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(コンピュータなくて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

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？

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

Teach your kids to code
to turn them into billionaires

「著作権保護コンテンツ」

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

あなたが
10年後に
生き残って
いるために

A Programming Course for
Business People

福嶋紀仁

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。

日経Kids+ 子どもの未来を拓く!

いよいよ
小中学校で
必修化!

すごい! なぜ? どうして?

Windows
パソコン
だけで
OK!

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 | 暗号化・復号 | 計算誤差
小数点数 | バグ・デバッグ | ゲーム | 画像処理 | 学習機能

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
はじめる
わくわく
プログラミング

Scratch 1.4/2.0
対応

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!

アラン・ケイ博士
「究極のコンピューター」教室

創造力 論理的思考力 共鳴力

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

高校生から
はじめる
プログラミング

古村総一郎 著

N高校のプログラミング教育
メソッドを大公開!

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

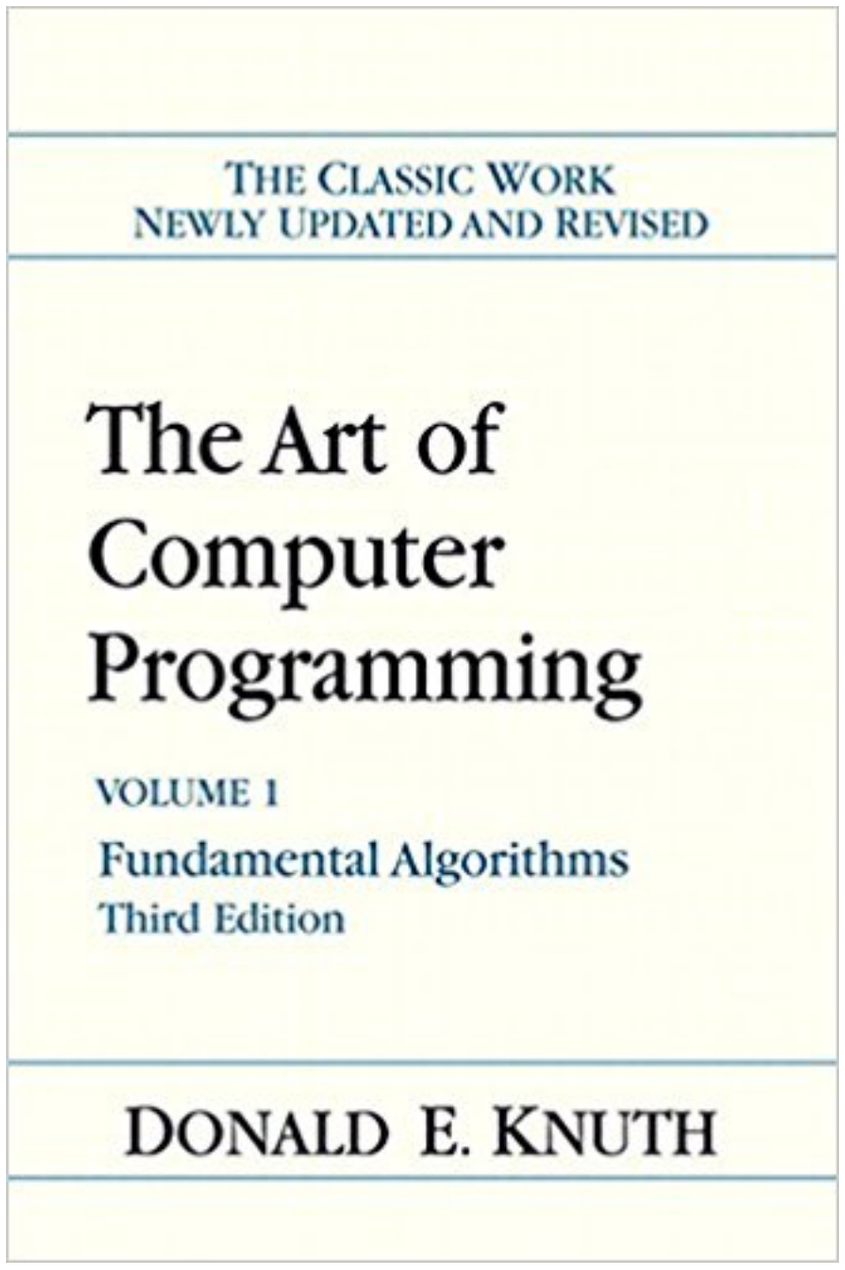
！？

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 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."

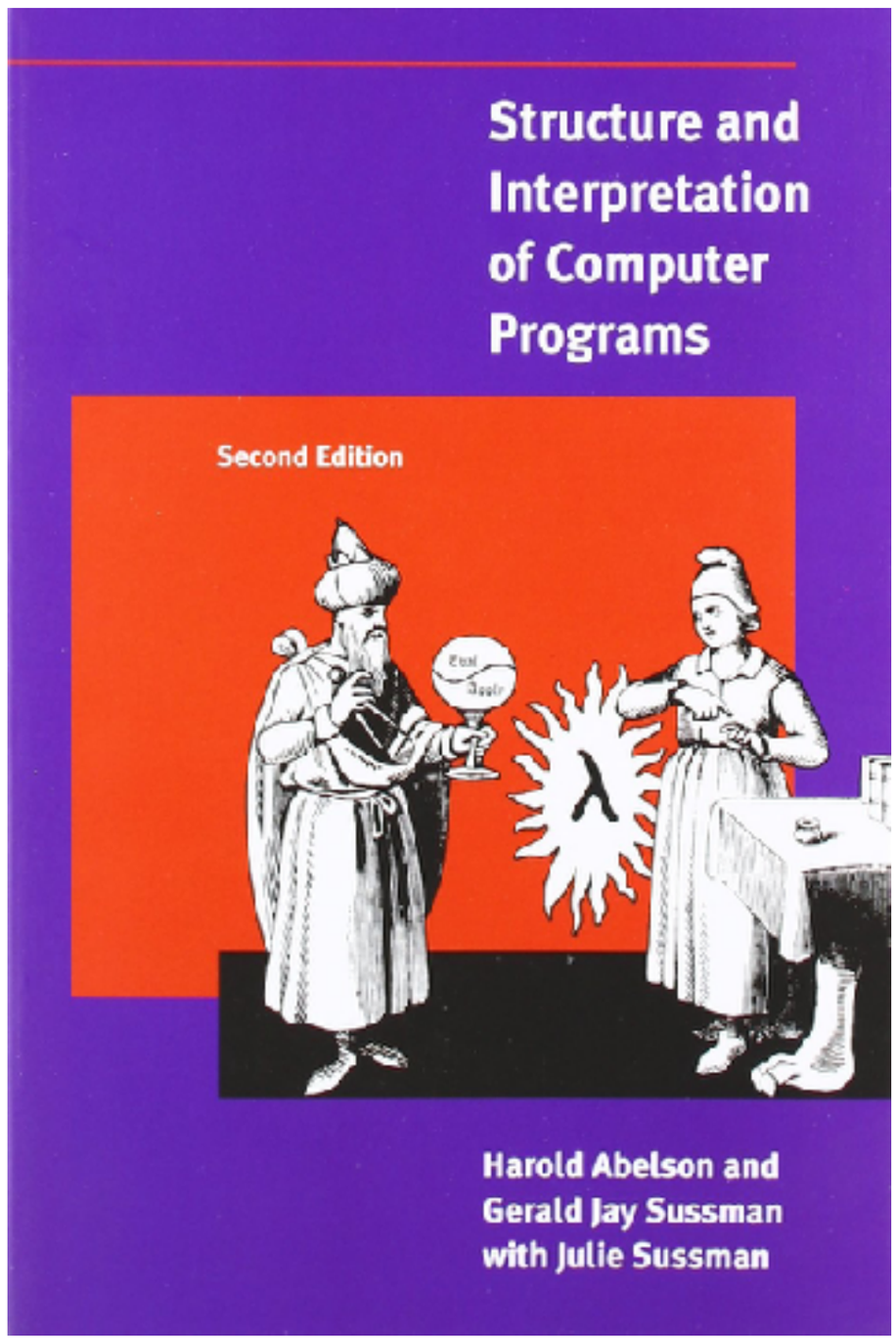
In this talk I shall try to explain why I think "Art" is the appropriate word. I will discuss what it means for something to be an art, in contrast to being a science; I will try to examine whether arts are good things or bad things; and I will try to show that a proper viewpoint of the subject will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

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

プログラミングは「魔法をかける呪文」

バイブル 通称"SICP"



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、設計してみましょう。
それらをただ遊ぶだけではなく、プログラムしてみましょう。(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思っていたんです。（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

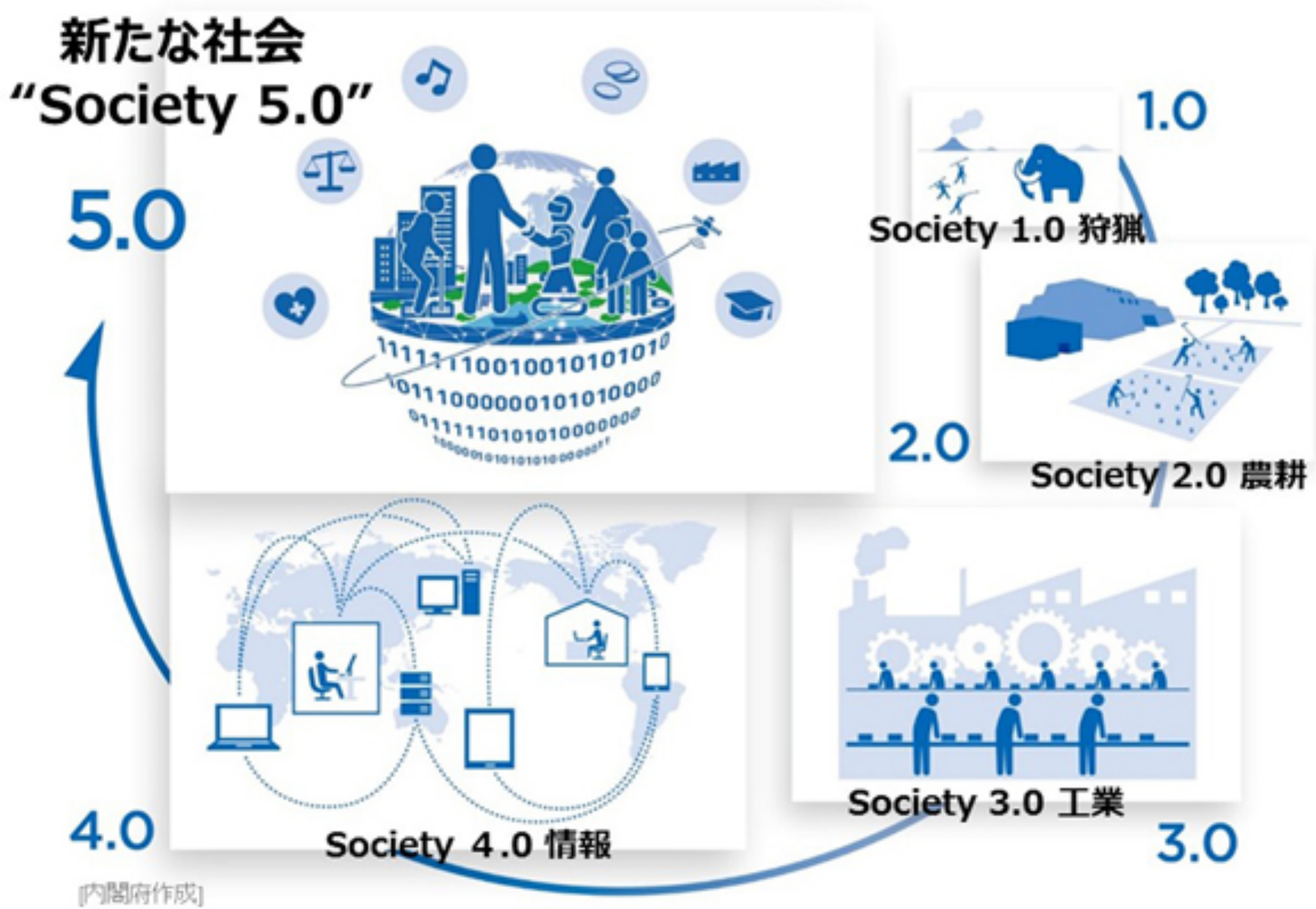
- パソコン、スマホ、タブレットPC
- インターネット (Webサービスやネットショップ)
- ゲーム
- 映画、音楽、CG、アニメ、ライブ
- 銀行や市場の基幹システム
- 自動車、航空、宇宙、原発、防衛など
- 電気・ガス・水道などのインフラ
- ビジネスや産業
- 物流・運輸
- 科学

Society 5.0

内閣府
Cabinet Office, Government of Japan

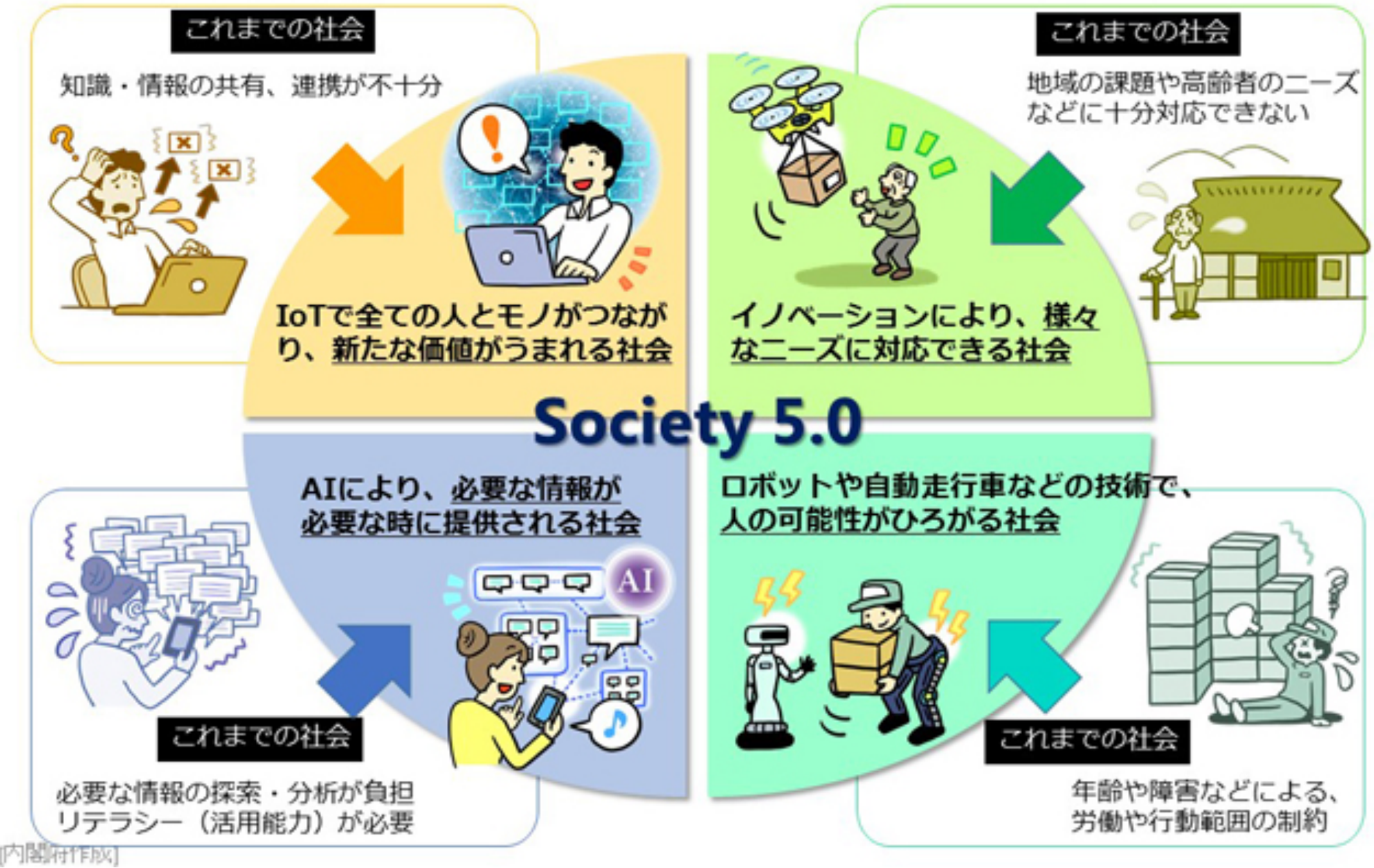
http://www8.cao.go.jp/cstp/society5_0/index.html

Society 5.0



[内閣府作成]

Society 5.0



[内閣府作成]

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

テスト

運用保守

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

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が…

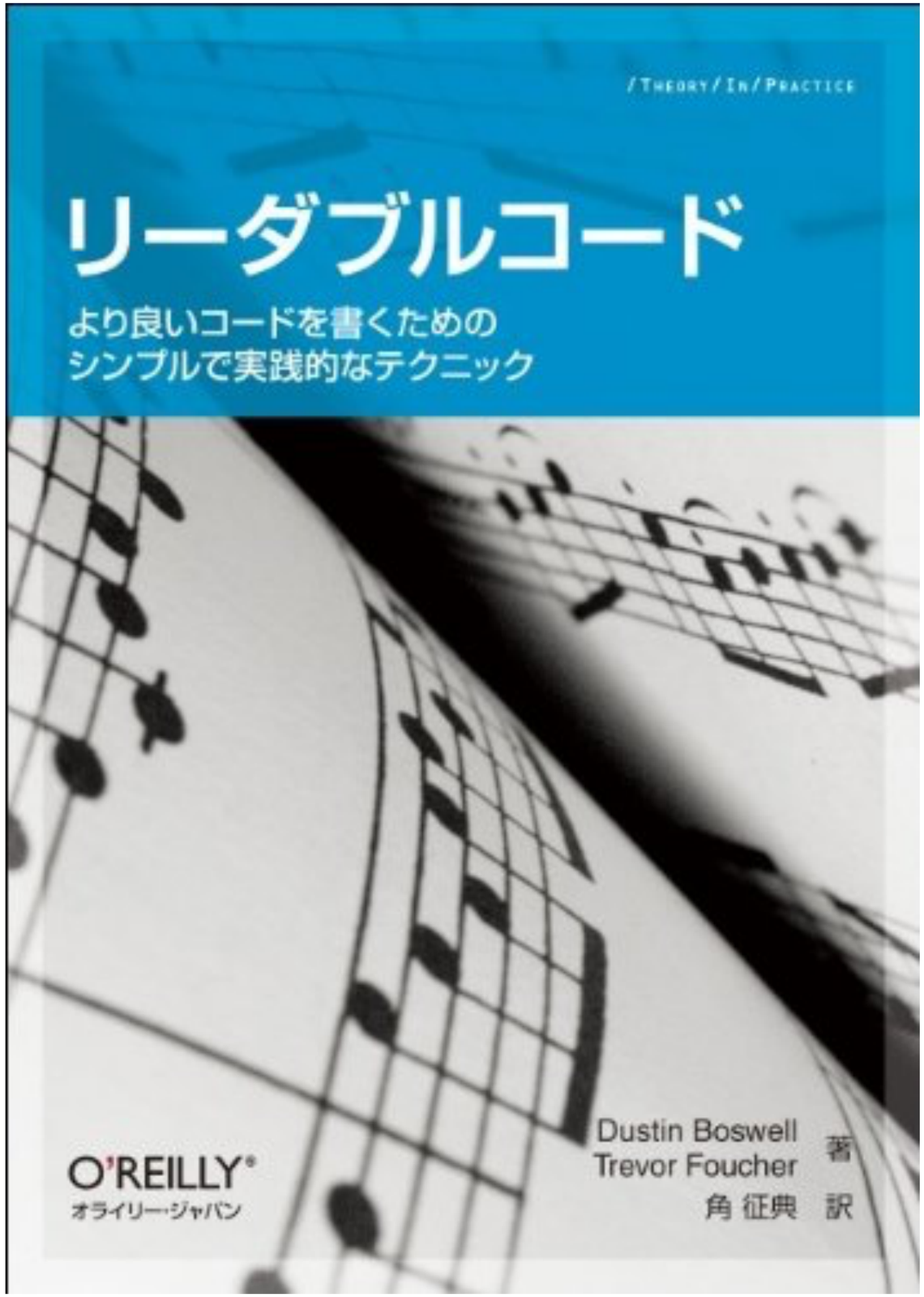
1. Pythonの少し発展的な部分：特に「クラス」
   → これはPythonに限らずJavaやrubyでも理解が必要
   「オブジェクト指向プログラミング」という設計思想

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
第2版
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
Framework
Component
Set Theory
Responsibility
UML
Use Case
Activity
State Machine
Modeling
Analysis
Design
Iteration
RUP
Architecture
XP
Refactoring
Pair Programming
Agile Manifesto
日経BP社


/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
Trevor Foucher
著
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

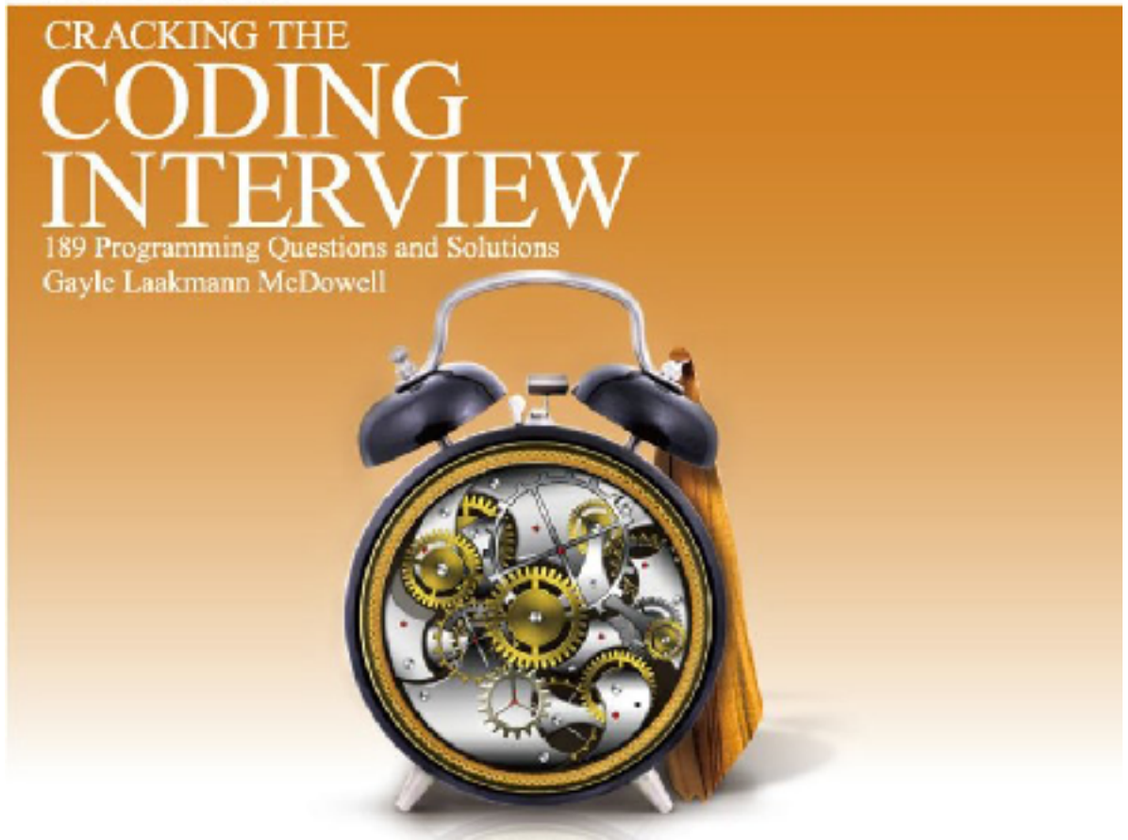
https://www.slideshare.net/iwiwi/wakate-web-14323842

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]



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。

マイナビ

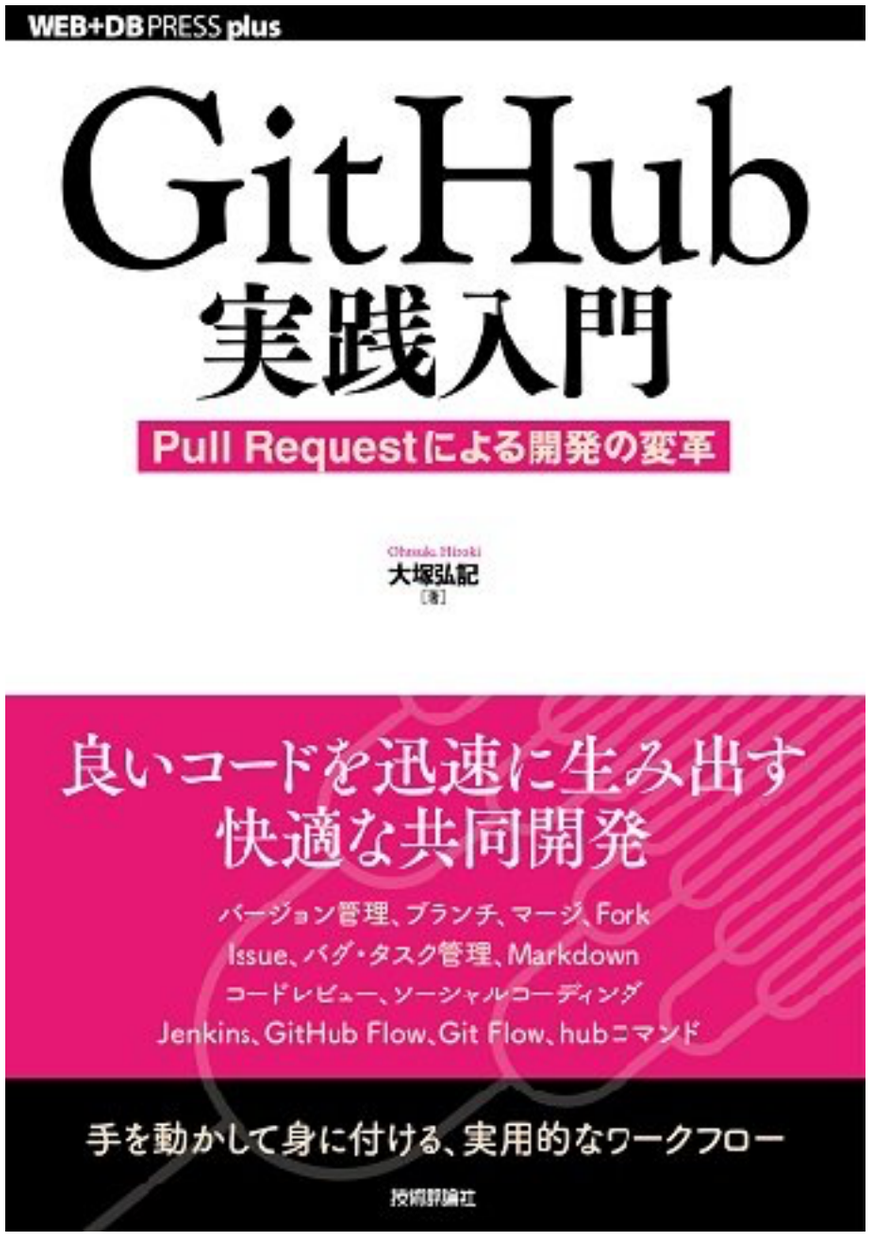
世界で闘う
プログラミング力を鍛える150問
―トップIT企業のプログラマになるための本―

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

マイナビ

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



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

edX

Introduction to Python: Absolute Beginner

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

Microsoft

Introduction to Python: Fundamentals

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

Microsoft

UDACITY

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

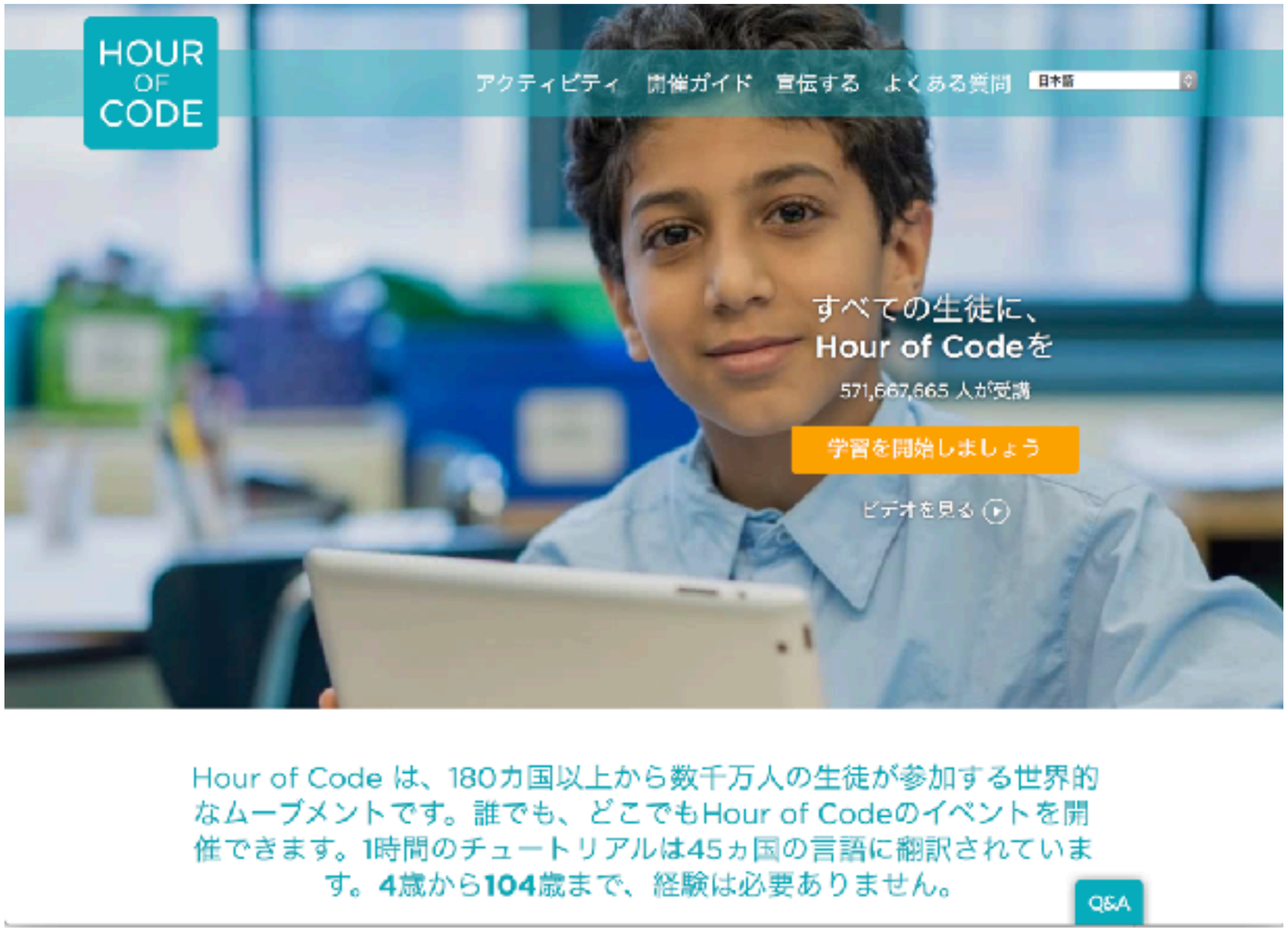
VRや3Dゲームに関心あるなら

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



Hour of Codeとは何ですか？

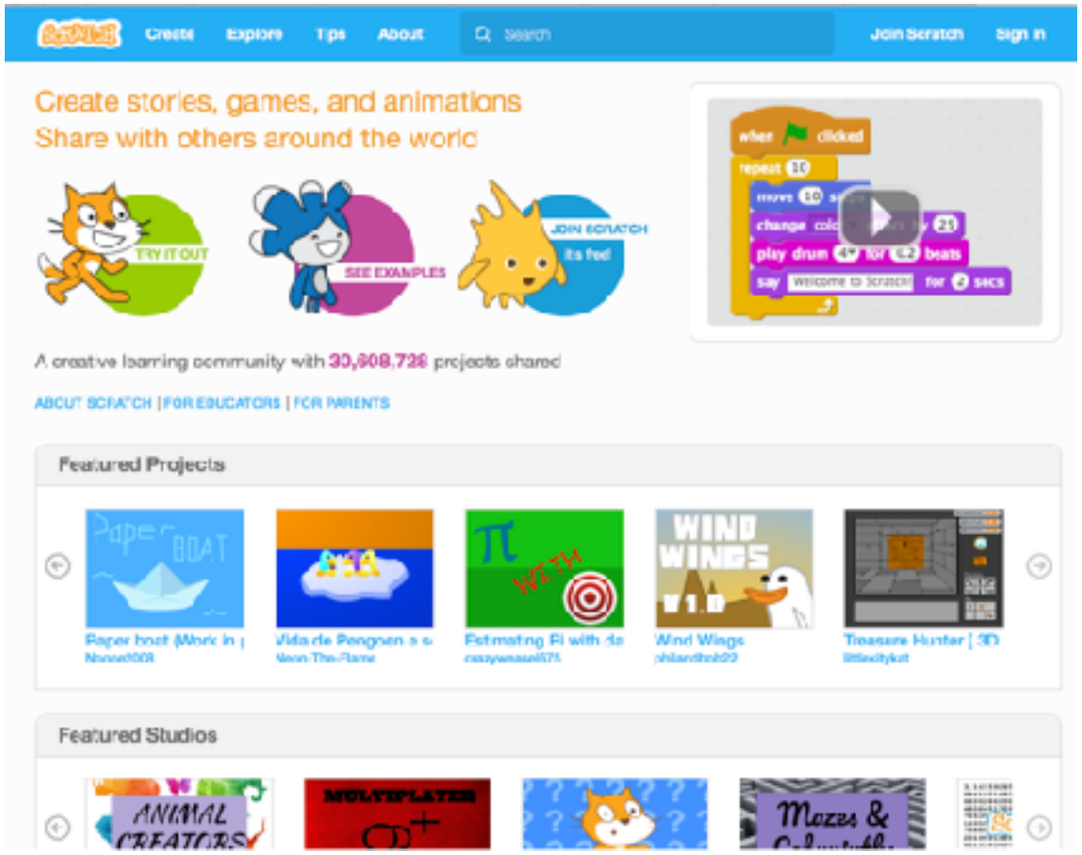
Hour of Code は、「コード（プログラミング）」の謎を解き明かし、全ての人が基本を学ぶことができることを示し、コンピューターサイエンス分野の幅広い参加者に向けて作成された、1時間のコンピューターサイエンス入門として始まりました。

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

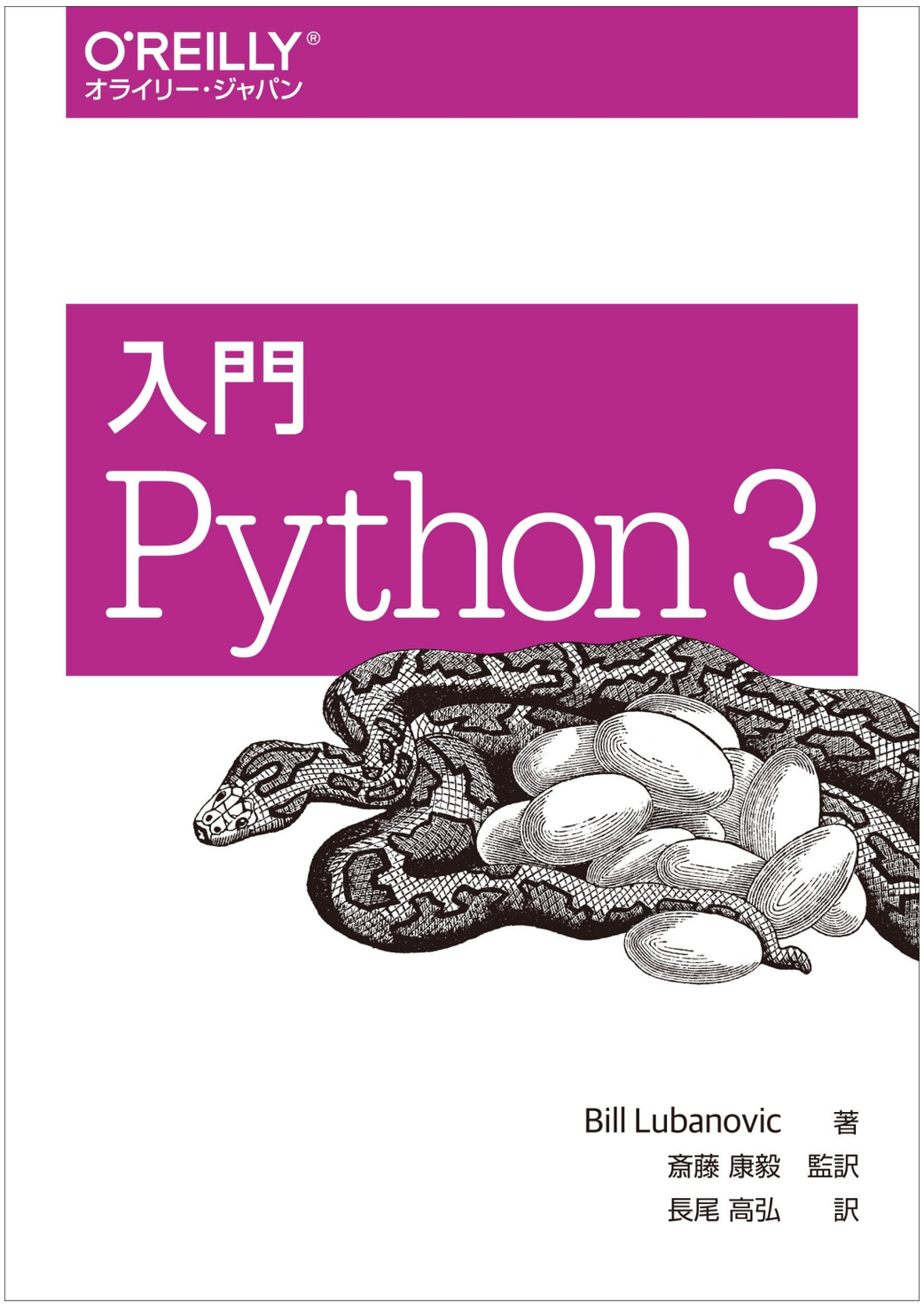
2. Hour of Code (コードの時間)

1時間でできるらしいので下記「初めてのコンピュータープログラムを書く」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

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

edX

Introduction to Python: Absolute Beginner

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

Microsoft

Introduction to Python: Fundamentals

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

Microsoft

UDACITY

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

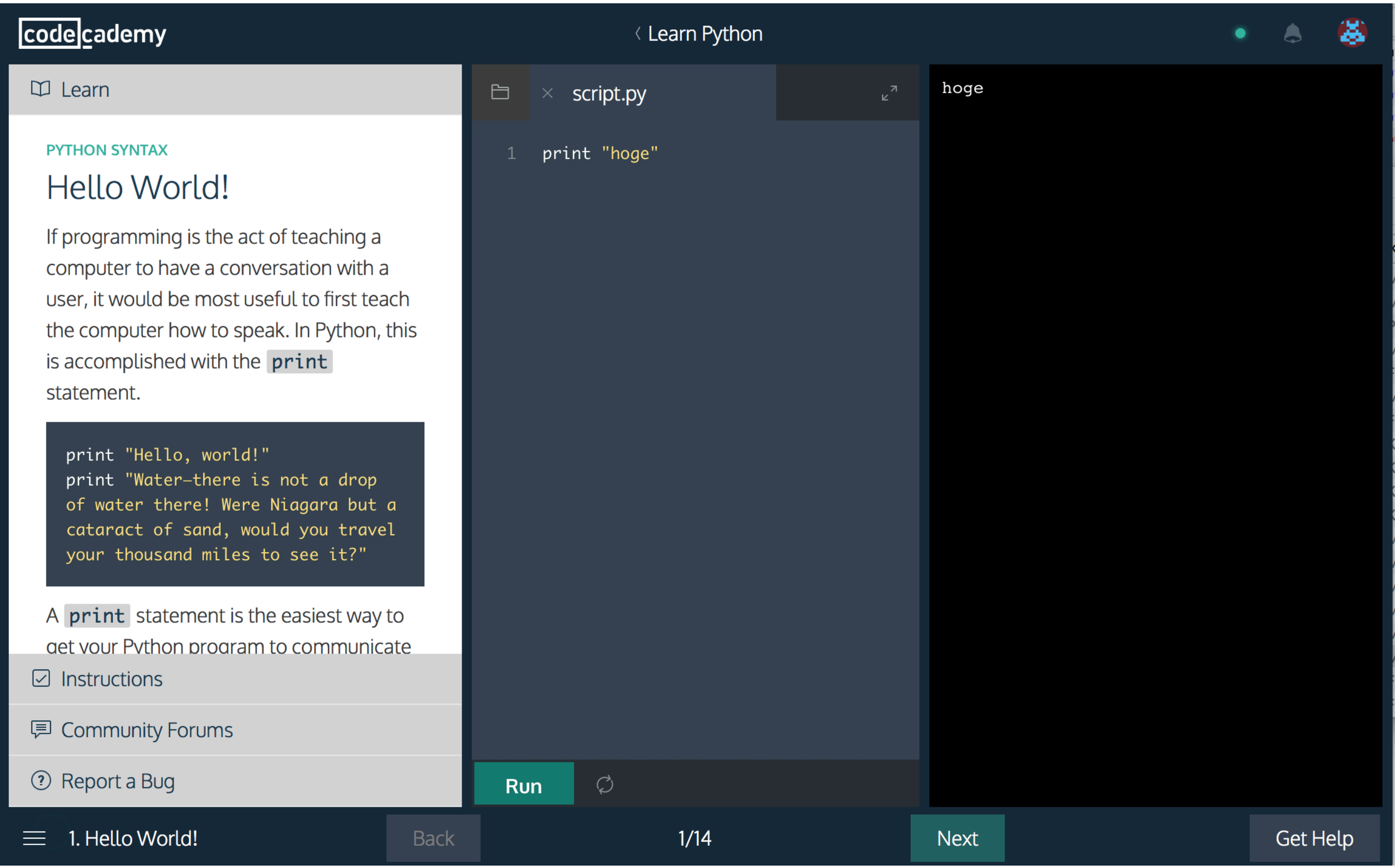
Exercise

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

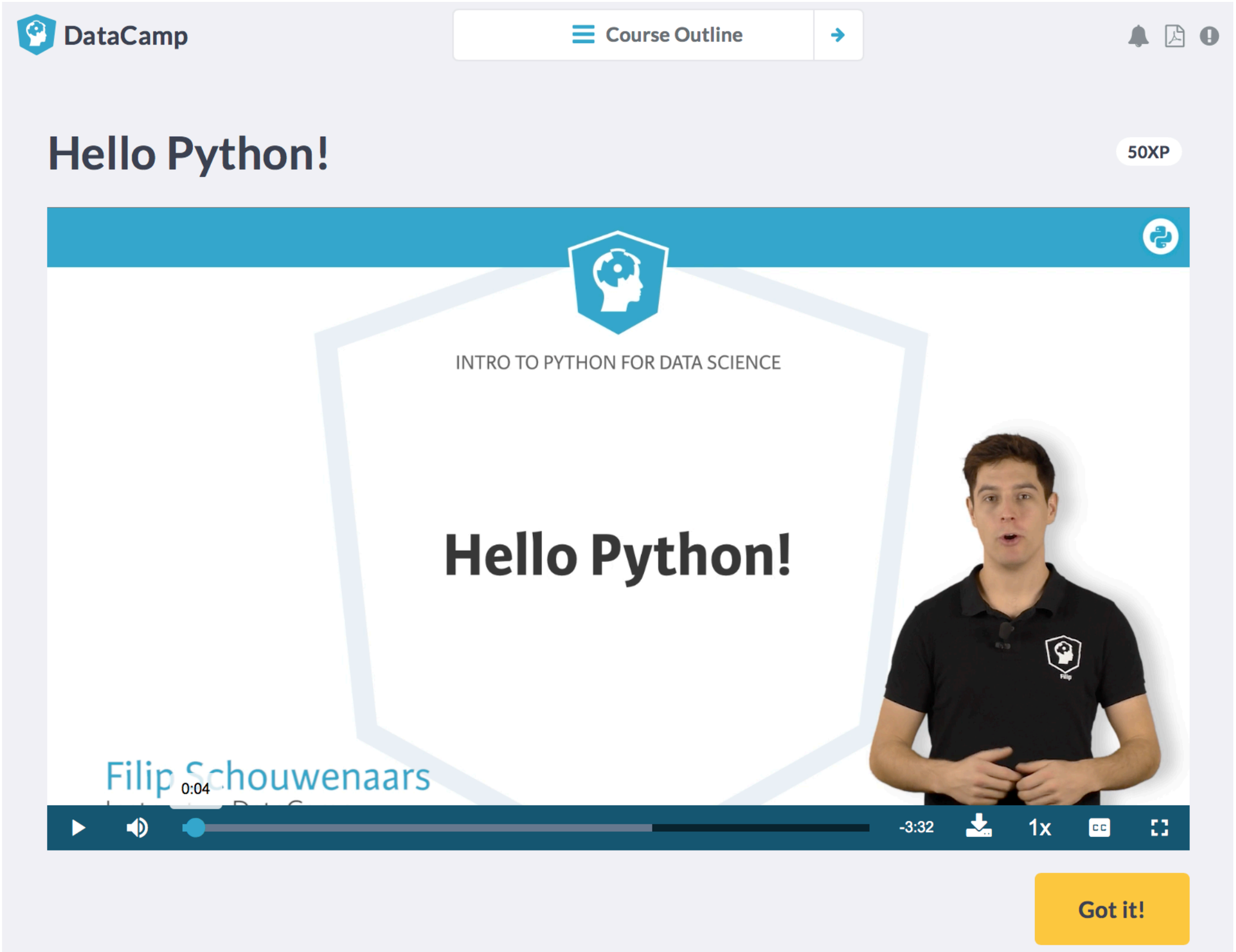
script.py | IPython Shell

```
1  print("Goodbye, World!")
```

- CodeAcademy (英語)



- Intro to Python for Data Science (英語)



- 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

```
print('Hello Python')
print("Hello Python")
```

script.py

```
# 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
```

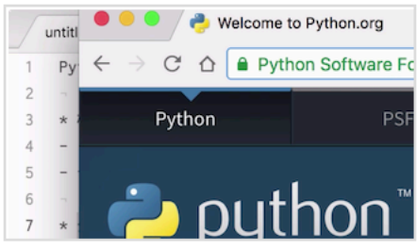
✕ エラー！！：必ず文字列はクォーテーションで囲む！

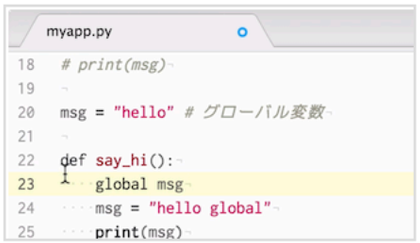
- ドットインストール

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。

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** レッスンを **5,186** 本の **3分動画** にて提供中

人気のレッスン

» すべてのレッスンを見る (349)

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 (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 (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 [Windows編] (全4回)

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て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