the programmer who uses it, not because of the computer. The programmer does not even have to be exact in his own ideas—he may have a range of acceptable computer answers in mind and may be content if the computer's answers do not step out of this range. The programmer does not have to fixate the computer with particular processes. In a range of uncertainty he may ask the computer to generate new procedures, or he may recommend rules of selection and give the computer advice about which choices to make. Thus, computers do not have to be programmed with extremely clear and precise formulations of what is to be executed, or how to do it.





新しいビデオゲームを買うだけでなく、自ら作りましょう。
最新のアプリをダウンロードするだけでなく、設計してみましよう。
それらをただ遊ぶだけでなく、プログラムしてみましよう。(中略)
あなたが、誰であっても、どこに住んでいてもコンピューターはあ
なたの将来において重要な役割を占めます。あなたがもし勉強を頑
張れば、あなたの手で未来を創り出すことができるでしょう。



———アメリカ大統領 バラク・オバマ

アメリカ人は全員コンピュータのプログラミングを学ぶべきだと思うね。なぜなら、コンピュータ言語を学ぶことによって「考え方」を学ぶことができるからだ。ロースクールに行くようなものだよ。全員が弁護士になるべきだとはいわないけれど、現実にロースクールに通うことは人生に役立つはずだ。一定の方法で物事の考え方を学べるからね。

—— Apple 社創業者 スティーブ・ジョブズ



プログラミングの勉強をはじめたのは、コンピュータサイエンスのすべてを知りたいとか、原則をマスターしようとか、そういうことではまったくありませんでした。ただ、やりたいことがひとつあって、自分と自分の妹たちが楽しめるものを作りたいと想っていたんです。(中略)

大学の寮の部屋で何かをはじめることができる。大きい会社なんて作ったことがない友達と集まって、何億という人が日常生活の一部として使うものを作る。想像するだけですがすごいことです。ちょっと怖いけれど、素晴らしいんです。



—— Facebook 社創業者 マーク・ザッカーバーグ

The Zen of Python (by Tim Peters)の抜粋いくつか

>> import this

Beautiful is better than ugly.

醜いより美しいほうがいい。

Explicit is better than implicit.

暗示するより明示するほうがいい。

Simple is better than complex.

複雑であるよりは平易であるほうがいい。

Readability counts.

読みやすいことは善である。

If the implementation is hard to explain, it's a bad idea.

コードの内容を説明するのが難しいのなら、それは悪い実装である。

美学がある！



現代社会とプログラム

- パソコン、スマホ、タブレットPC
- インターネット (Webサービスやネットショップ)
- ゲーム
- 映画、音楽、CG、アニメ、ライブ
- 銀行や市場の基幹システム
- 自動車、航空、宇宙、原発、防衛など
- 電気・ガス・水道などのインフラ
- ビジネスや産業
- 物流・運輸
- 科学

Society 5.0



http://www8.cao.go.jp/cstp/society5_0/index.html

Society 5.0

新たな社会
“Society 5.0”

5.0



1.0

Society 1.0 狩猟

2.0

Society 2.0 農耕

4.0



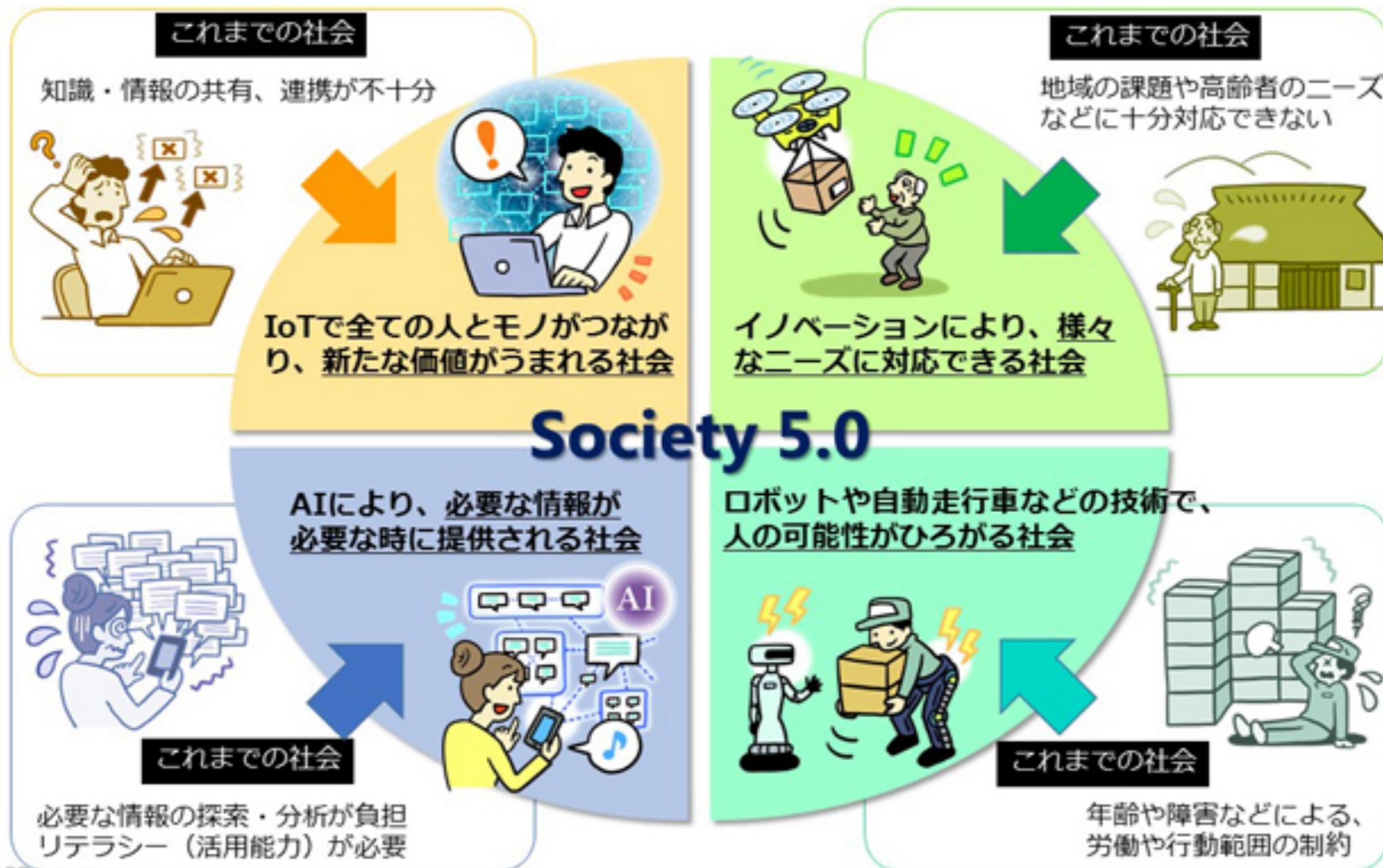
Society 4.0 情報

Society 3.0 工業

3.0

[内閣府作成]

Society 5.0



到達目標（シラバスより）

1. 「プログラミングとは何か？」について自分なりの答えが持てるようになる。
2. プログラミングに最低限必要な「コンピュータの仕組み」や「アルゴリズム」「データ構造」と言った初歩的知識を身につける。
3. 実際にプログラミング言語を用いて簡単なプログラムを作成するための基本的なスキルを身につけ実践することができる。
4. 今後、プログラミングを自ら習得していくために何をしていけば良いかの理解や知識を得る。
5. **コンピュータプログラムが我々の生活社会の基盤にあることを意識し、品質の高い美しいプログラムを作り上げることの価値を議論できるようになる。**

プログラミングを考える

- 社会で必要とされる「プログラム」は一人で開発できる規模ではなく大人数で開発するしかないが、大人数で一つのプログラムを効率的に作るにはどうすれば良いだろう？
- 原発や防衛ミサイルや航空機の制御など、ミスがゆるされない制御の「プログラム」をどう設計しどうテストすれば良いだろう？

→ 「ソフトウェア工学」という専門分野のQuestion

ざっくり言えば「プログラム」とは「人」が作り

「人」がメンテ・改修のため読むもの、という点を
真剣に考えなければいけない、ということ

初心者から中級者・上級者への道

- レベル1:

プログラム言語の読み書きができる

- レベル2:

自分で作りたいプログラムを計画し、設計し、実際にコードを書いて実行できる



- レベル3 (技術者・専門家)

コードを読む
専門知識をつける

- レベル3 (趣味)

コードを読む
コードを公開する

ソフトウェア開発とは

要件定義・要求分析

何をするか？

設計

どのようにするか？

実装

これができるにはプログラミング言語で何が
できるかや対象についての専門知識が必要

テスト

運用保守

このあとどうする？

プログラミングを学ぶ上で最も大事なこと
実際に手を動かし、自ら考えて開発を体験すること

スポーツと同じでルールを知っているのと実際に行うことの
間に越えることのできない大きな壁があります！

このあとどうする？

このような状況になるなら「プログラミング」の本質を
理解できていません

- 書籍に書いてあるサンプルは作ることができたけど、いざ自分が作りたいものを作ろうとすると何から始めてよいかわからなくなる
- 書いてあることの意味は理解できるけれど、ゼロから書くことはできない

一旦基本が理解できたら「ここを変えたらどうなるのかな」
「このプログラムを元にこう拡張してみよう」「このプログラムに機能を一つ追加してみよう」と少しづつでも自ら考え
実践することが必要です！

このあとどうする？

1. まずはプログラミング言語(例えばPython)について基本的なこと(関連ライブラリの知識含む)を学ぶ
2. 専門書を読んだり、ネット検索で必要な情報を探しだし(このスキル重要)、知識を深める
3. 場合によっては関連する必要な知識も合わせて理解しておく
スマホアプリならインターネットやWebの基本知識、
ゲームを作るならグラフィクスやCGの基本知識、
ロボットを動かすなら物理や回路や制御の基本知識、など
4. 用途によっては適切なプログラミング言語があるのでそれについて上記を行う

授業でやらなかったこと

むしろやったことのほうが限られています…

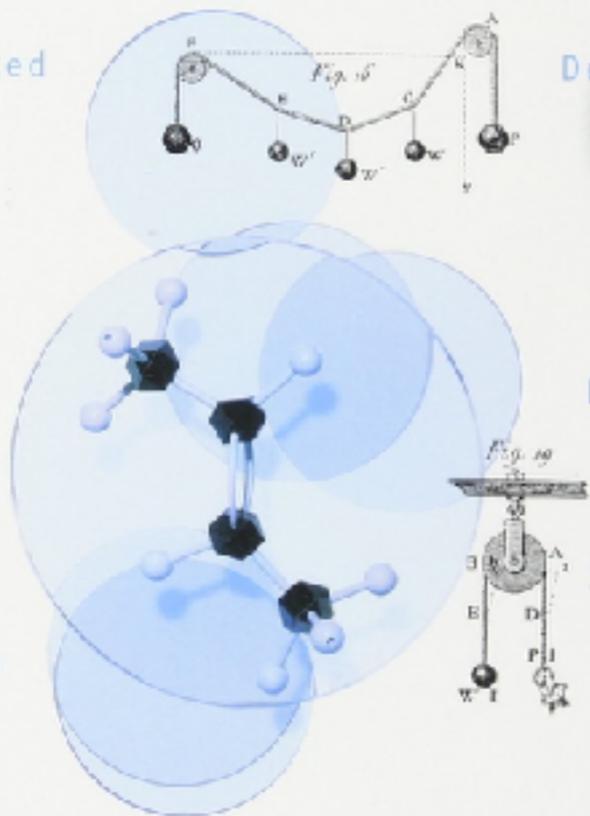
1. Pythonの少し発展的な部分：特に「クラス」
→ これはPythonに限らずJavaやrubyでも理解が必要
「オブジェクト指向プログラミング」という設計思想
2. Pythonの関係ライブラリやライブラリの詳細
→ これは用途によるので(AI、Web、CG、GUI、通信、等)
3. 他人が書いたコードを読む練習
→ 書けるためには読める必要があります
4. 「良いコード」とはの追求 (PEP8がなぜ大事か)
→ ある程度かけたら書籍「リーダブルコード」がオススメ

How Objects Work
2nd Edition

オブジェクト指向で なぜつくるのか

● 知っておきたい OOP、設計、関数型言語の基礎知識 ●
平澤 章 著

Object Oriented
Simula67
Smalltalk
C++
Java
C#
Class
Instance
Method
Package
Interface
Superclass
Subclass
Polymorphism
Inheritance
Aggregation
Relation
Dependency
Message Passing
Garbage Collection
Aspect Oriented
Agent Oriented



日経BP社

第 2 版

Design
Class Library
Framework
Component
Set Theory
Responsibility
UML
Use Case
Activity
State Machine
Modeling
Analysis
Design
Iteration
RUP
Architecture
XP
Refactoring
Pair Programming
Agile Manifesto

/THEORY/IN/PRACTICE

リーダブルコード

より良いコードを書くための
シンプルで実践的なテクニック

O'REILLY®
オライリージャパン

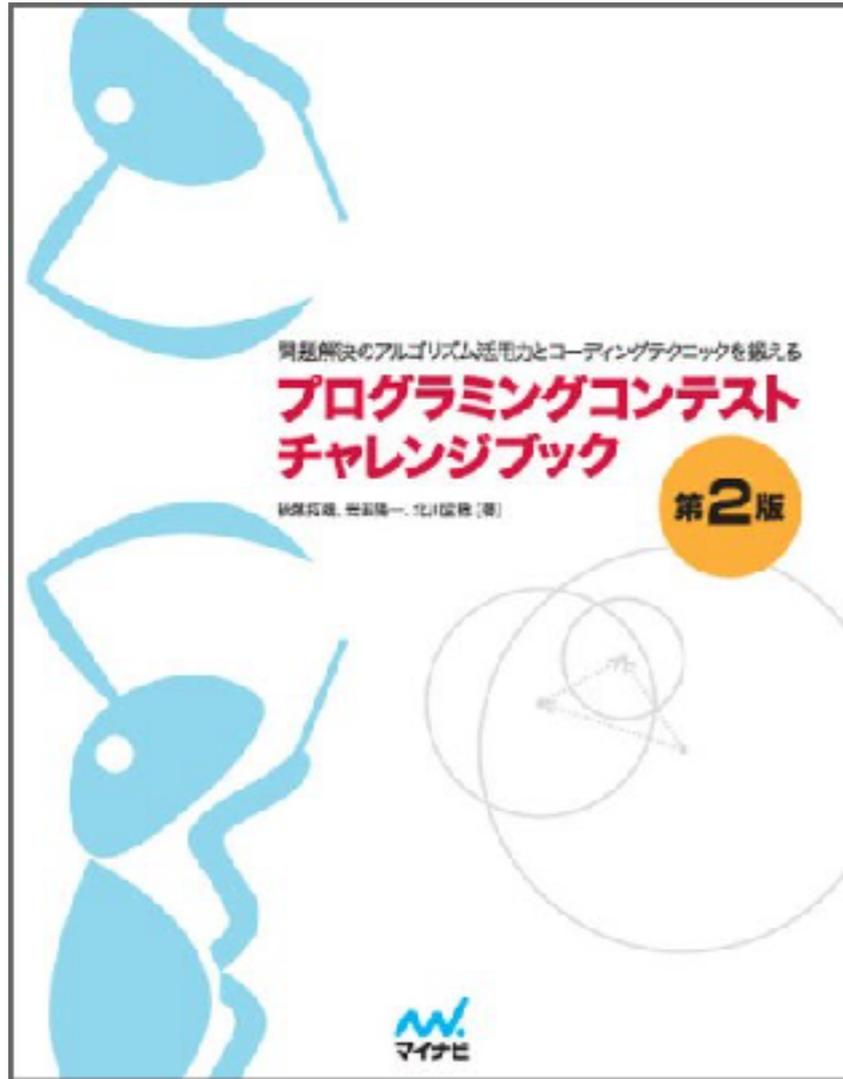
Dustin Boswell 著
Trevor Foucher 訳
角 征 典 訳

プログラミングコンテスト(プロコン) 競技プログラミング(競プロ)

<https://ja.wikipedia.org/wiki/競技プログラミング>



<https://www.slideshare.net/iwiwi/wakate-web-14323842>



初心者にもオススメ！競技プログラミングの問題を解けるサイト8選

<http://programming-study.com/trouble/problem/>

世界で闘う プログラミング力を鍛える本

—コーディング面接189問とその解法—

Gayle Laakmann McDowell [著]
岡田佑一、小林啓倫 [訳]

CRACKING THE CODING INTERVIEW

189 Programming Questions and Solutions
Gayle Laakmann McDowell



全プログラマー“必読”

Google・Apple・Amazon.. 有名IT企業のコーディング面接での出題から選び抜いた最高の189問を収録。プログラマーとしての技術力をもう一段階上のレベルに引き上げよう。



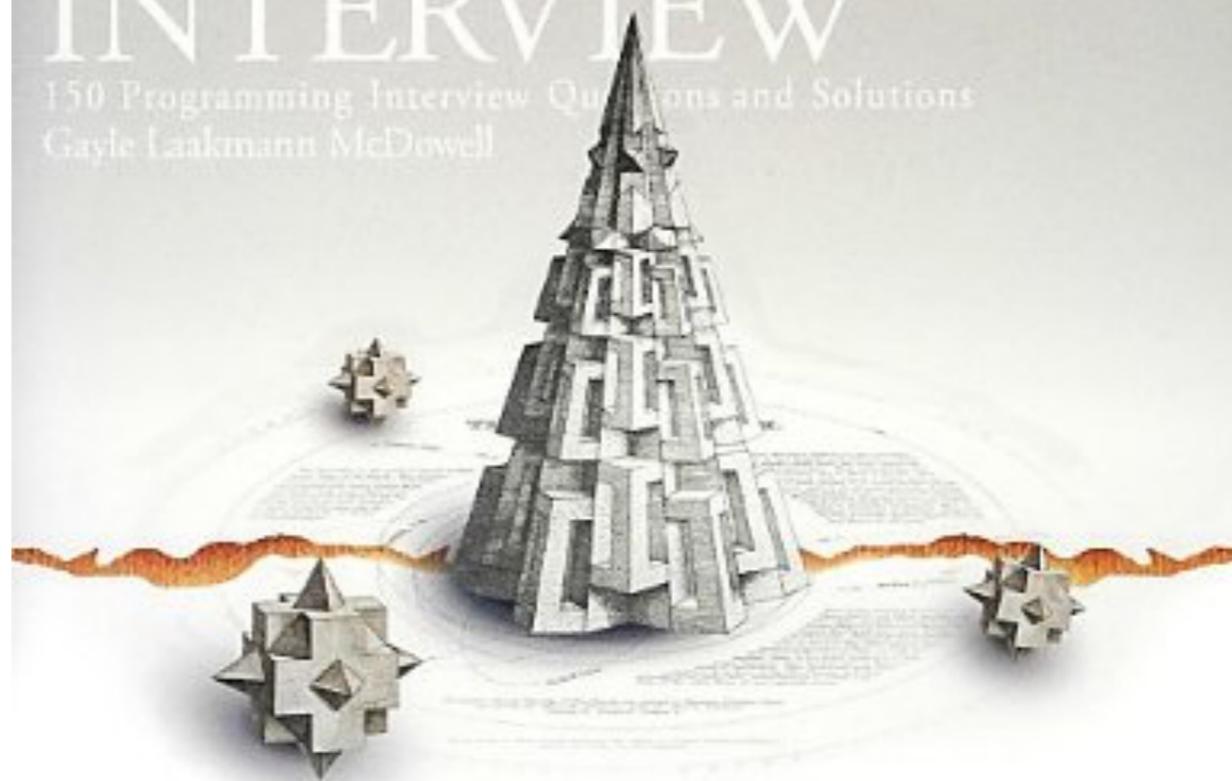
世界で闘う プログラミング力を鍛える150問

—トップIT企業のプログラマーになるための本—

Gayle Laakmann McDowell [著] Ozy [訳]
秋葉拓哉、岩田陽一、北川宜徳 [監訳]

CRACKING THE CODING INTERVIEW

150 Programming Interview Questions and Solutions
Gayle Laakmann McDowell



オープンソースソフトウェア(OSS)

<https://ja.wikipedia.org/wiki/オープンソースソフトウェア>

「著作権保護コンテンツ」
図解入門 ビジネス Shurwasystem Business Guide Book How-nual

最新
オープンソースが
よ〜くわかる**本**

人工知能も、FinTech(ブロックチェーン)も、
ビッグデータも、みんなオープンソースでできている!

Facebookやマイクロソフト、
Googleは、なぜ人工知能を
オープンソースで公開するのか?



- 「昔」と違う「今」のオープンソース!
- システムインテグレーションは減るのか?
- 生き残るためにはオープンソース戦略!
- これからのシステムは「つくらない」!
- Slerの脱「下請」は、OSSビジネスで!
- 今始めるべき「旬のOSS」を一挙公開!

寺田 雄一 著 西和システム
「著作権保護コンテンツ」

WEB+DBPRESS plus

GitHub 実践入門

Pull Requestによる開発の変革

Ohno Hiroki
大塚弘記
(著)

良いコードを迅速に生み出す
快適な共同開発

バージョン管理、ブランチ、マージ、Fork
Issue、バグ・タスク管理、Markdown
コードレビュー、ソーシャルコーディング
Jenkins、GitHub Flow、Git Flow、hubコマンド

手を動かして身に付ける、実用的なワークフロー

技術評論社

Pythonでできること

- 身近な作業の自動化 (家計簿、目覚まし、英単語学習)
- 簡単なゲームAI
- スマホアプリ (Webアプリケーション)

Pythonでできたもの

- Instagram
- DropBox
- Youtube

Instagramの開発

- 立ち上げ時は2人で開発。プログラミングは独学
- ユーザーが増えるに伴い試行錯誤でスケールさせてきた
- Webフレームワークにはpythonの「Django (ジャンゴ)」
- Facebook社が10億ドル(約1100億円)で買収

DropBoxの開発

- 125人のエンジニアを編成
- すべてPythonでプログラミング
- サーバとクライアント関連で合わせて93万7,707行にも及ぶ大規模なもの

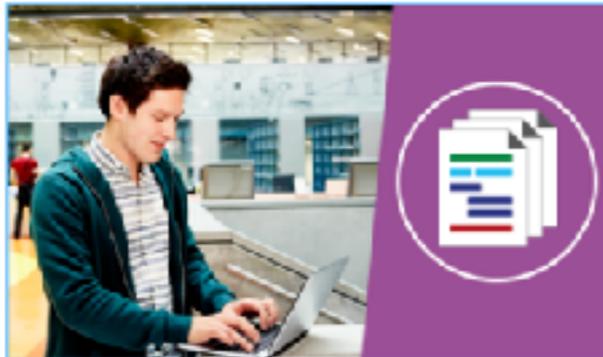
オンラインで無料の良い講義も多数！

<https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>



Introduction to Python: Absolute Beginner

In this course that's perfect for true beginners, learn Python basics and start coding right away.



Introduction to Python: Fundamentals

Build on what you learned in the "Introduction to Python: Absolute Beginner" course, and dig into data structure basics.



FREE COURSE

Introduction to Python

Starting Out in Python 3

START FREE COURSE

Python以外の言語について

Webに関心あるなら

まずは、インターネットとは？Webとは？を理解しHTMLとCSS
プログラミング言語としてはJavaScriptは必須
(それ以外にpython or ruby or 古いシステムならPHPやperl)

iPhoneアプリに関心あるなら

プログラミング言語としてはSwiftは必須かも

Androidアプリに関心あるなら

プログラミング言語としてはJavaは必須かも

VRや3Dゲームに関心あるなら

まずは、コンピュータグラフィックスの基礎を理解
プログラミング言語としてはC#などがいいかも

技術者をめざすなら

王道は(Javaに加えて)情報系ならCとC++、工学系ならFORTRAN

心理的な障壁を感じる...という場合



The screenshot shows the Japanese homepage for Hour of Code. At the top left is the 'HOUR OF CODE' logo. A navigation bar contains links for 'アクティビティ' (Activities), '開催ガイド' (Event Guide), '宣伝する' (Promote), 'よくある質問' (FAQ), and a language dropdown menu set to '日本語'. The main content area features a large image of a young boy looking at a tablet. Text overlaying the image reads: 'すべての生徒に、Hour of Codeを' (For all students, Hour of Code), '571,667,665人が受講' (571,667,665 people have taken the course), and a prominent orange button that says '学習を開始しましょう' (Let's start learning). Below the button is a link 'ビデオを見る' (Watch video) with a play icon. At the bottom of the page, a teal box contains the text: 'Hour of Code は、180カ国以上から数千万人の生徒が参加する世界的なムーブメントです。誰でも、どこでもHour of Codeのイベントを開催できます。1時間のチュートリアルは45カ国の言語に翻訳されています。4歳から104歳まで、経験は必要ありません。' (Hour of Code is a global movement with millions of students from over 180 countries. Anyone, anywhere can host an Hour of Code event. The 1-hour tutorial is translated into 45 languages. Ages 4 to 104, no experience necessary.) A 'Q&A' button is visible in the bottom right corner.

Hour of Codeとは何ですか？

Hour of Code は、「コード（プログラミング）」の謎を解き明かし、全ての人が基本を学ぶことができることを示し、コンピューターサイエンス分野の幅広い参加者に向けて作成された、1時間のコンピューターサイエンス入門として始まりました。

ビジュアルでもよいかも

「ビジュアル」プログラミング

lightbot



Scratch



自習のススメ (論理的思考とは?)

1. lightbot

demo版をとりあえずクリアしてみる

<http://lightbot.com/hoc.html>

2. Hour of Code (コードの時間)

1時間でできるらしいので下記「初めてのコンピュータプログラムを書く」をぜひ。講師ビデオにはビル・ゲイツやマーク・ザッカーバーグも登場。lightbot同様、ゲーム感覚のもの。

<https://hourofcode.com/code>

本: 本当の本当に 0 スタート向け

いちばんやさしい
Python パイソン
入門教室 大澤文孝 [著]

豊富な
カラー図解と
イラストで
超わかる!



必須の**基礎知識**と**基本文法**が
この1冊でしっかり身につきます。

【  コードの読み書き
がはじめての
未経験者 】 【  スキルアップ
で上を目指す
初級者 】 **プログラミングを学ぶ
すべてのビギナーに
最良の入門書!**

プログラムの**読み方****書き方****しくみ****動かし方**を
根本から理解し、作りながらしっかり学べます。

 Webからサンプルを
ダウンロードできます

いちばんやさしいパイソンの本

Python

スタートブック
[増補改訂版]
辻真吾 Shingo Tsuji

バージョン
3
に完全対応!

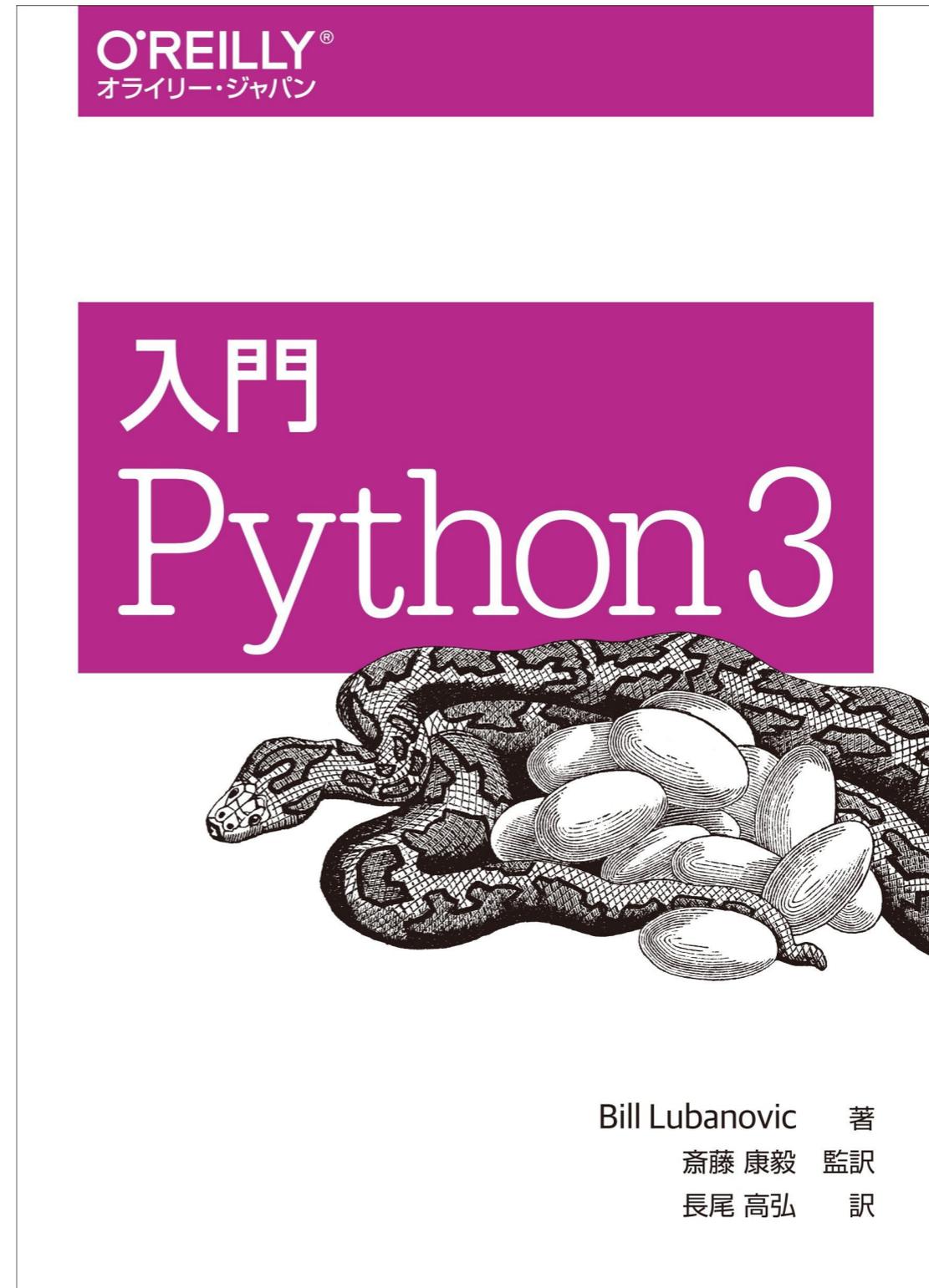


技術評論社

● Python 3.x による最新の開発環境に完全対応
● 知識ゼロでもわかるイラスト&サンプル満載の解説
● オブジェクト指向の考え方をしっかり理解できる
● Web アプリ開発やデータ処理の基礎もわかる!

まったくの ゼロからでも大丈夫!

本: もう少し広いこともある入門



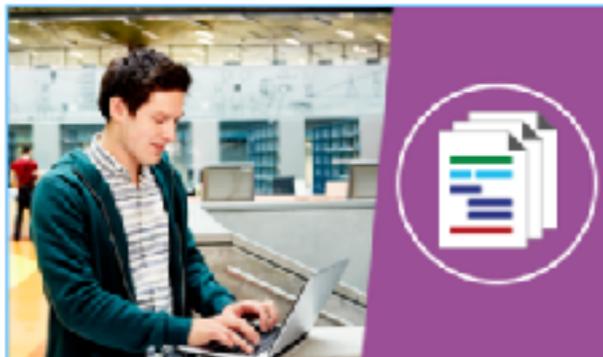
オンラインで無料の良い講義も多数！

<https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>



Introduction to Python: Absolute Beginner

In this course that's perfect for true beginners, learn Python basics and start coding right away.



Introduction to Python: Fundamentals

Build on what you learned in the "Introduction to Python: Absolute Beginner" course, and dig into data structure basics.



FREE COURSE

Introduction to Python

Starting Out in Python 3

START FREE COURSE

- LearnPython.org interactive Python tutorial (英語)

Indentation

Python uses indentation for blocks, instead of curly braces. Both tabs and spaces are supported, but the standard indentation requires standard Python code to use four spaces. For example:

```
script.py  IPython Shell
1 x = 1
2 if x == 1:
3     # indented four spaces
4     print("x is 1.")
```

Run

Powered by DataCamp 

Exercise

Use the "print" command to print the line "Hello, World!".

```
script.py  IPython Shell
1 print("Goodbye, World!")
```

CodeAcademy (英語)

The screenshot shows the Codecademy interface for a Python lesson. On the left, there is a sidebar with the Codecademy logo and a 'Learn' button. The main content area is titled 'PYTHON SYNTAX' and 'Hello World!'. It contains a paragraph explaining that programming is teaching a computer to have a conversation with a user, and that this is done using the 'print' statement. Below this is a code block with the following text:

```
print "Hello, world!"  
print "Water--there is not a drop  
of water there! Were Niagara but a  
cataract of sand, would you travel  
your thousand miles to see it?"
```

 Below the code block, it says 'A print statement is the easiest way to get your Python program to communicate'. At the bottom of the sidebar, there are three options: 'Instructions' (checked), 'Community Forums', and 'Report a Bug'. The right side of the interface is a code editor with a file named 'script.py' containing the code

```
1 print "hoge"
```

. Below the code editor is a 'Run' button and a refresh icon. The output area on the right shows the text 'hoge'. At the bottom of the interface, there is a navigation bar with a hamburger menu, '1. Hello World!', a 'Back' button, '1/14', a 'Next' button, and a 'Get Help' button.

codecademy < Learn Python

Learn

PYTHON SYNTAX

Hello World!

If programming is the act of teaching a computer to have a conversation with a user, it would be most useful to first teach the computer how to speak. In Python, this is accomplished with the `print` statement.

```
print "Hello, world!"  
print "Water--there is not a drop  
of water there! Were Niagara but a  
cataract of sand, would you travel  
your thousand miles to see it?"
```

A `print` statement is the easiest way to get your Python program to communicate

Instructions

Community Forums

Report a Bug

script.py

```
1 print "hoge"
```

hoge

Run

1. Hello World! Back 1/14 Next Get Help

- Intro to Python for Data Science (英語)

The screenshot shows a video player interface for a DataCamp course. At the top left is the DataCamp logo. In the top center, there is a 'Course Outline' button with a right-pointing arrow. At the top right, there are icons for notifications, a document, and a warning sign. Below the navigation bar, the title 'Hello Python!' is displayed in large, bold, dark blue font. To the right of the title, a badge indicates '50XP'. The video content area features a blue header with a Python logo icon. Below this, a large white shield-shaped graphic contains the text 'INTRO TO PYTHON FOR DATA SCIENCE' and 'Hello Python!' in bold black font. A man, Filip Schouwenaars, is shown in the bottom right corner of the video frame, wearing a black polo shirt with the DataCamp logo. At the bottom left of the video frame, the name 'Filip Schouwenaars' and a timestamp '0:04' are visible. The video player controls at the bottom include a play button, a volume icon, a progress bar, a time display of '-3:32', a download icon, a '1x' speed control, a CC icon, and a full-screen icon. A yellow 'Got it!' button is located at the bottom right of the player interface.

● Progate (プロゲート)



文字列

文字列とは？

文字列

クォーテーション

先ほどの例で用いた「Hello Python」という文字は、プログラミングの世界では「文字列」と呼ばれます。

文字列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「"」で囲む必要があります。どちらで囲んでも出力結果は同じとなります。どちらかで囲んでいない場合、コードは動かなくなります。

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

```
script.py x  
print('Hello Python')  
print("Hello Python")
```

```
script.py x  
# エラー発生  
print(Hello Python)
```

出力結果

>_ コンソール

```
Hello Python  
Hello Python
```

出力結果は同じ

>_ コンソール

```
SyntaxError: invalid syntax  
~~~~~  
X エラー!!!: 必ず文字列はクォーテーションで囲む!
```



● ドットインストール

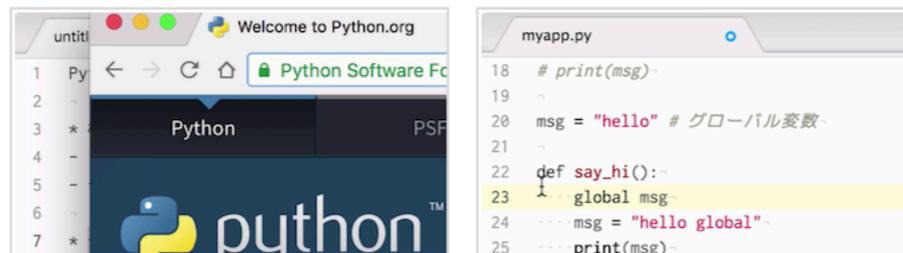
 レッスン一覧 **プレミアム会員** 法人でのご利用 

[トップ](#) / [マイページ](#) / [レッスン一覧](#) / Python 3入門

Python 3入門 (全31回) **PREMIUM**

データ解析や機械学習などにも利用される、シンプルなオブジェクト指向型言語であるPythonについて見ていきます。

i このレッスンでは **Python 3.5.2** を使用しています。



[動画レッスン一覧 \(31\)](#)

[よくある質問一覧 \(9\)](#)

#01 Pythonを使ってみよう (01:58) 無料公開中

 未完了

- 概要
- 公式サイト
- 必要となる知識
- レッスンにおける環境

#02 はじめてのPythonプログラム (02:45) 無料公開中

 未完了

- myapp.pyの作成
- 実行方法
- コメントの書き方

3分動画でマスターする プログラミング学習サイト

初心者向け

すべてのレッスンを見る



ご利用はこちらから

新規登録 (無料)

ログイン



Sign in with Google

Python 3入門、HTML入門、iPhoneアプリ開発入門 など、**349** レッソンを **5,186** 本の **3分動画** にて提供中

プログラミング学習がもっと捗る
プレミアムサービス

月額980円税込

完了

1.01 例えばtanakaというキーが
すね。
1.08 これが例えばnakamuraだっ

```
def update
  @project = Project.find(params[:id])
  if @project.update(project_params)
    redirect_to projects_path
  else
    render 'edit'
  end
end
```

企業内の研修等にご利用いただける
法人向けライセンス

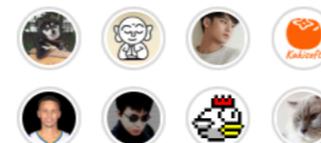
人気のレッスン

» すべてのレッスンを見る (349)

HTML
入門

HTML入門 (全15回)

簡単なプロフィールサイトを作りながらHTMLについて学んでいきます。



新着レッスン NEW !!

CSS
入門

CSS入門 (全17回)

ウェブページの見た目を整えるための言語であるCSSについて学んでいきます。

HTML
入門

HTML入門 (全15回)

簡単なプロフィールサイトを作りながらHTMLについて学んでいきます。

HTML/CSS
学習環境

HTML/CSSの学習環境を整えよう [macOS編] (全4回)

初めてHTML/CSS入門を受講する方に向けて、必要なツールや設定について解説していきます。

HTML/CSSの学習環境を整えよう

[トップ](#) / [マイページ](#) / [レッスン一覧](#)

レッスン一覧 349レッスンを5,186本の動画で提供中

[目的別で探す](#)[絞り込み検索をする](#)

ホームページを 作れるようになろう

ホームページを作るための基礎知識をマスターできるレッスンパックです。まずはここから始めてみましょう。

[HTML/CSSの学習環境を整えよう \[Windows編\] \(全4回\)](#)[HTML/CSSの学習環境を整えよう \[macOS編\] \(全4回\)](#)[HTML入門 \(全15回\)](#)[CSS入門 \(全17回\)](#)[CSSレイアウト入門 \(全15回\)](#)[実践！ウェブサイトを作ろう \(全17回\)](#)[CSSで吹き出しを作ろう \(全8回\)](#)[レスポンシブウェブデザイン入門 \(全14回\)](#)[JavaScriptでモーダルウィンドウを作ろう \(全8回\) **PREMIUM**](#)

JavaScriptから始める お手軽プログラミング

ブラウザとテキストエディタがあればすぐに始められるJavaScriptをマスターするためのレッスンパックです。

[JavaScript入門 \(全24回\)](#)[JavaScriptでおみくじを作ろう \(全9回\)](#)[JavaScriptで5秒当てゲームを作ろう \(全9回\)](#)[JavaScriptで割り勘電卓を作ろう \(全12回\)](#)[JavaScriptでパスワードジェネレータを作ろう \(全8回\)](#)[JavaScriptで文字数チェッカーを作ろう \(全8回\)](#)[JavaScriptでストップウォッチを作ろう \(全13回\)](#)[JavaScriptでカウントダウンタイマーを作ろう \(全12回\)](#)[JavaScriptでスロットマシンを作ろう \(全11回\) **PREMIUM**](#)

iOSアプリを 作れるようになろう

iPhoneやiPad向けアプリの作り方をマスターできるようになるためのレッスンパックです。

iPhoneアプリ開発入門 (全13回)

iOSレイアウト入門 (全14回) **PREMIUM**

Swift 3入門 (全36回) **PREMIUM**

iOSでおみくじアプリを作ろう (全6回) **PREMIUM**

iOSで姓名診断アプリを作ろう (全12回) **PREMIUM**

iOSでストップウォッチを作ろう (全11回) **PREMIUM**

iOSでブラウザを作ろう (全17回) **PREMIUM**

iOSで一行メモアプリを作ろう (全18回) **PREMIUM**

 9,476 人が学習中です

Androidアプリを 作れるようになろう

Androidのスマートフォンやタブレット向けのアプリを作れるようになるためのレッスンパックです。

Androidアプリ開発入門 (全11回)

Androidでおみくじアプリを作ろう (全8回) **PREMIUM**

 1,903 人が学習中です

仕事で使える技術に 挑戦してみよう

仕事で使えるさまざまな技術やツールを学習していくためのレッスンパックです。

WordPress入門 (全23回)

Java 8入門 (全43回) **PREMIUM**

ゲームプログラミングに 挑戦してみよう

ゲームを作りながらプログラミングを楽しく学習していくためのレッスンパックです。

Unity入門 (全26回)

Scratch 2.0入門 (全19回)

Pythonの学習について

- Python公式チュートリアル

<https://docs.python.jp/3/tutorial/>

- LearnPython.org interactive Python tutorial (英語)

<https://www.learnpython.org>

- CodeAcademy (英語)

<https://www.codecademy.com/learn/learn-python>

- Intro to Python for Data Science (英語)

<https://www.datacamp.com/courses/intro-to-python-for-data-science/>

- Python tutorials for beginners (英語)

<http://thepythonguru.com>

- Progate (プロゲート)

<http://prog-8.com/languages/python>

- ドットインストール

https://dotinstall.com/lessons/basic_python_v3

まとめ：

- 基本は「公式チュートリアル」
<https://docs.python.jp/3/tutorial/>
- 本はものすごくいろいろ出ている個人によって難しすぎたり、簡単(冗長)すぎてつまらなかったりするるので、本屋や図書館で手にとって自分の好みか確認するのが良い
- 入門むけのinteractiveなWeb教材は無料でもできが良い。英語もシンプル。ただし発展になると有料になったりするので注意

今日のお題：最終講義

- フォロアーアップQ & A
- この授業の評価について
- 授業のふりかえり：プログラミングとは？
- このあとどうする？
 - 授業でやらなかったこと
 - Python以外の言語
- 授業のまとめ