自然言語処理 Deep Learning

―導入―

https://satoyoshiharu.github.io/nlp/

導入の概要

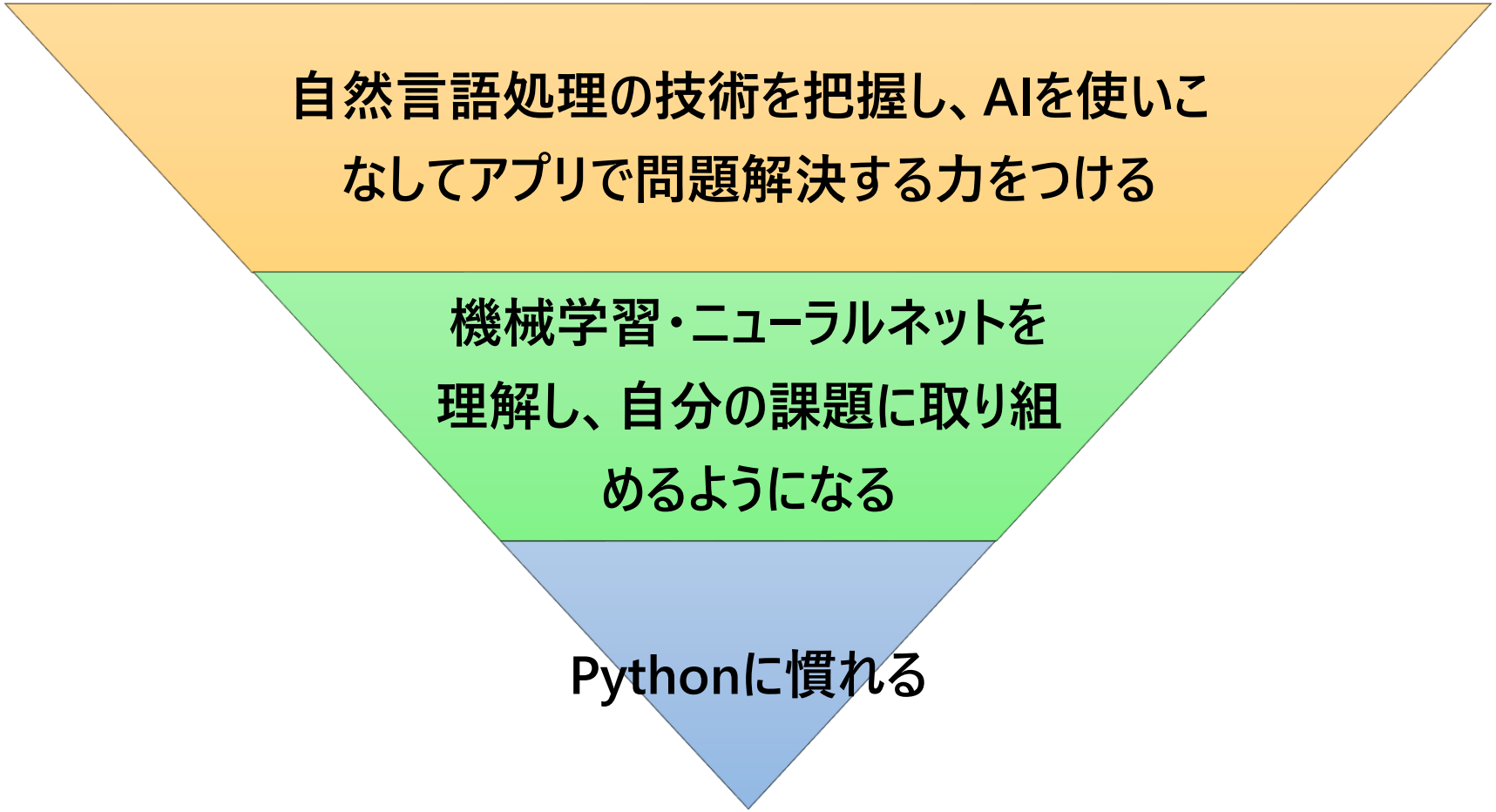
授業の目的

授業の概要

環境準備

この授業の目的は？

どこまでを目標にするかは各人の選択



導入ヒヤリング１

過去に機械学習、ニューラルネットワーク、あるいはディープラーニングの授業をとったか、自分で本を勉強したことがありますか？

人工知能って何に使えるの？
（ChatGPT以前の話）

制御、診断

- 異常検知
  - 工場の製造部品や、野外の建造物などを、カメラ等でモニターし、不良品や異常を検知する。
- ロボット制御
  - カメラやセンサーの入力信号から、次の行動を決定し、モーター駆動機構へ命令する。
  - 自動運転。
- 病理診断
  - レントゲン写真から病気を見つける。

予測

- 株価予測
  - 過去のデータから、今後の株価を予測し、売買を実行する。
- レコメンデーション
  - あるユーザの過去の購買履歴やブラウザ履歴から、その人の関心を予測し、関連する広告を選んで表示する。
- 顧客属性予測、行動予測
  - あるユーザの過去の購買履歴や投稿内容から、優良顧客かクレームユーザになりそうかを予測する。

情報変換・加工（して人を助ける）

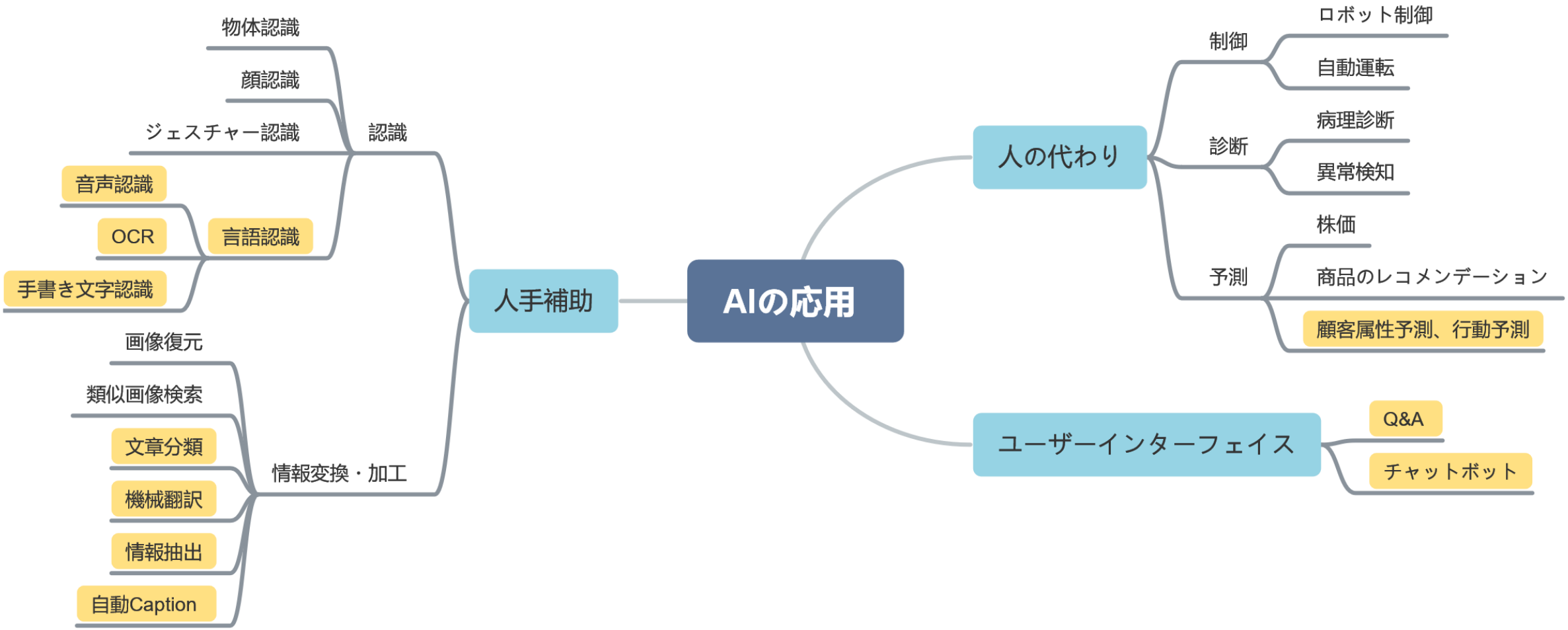
- 画像復元
  - 人工衛星の地表映像から、より鮮明な地表映像を復元する。
- 類似画像検索
  - 指定された画像と類似したものを、提示する。
- 文章の分類
  - 文章を読み、カテゴリーに分ける（例：商品コメントを、ポジティブとネガティブに分けて、ユーザの評価を客観視する）。
  - 検索したページに含まれる記事を読み、カテゴリーに分類し、キュレーションサイトとして提供する。
- 機械翻訳
  - ある言語の文章を、他言語に翻訳する。
- 情報抽出
  - 商品へのコメントから、重要なキーワードを抽出し、評判を分析する。
- 画像キャプション
  - 画像に、表現されている物体を文章化する。

認識（して後続の処理に提供する）

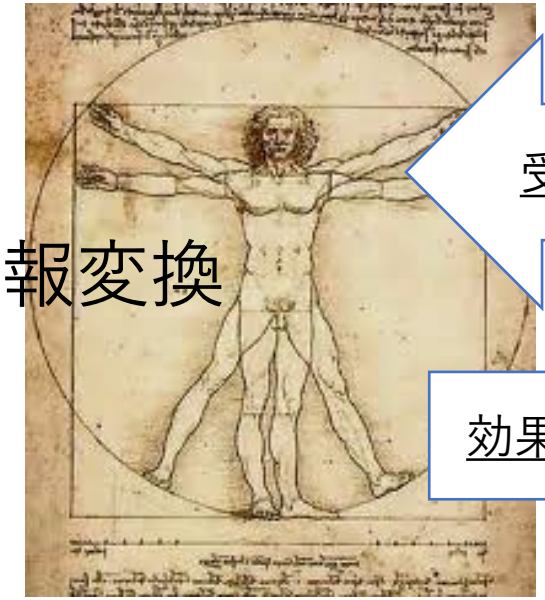
- 物体認識
  - カメラ画像から、物体を認識する（例：自動車の車載カメラで、対向車や横断歩行者を認識）。
  - 手指を認識し、空中のジェスチャーを認識する。
- 顔認識
  - ある人の顔と、他の人の顔と区別し、個人を特定する。
- ジェスチャー認識
  - 加速度センサー、ジャイロスコープセンサーの信号から、装置の動きを計算し、装着者のジェスチャーを認識する。
- 音声認識
  - マイク入力音声信号から、話された文章を認識する（例：Siri、Echo）。
- OCR
  - 印刷された文字を認識し、文字コードに復元する。
  - 外国で、メニューをカメラに映して、自国語に翻訳させる。
- 手書き文字認識
  - 手書き文字を認識する。

インターフェイス

- Q&A
  - サポートへの質問に対し、回答を生成する。
- チャットボット
  - ユーザのタイプした文章に、適当な回答をして、対話する。


AIの応用
人手補助
認識
物体認識
顔認識
ジェスチャー認識
言語認識
音声認識
OCR
手書き文字認識
情報変換・加工
画像復元
類似画像検索
文章分類
機械翻訳
情報抽出
自動Caption
人の代わり
制御
ロボット制御
自動運転
診断
病理診断
異常検知
予測
株価
商品のレコメンデーション
顧客属性予測、行動予測
ユーザーインターフェイス
Q&A
チャットボット

人は膨大な量の情報変換をしている。人工知能はそれを模倣する。



人

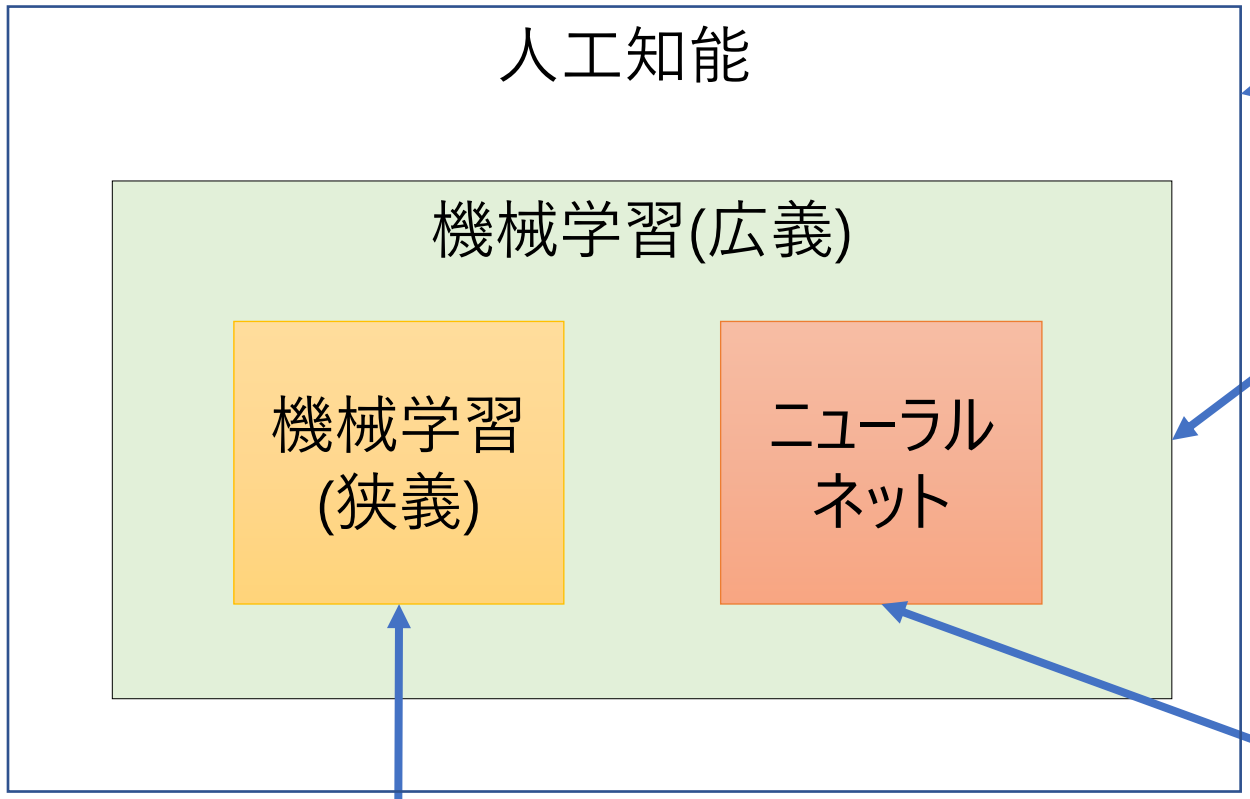
情報変換

受容器官: 10億ビット/秒

効果器官: 1千万ビット/秒

環境

人工知能、機械学習、ニューラルネットの違い

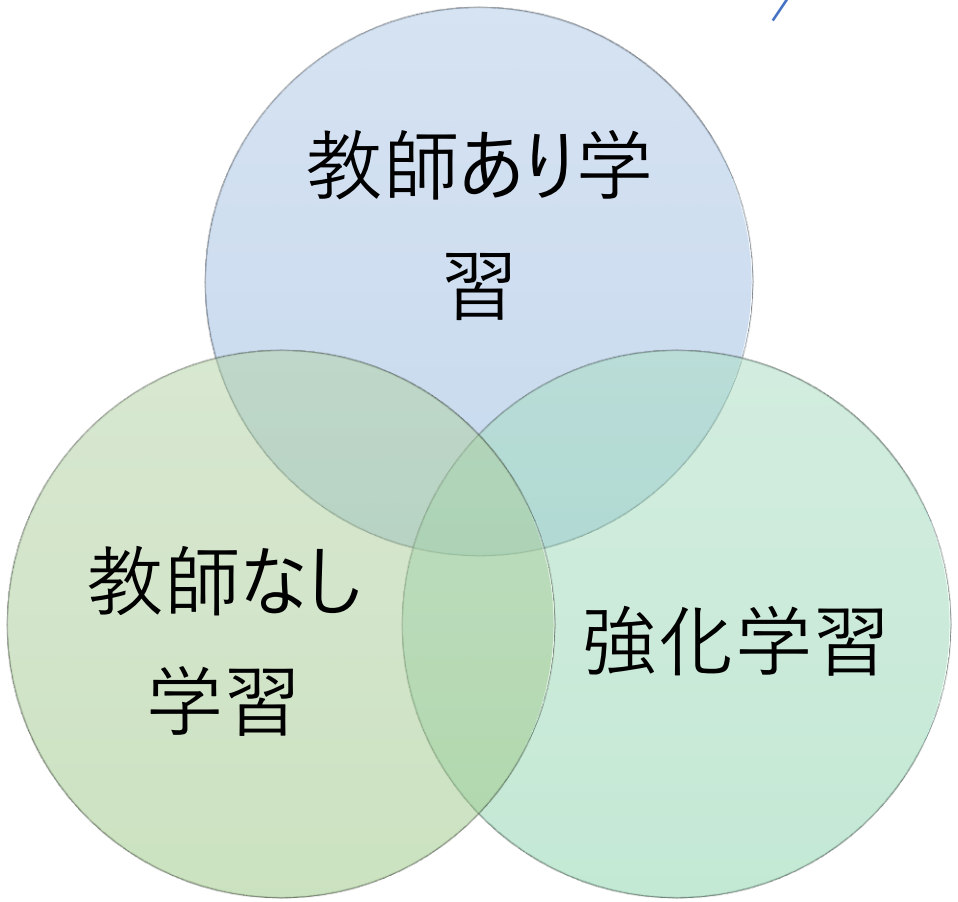


- 人の知能を模倣する。
- データでトレーニングして予測する

- 推論根拠となる入力特徴を人が定義する
- 推論根拠がわかりやすい

- 推論根拠となる入力特徴を機械が学習する（複雑な対応関係でもOK）
- 推論根拠がわかにくい

学習の観点からの分類

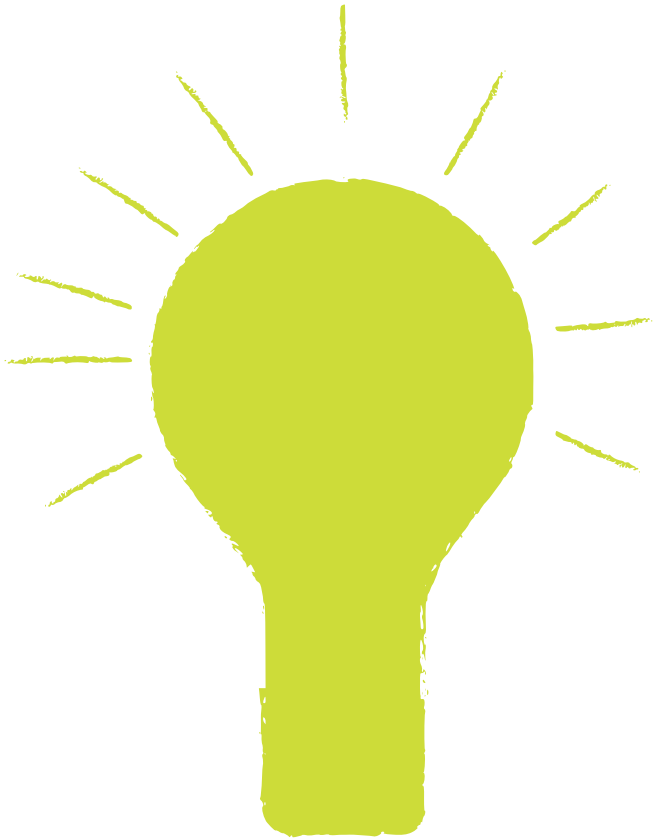


認識、変換、診断、予測

ゲーム、制御

ＡＩの応用＝人がやっているような情報変換を模倣したもの

応用は無数にありうる。
その限界はあなたの想像力のみ。

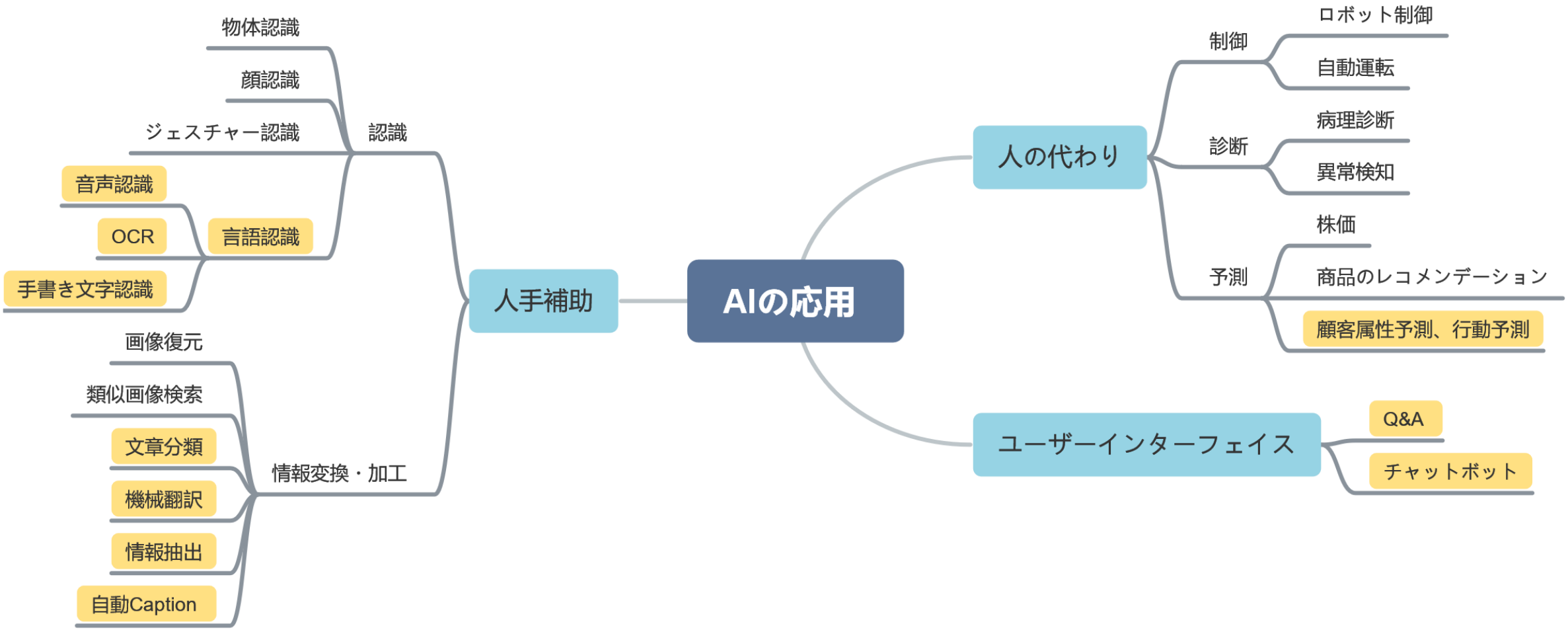


参考

- ディスカバリーチャンネル、AI革命～激変する世界を探る～Part1 言語の理解/機械との議論/路上を走行する人工知能
- Deep Learning入門：Deep Learningでできること
- Deep Learning入門：今Deep Learningに取り組むべき理由
- 自然言語処理 05 深層学習×自然言語処理
- Deep Learningで行う自然言語処理入門

自然言語処理って？

先の事例の中で、信号・画像でなくて、
人がしゃべったり、書いたりした言葉を
対象にしたものは？

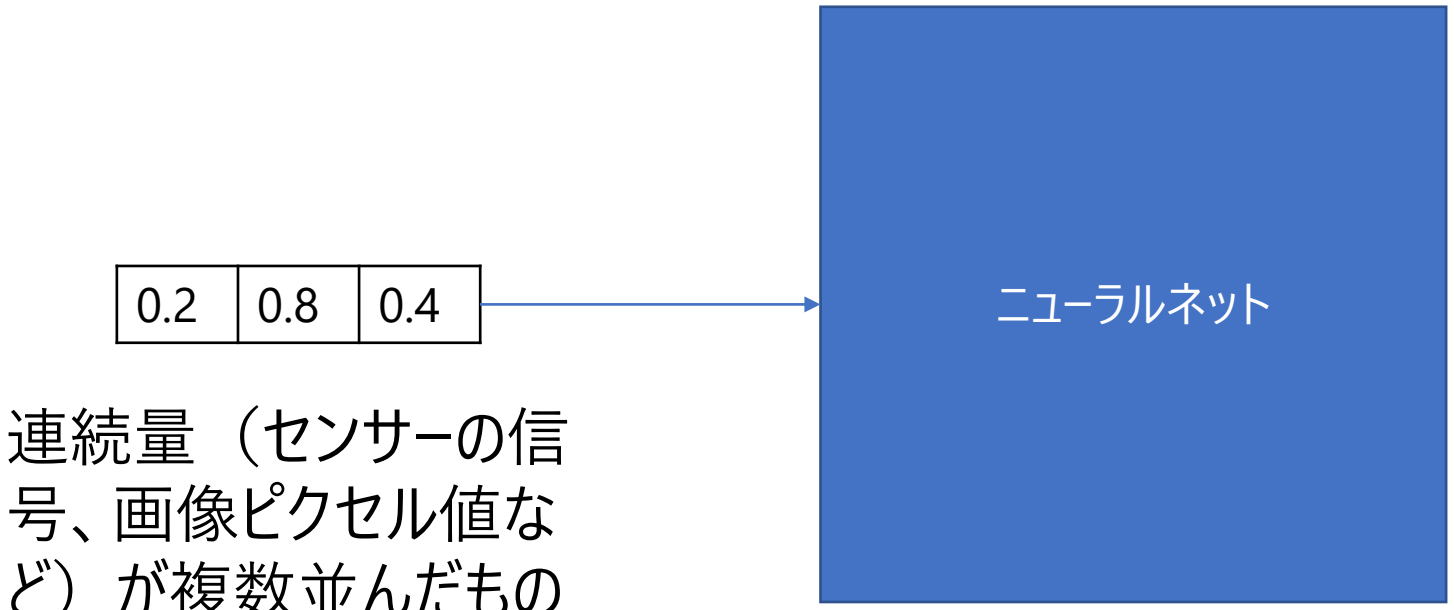

AIの応用
人手補助
認識
物体認識
顔認識
ジェスチャー認識
言語認識
音声認識
OCR
手書き文字認識
情報変換・加工
画像復元
類似画像検索
文章分類
機械翻訳
情報抽出
自動Caption
人の代わり
制御
ロボット制御
自動運転
診断
病理診断
異常検知
予測
株価
商品のレコメンデーション
顧客属性予測、行動予測
ユーザーインターフェイス
Q&A
チャットボット

センサーの信号処理、画像処理と比べて、自然言語処理のDeep Learningは、どう違う？

連続量と離散量

- 連続量
  - 例
    - 身長：170cmと170.11cmとの間に無限に取りうる値がある。
    - 体重
    - カメラから写っている物体までの距離
    - 画像のあるピクセルのRedチャネルの量(floatの場合)
    - 加速度センサーのX軸の反応量
    - スピーカーの音量
    - ・・・
- 離散量
  - 例
    - さいころの目： 1 から 6 の間の値を飛び飛びでとり、それらの間の値を取りえない。
    - 何月か
    - 学生番号
    - トランプのマーク
    - 人の性別
    - コロナ検査で陽性か陰性か
    - ・・・

ニューラルネットの入力は通常連続量のベクトル



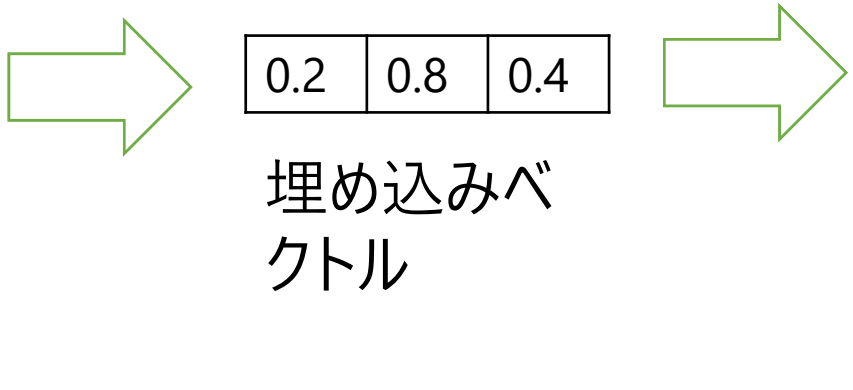
連続量（センサーの信号、画像ピクセル値など）が複数並んだもの

自然言語の処理対象は離散的なので、データの変換が必要

- 文字：あ、い、山、川、…
- 単語：今日、明日、太郎、花子、…
- 文：今日は晴天です。明日は木曜日です。週末が楽しみです。…

埋め込みベクトル

- 文字：あ、い、山、川、…
- 単語：今日、明日、太郎、花子、…
- 文：今日は晴天です。明日は木曜日です。週末が楽しみです。…



LLMのインパクト

調査論文 “GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models” から読み取れること

LLMは自然言語処理だけでない

- LLMs can process and produce various forms of sequential data, including assembly language, protein sequences and chess games, extending beyond natural language applications alone.
- LLMは、自然言語に限らず、どんな系列情報をも処理し、生産できる。プログラム、遺伝子情報、ゲームの手など。

LLMは急激に成長している

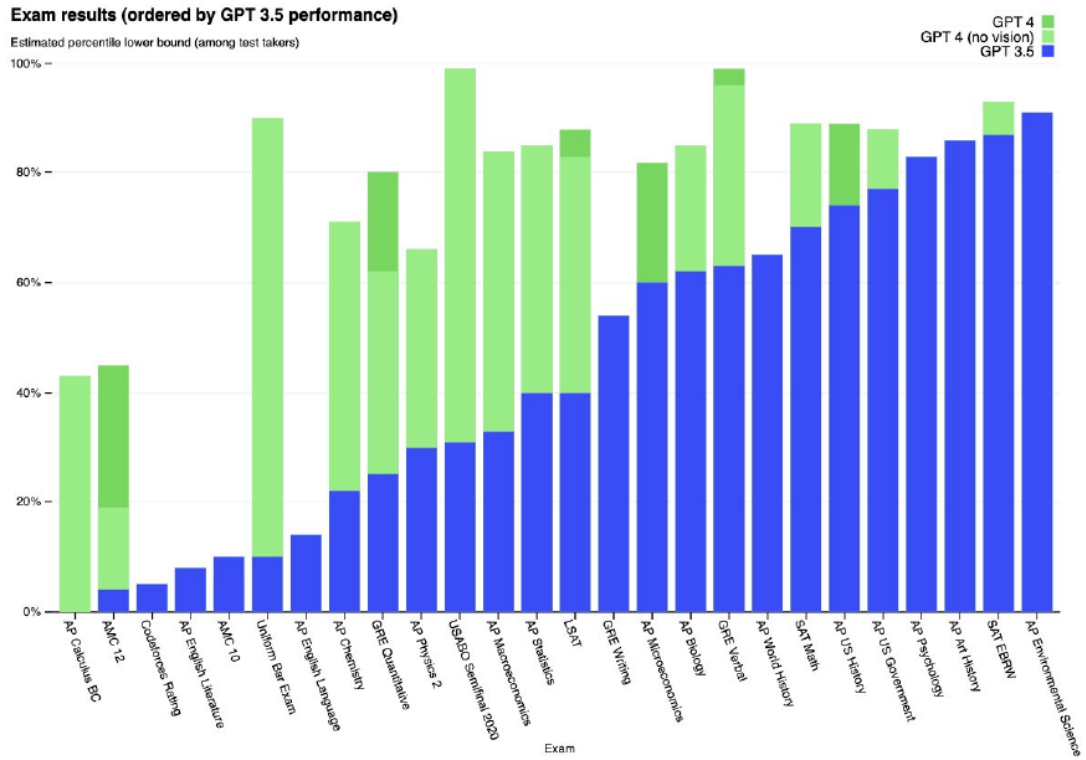
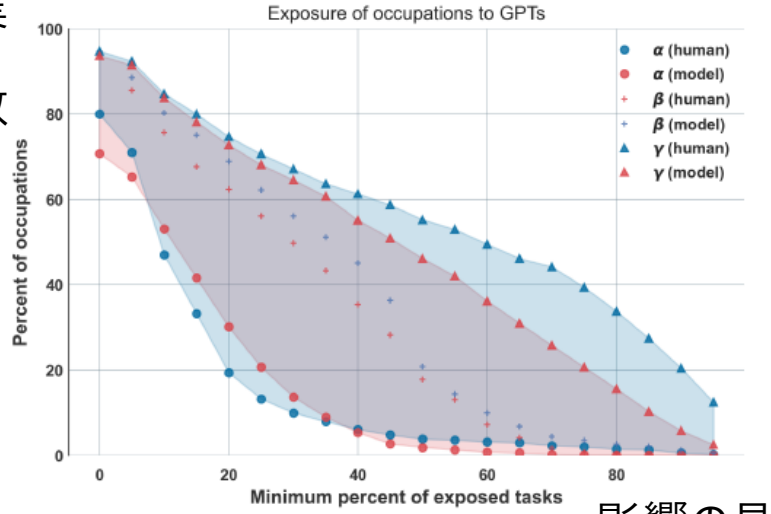


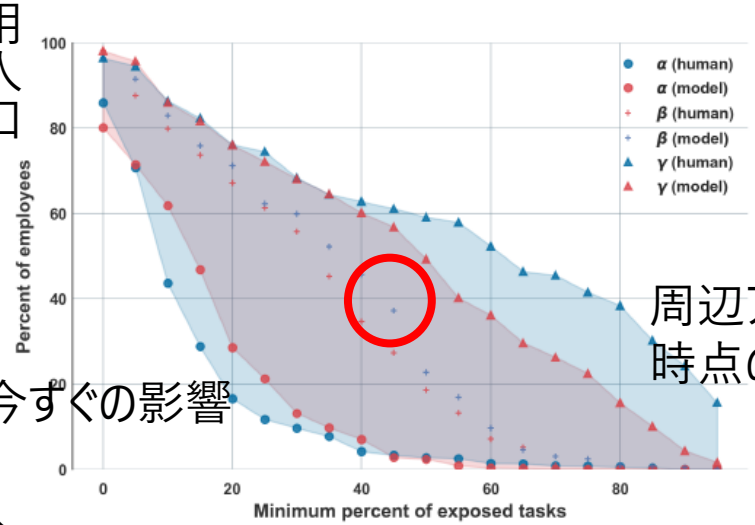
Figure 1: To get a sense of how quickly model capabilities are progressing – consider the jump in exam performance between GPT-3.5 and GPT-4 (OpenAI, 2023b).

40%の人が仕事の45%に影響を受ける…など

職業の数



雇用人口



今すぐの影響

周辺アプリがそろってきた時点の影響

影響の具合

Figure 3: Exposure intensity across the economy, displayed on the left in terms of percent of affected occupations and on the right as percent of affected workers. The distribution of exposure is similar across occupations and across workers, suggesting that worker concentration in occupations is not highly correlated with occupational exposure to GPTs or GPT-powered software. We do however expect that it could be more highly correlated with investment in developing GPT-powered software for particular domains.

スキルと影響受け具合の相関：

読み書き、そろばん、プログラミングなどのスキルは、強く影響を受ける

今すぐの影響

周辺アプリがそろってきた時点の影響

Basic Skill	α (std err)	β (std err)	ζ (std err)
All skill importance scores are normalized to be between 0 and 1.			
Constant	0.082*** (0.011)	-0.112*** (0.011)	0.300*** (0.057)
Active Listening	0.128** (0.047)	0.214*** (0.043)	0.449*** (0.027)
Mathematics	-0.127*** (0.026)	0.161*** (0.021)	0.787*** (0.049)
Reading Comprehension	0.153*** (0.041)	0.470*** (0.037)	-0.346*** (0.017)
Science	-0.114*** (0.014)	-0.230*** (0.012)	-0.346*** (0.017)
Speaking	-0.028 (0.039)	0.133*** (0.033)	0.294*** (0.042)
Writing	0.368*** (0.042)	0.467*** (0.037)	0.566*** (0.047)
Active Learning	-0.157*** (0.027)	-0.065** (0.024)	0.028 (0.032)
Critical Thinking	-0.264*** (0.036)	-0.196*** (0.033)	-0.129** (0.042)
Learning Strategies	-0.072* (0.028)	-0.209*** (0.025)	-0.346*** (0.034)
Monitoring	-0.067** (0.023)	-0.149*** 0.020)	-0.232*** (0.026)
Programming	0.637*** (0.030)	0.623*** (0.022)	0.609*** (0.024)

Table 5: Regression of occupation-level, human-annotated exposure to GPTs on skill importance for each skill in the O*NET Basic skills category, plus the programming skill. Descriptions of the skills may be found in Appendix B

GPT-4時代のエンジニアの生存戦略

https://qiita.com/lazy-kz/items/e4932f1a90c2a7986ef5

- GPT-4は適切なプロンプトを与えれば、ほとんどのエンジニアを凌駕するコーディング能力を持っています。また、人間では真似できないコーディング速度や事務処理速度を併せ持つ非常に優秀な存在です。全エンジニアに「年収10万ドル以上の市場価値があるスキルを持った優秀なメンター/部下」が常についているイメージです。
- そのため、**AIとのペアプログラミングは今後、全てのエンジニアにとって必須技能となるでしょう。**
- また、それに加えて、**AIがカバーできる領域が広がったことにより、エンジニアには「コードを書くこと」自体では無く、「ビジネスサイドが実現したいこと」を実現する能力がより強く求められるようになります。**

AIとのペアプログラミング（今見えている利用法）

プログラムコードの解説をさせたり、コメントを追加させる

Bugを探させたり、改善を助言させる

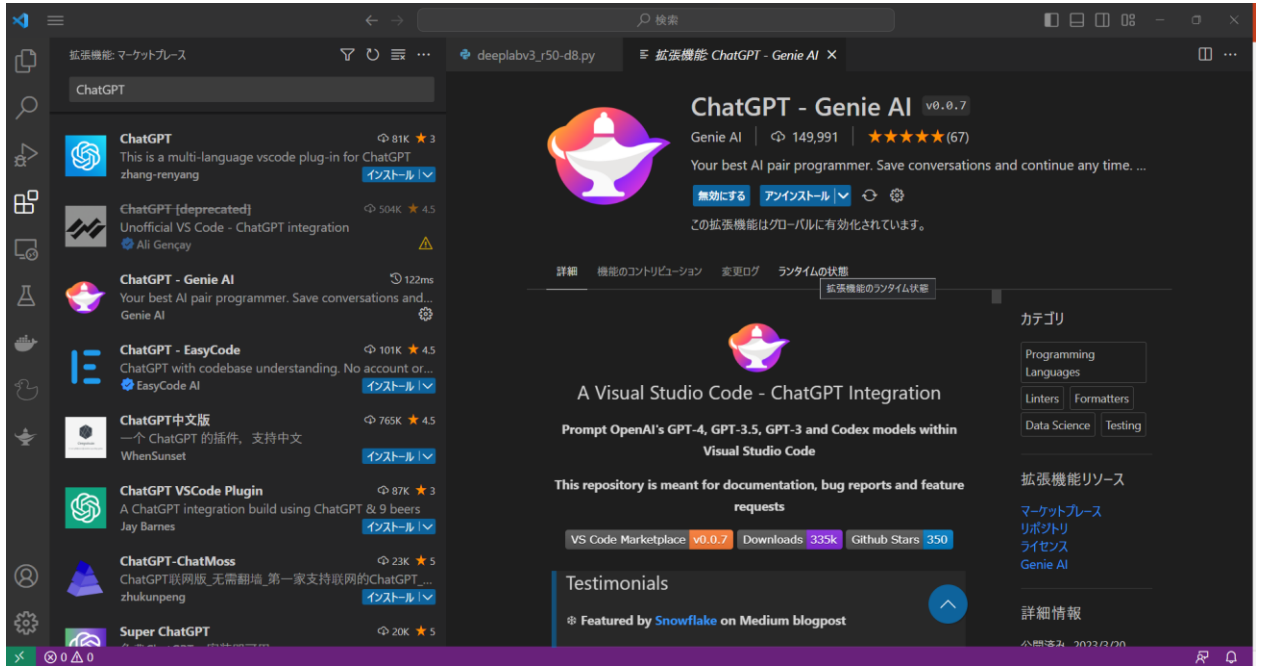
文章や箇条書きからコードを生成させる

コードを最適化する

サンプルテストデータを作ってもらう

AIとのペアプログラミング

- GitHub Copilot：10$/Month
- VSCodeの拡張、ChatGPT Genie：無料



実現能力？

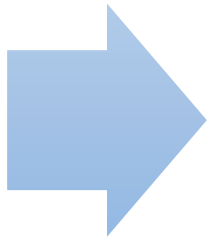
この授業では、ChatGPTに至るまでの技術要素を学びますが、まずはどういう技術がどういう**応用**に使えるのかという点の理解に重きを置くといいです。

さらに、サンプルコードがあるトピックは、コードの全体的な構成をみて、どういう**仕組み**でという面を押えてください。すると、自分である課題を持った時に、クラスライブラリあるいはAPIを使って、アプリを自分で**書ける（AIとのペアプログラム）**状態になります。それが「**実現する力**」となります。

この授業で何をする？

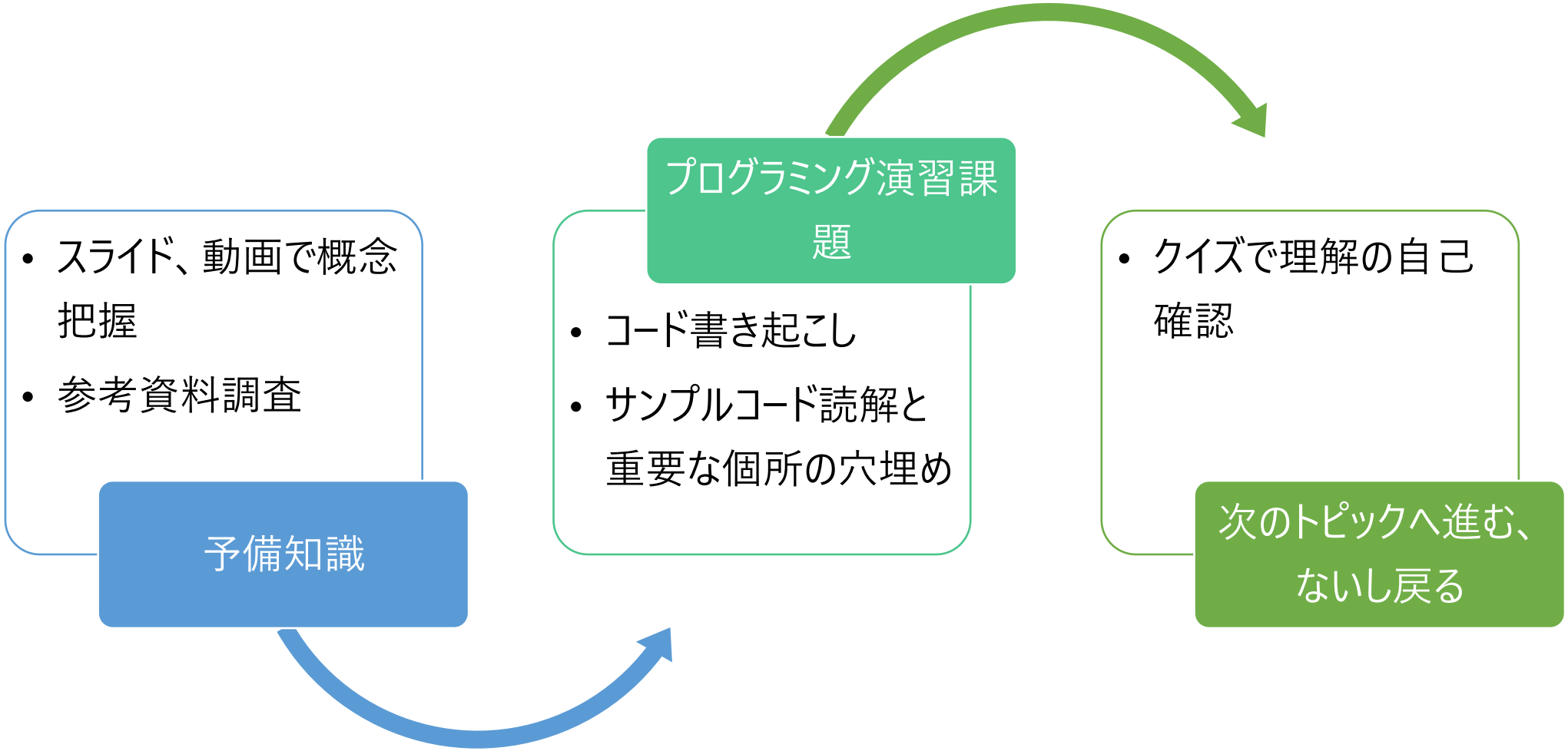
学習ロードマップ

Pythonの復習は、一斉授業。



高校数学の復習から、機械学習、Deep Learning、ChatGPTなど最先端の高度なトピックまで、選択式。

選択トピックをやる際の学習の流れ



教科書はありません

- いい教科書がない。
  - 既存の教科書は、大学の教養課程の数学（線形代数、解析）を前提としていて、いきなり難しい数式が出てきて説明がない。
  - この分野は変化が激しいため、本になったものは、たいてい、書籍になった時点で、実は、扱う内容が、すでに時代遅れです。

＝＞

- 授業の教材として、主に、最近の技術を扱ったサンプルや課題を利用し、まず手を動かし経験することを重視します。ハンズオンというやつ。

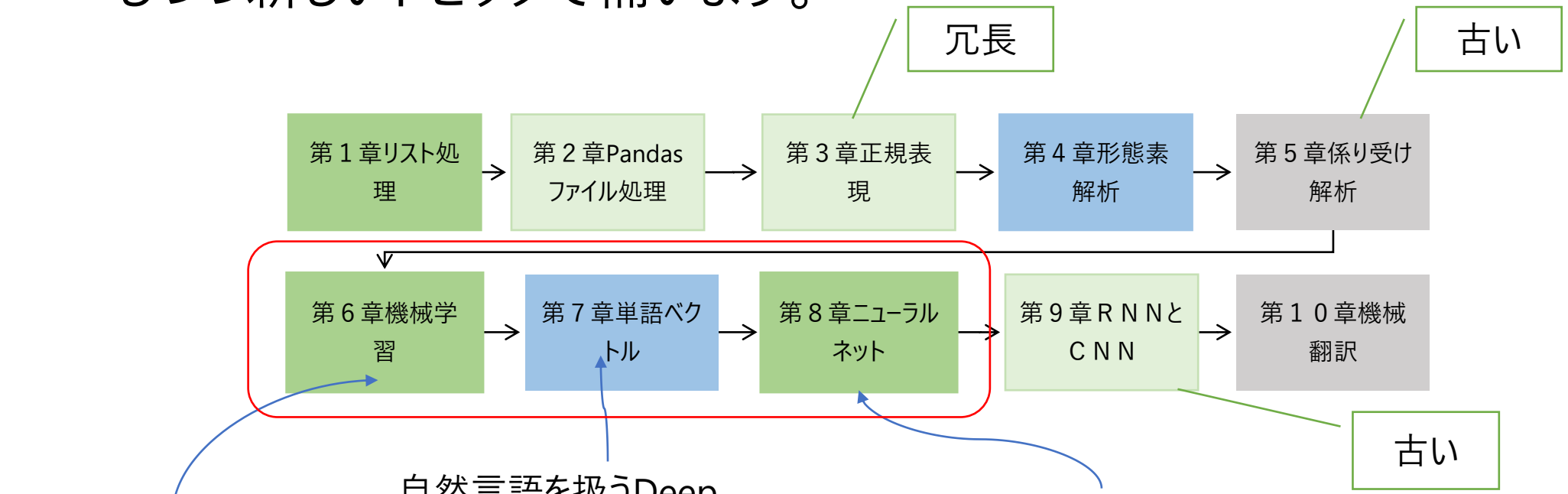
教材について

- 前提知識は、必要に応じて、スライド・動画で解説します。
  - ネットに十分質の高い教材がフリーでのっている場合（Python解説など）、あるいは特定分野専門の先生の解説でいいのがあれば（数学とか）、それをご紹介します。
  - 講師のほうがうまく説明できると思ったものは、講師の作った解説動画を紹介します。スライドの内容に言葉で説明を加えたものです。スライドをただ読むよりも、耳も使ったほうが記憶に残るので、スライドだけより動画を活用してください。

- 演習課題は、「言語処理１００本ノック」https://nlp100.github.io/ja/ という課題集から選んだものを主に利用します。さらに、１００本ノックではネタ的にプアあるいは古いトピックは、PyTorchのチュートリアル https://pytorch.org/tutorials/ サンプルなど他から補充します。

「言語処理１００本ノック」課題集

- いいところだけ活用し、冗長なところは適宜スキップし、古いところはスキップしつつ新しいトピックで補います。



注：１００本ノックの課題集の章立てと学習ロードマップとの関係

- 学習ロードマップに沿って学習を進めてもらい、その過程で適宜１００本ノックの課題を演習として利用します。が、１００本ノックは課題の都合によって順序付けられています。以下のことにご注意ください。

  - 機械学習のクラスタリング、次元圧縮は、１００本ノックの課題集「単語ベクトル」の中で扱われていて、単語ベクトルをデータにして動かしてみるという位置づけになっています。
  - １００本ノックの「単語ベクトル」は、すでにあるものを利用するという課題になっています。そのため、単語ベクトルの課題をやってからニューラルネットの課題をやるという順序になっています。しかし、本当の応用を組むには、応用に最適な自作単語ベクトルが必要です。それを作るにはニューラルネットを使います。そのため、学習の順序としては、ニューラルネットを先にやってから、単語ベクトルの課題に取り組んでください。

3つのケース

Pythonをマスターしたい人

- 準備Pythonのところは、一斉授業形式でやります。それで後続のトピックをコードレベルで把握するための素地を作ります。ここだけをさらに掘り下げたいという選択をしても構いません。

機械学習、ニューラルネットを使えるようになりたい人

- 学習ロードマップの図の赤いフォントの部分を丁寧にやってください。進みが早い人は、ネットワークアーキテクチャを軽くかじるところまで。

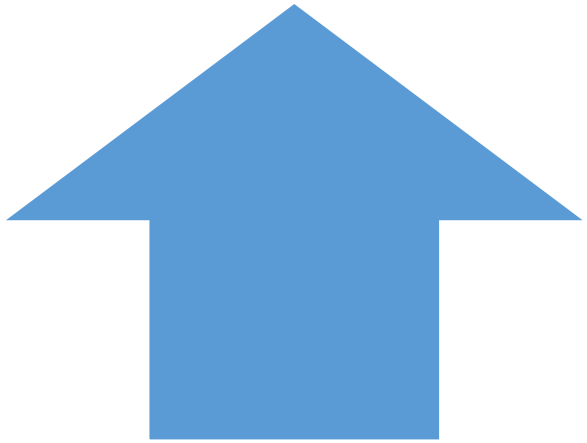
自然言語処理の技術を把握し、AIを使いこなしてアプリで問題解決する力をつけたい人

- 青い部分が自然言語処理に特有の部分です。それらを含めて、順序だてて、学習してください。量が多いので、浅く広くカバーして後で特定トピックを深くでOKです。

モンテッソーリ教育

- **自発的な成長意欲**
- 成長段階に応じた教具を提供

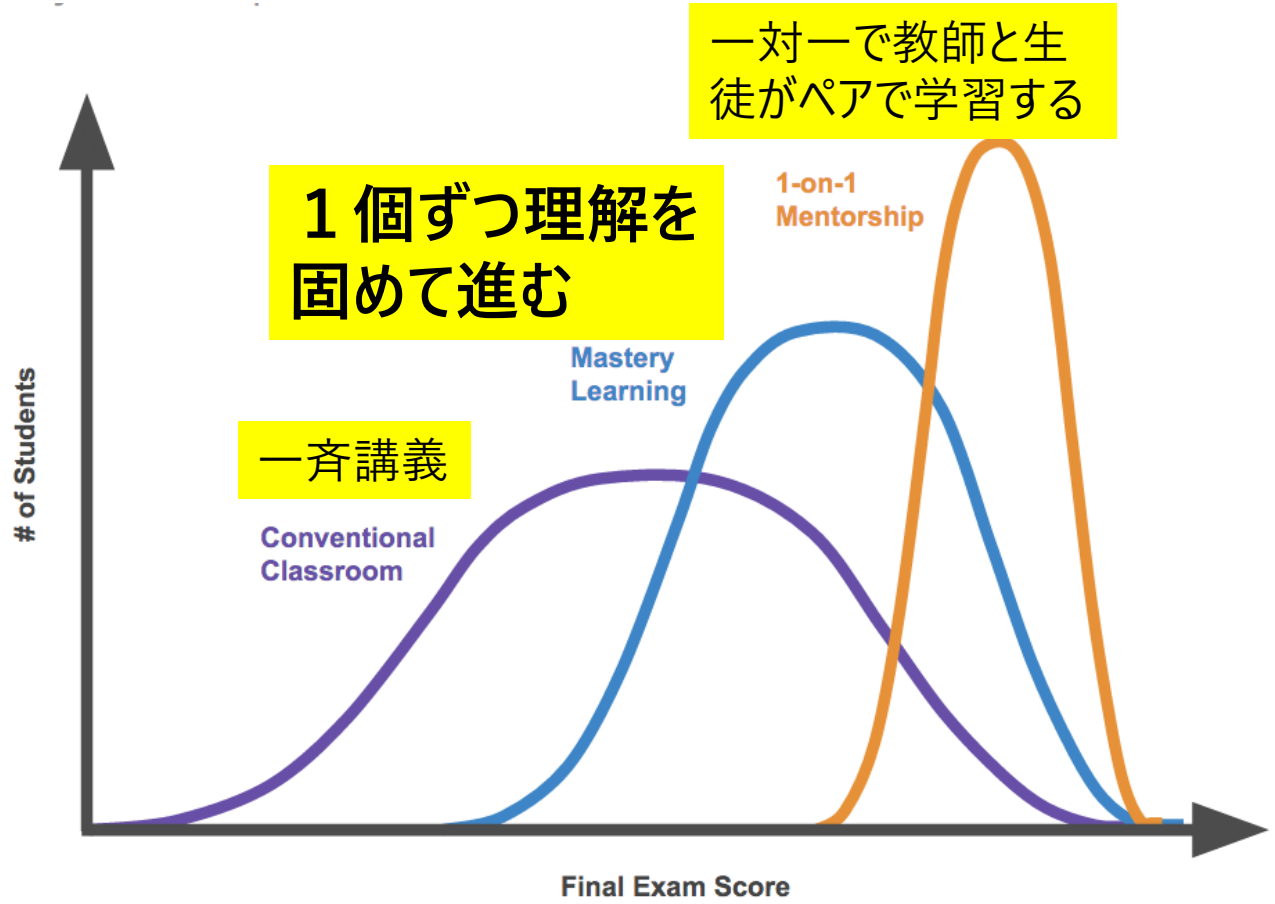
経験上の効果



自分の興味に基づいて、自分で目標を立てて、
自分のペースで学習する、このやり方のほうが、
ほとんどの人が高度なトピックまで消化できます。

経験上、一斉授業だと、高度なトピックまでつ
いてこれるのは、１割程度です。

各人が各人のペースで、1個ずつ理解を進めるやり方が効果ある



一斉授業と選択式学習のミックス

- とはいえ、Pythonに慣れるところに、授業15回分の時間を使うのは、AIの力を踏まえると、もう推奨しません。

- 時代の変化を踏まえると、Pythonを書く部分はAIが相当やってくれることをイメージして、Pythonで何をやらせるかという全体的な処理イメージ・概念を持つことに時間を使った方がいいです。

演習の形式

- 演習は、準備フェーズのPythonのところは、１００本ノックを使ったゼロからの書き起こし。

- 基礎、ネットアーキテクチャのフェーズの演習は、サンプルコードの穴埋め。新しいことに慣れるには、ゼロから書くのは難しく、まずサンプルを読み慣れることから。
  - 穴埋め課題は、AIができるかも、課題の手直しまでまだ手が回っていません。

選択部の学習内容は全部オプション

- この授業は、高度なトピックまでカバーします。
- 一方、皆さんのバックグラウンド、習熟ペース、ゴールは、様々です。

選択部の学習は全部オプション

- 準備のPythonの課題は、授業で一斉に一緒にやります。

- 他方、そのほかのトピックは、各人のバックグラウンド・習熟ペース・ゴールに合わせるため、事前知識の解説のどれを学習するか、課題のどれに取り組むか、**学習内容は、全部、選択性**です。
- **ご自分のペース**で、選んだ解説を視聴・読んだり、選んだ課題に取り組んでください。まずはわからないことがあっても気にせず手早く全体像を把握しては、そのあとで、1個ずつ、特定のトピックに潜って腑に落ちるまでゆっくりじっくりやる、これを繰り返すやり方がいいです。

講師の役割は個別フォロー

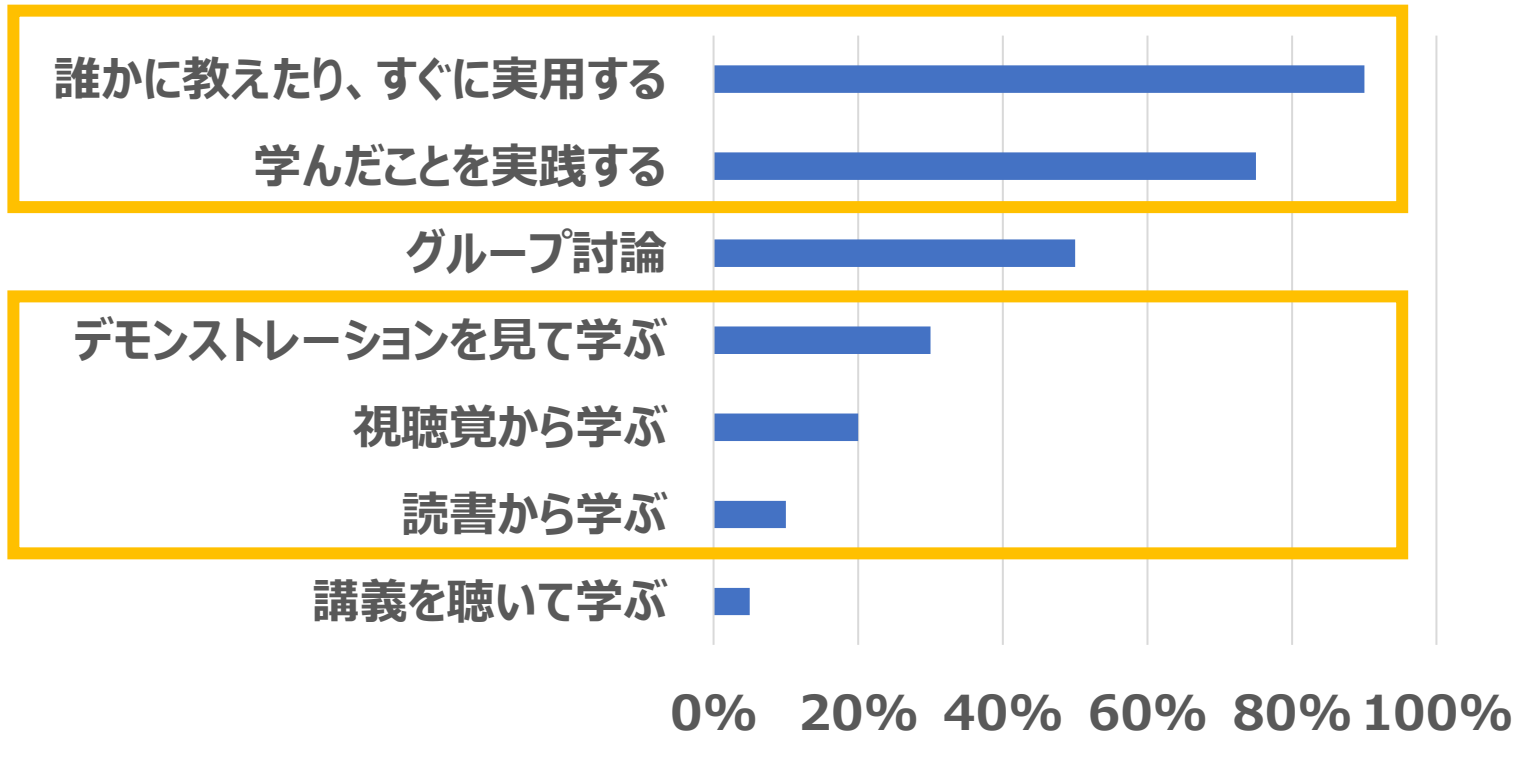
- 行き詰った時の１on１のヘルプ
  - 誰かが引っ掛かりQ&Aを行った場合、ほかの人も同じところでひっかる可能性が高い。そのため、Q&A事項はクラスにメールかお知らせで共有していきます。
- 各人のゴールに沿った学習パスのアドバイス
- 巡回１on１面談
  - 人数が多い場合、これほとんどできないと思います。

- まとめレポートには全部目を通し、たいてい返事を書きます。また、必要なら次の回に１on１面談でフォローします。

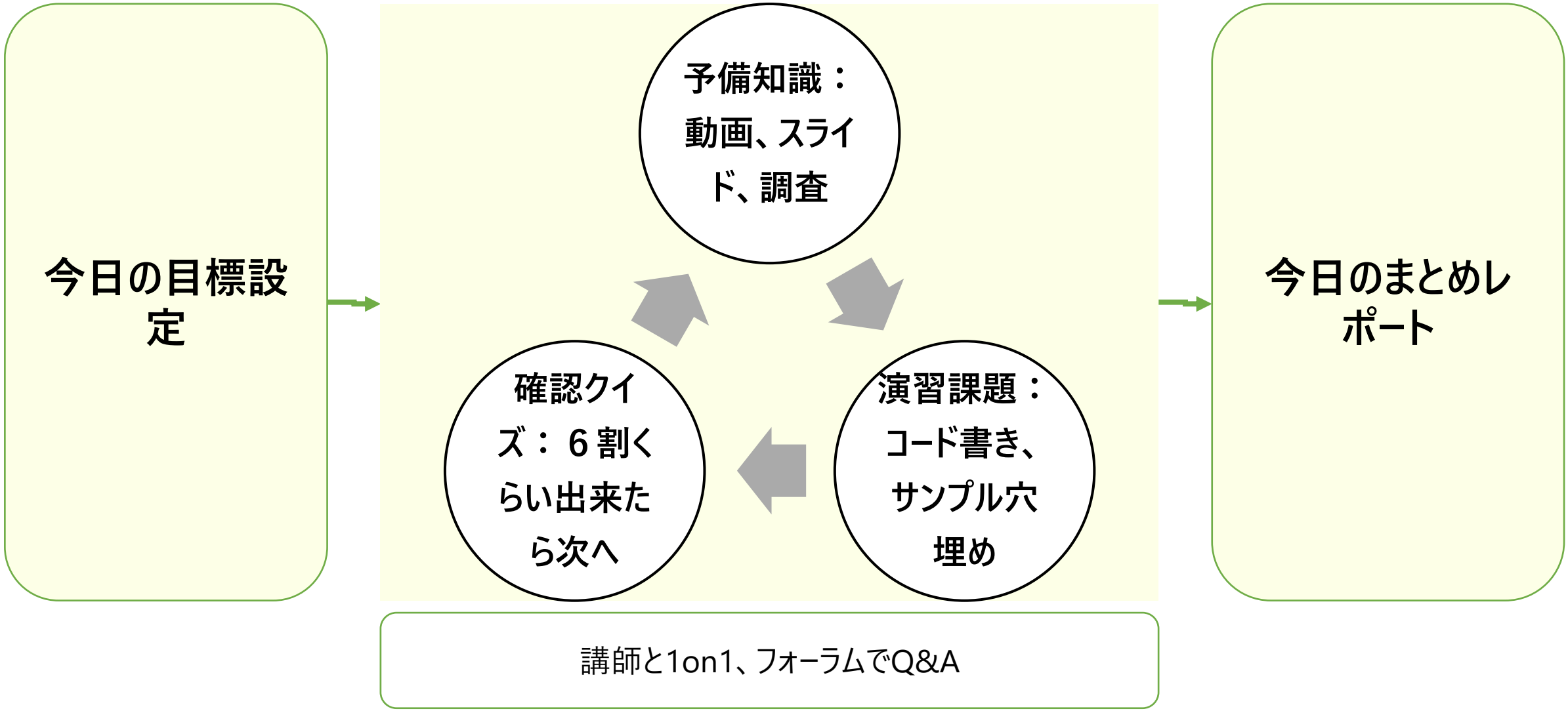
この授業で知識を得たら、自分の課題に取り組める状態になっています。この授業はそこへの道案内。

卒研や社会に出て、自分の課題に取り組むときに、一番力がつく

この授業



授業の時間


今日の目標設定
予備知識：動画、スライド、調査
演習課題：コード書き、サンプル穴埋め
確認クイズ：６割くらい出来たら次へ
今日のまとめレポート
講師と1on1、フォーラムでQ&A

単位認定？

- 取り組むトピック、演習課題、到達目標のどれも選択制です。そのため、単位認定で一律の試験問題はやりません。

- また、提出課題**数**やクイズ**正答率**も、各人の学習スタイルの選択に依存するので、成績評価には使いません。

単位認定？

- 主に、以下の評価の和で判定します。
  - **最後**に、各人の初期値に比べて、どのくらいこの授業を役立ててもらえたかを、文章で**レポート**してもらいます。15点。
  - また、**毎授業の設定目標とまとめレポートの内容**によって、２時間を有効に使ってくれたかどうかを判定します
    - 計画提出1点、まとめ提出2点
    - 初回はまとめのみで3点＋3点 x 13回。
- 出席率を加味します。40点。

学びの手引き

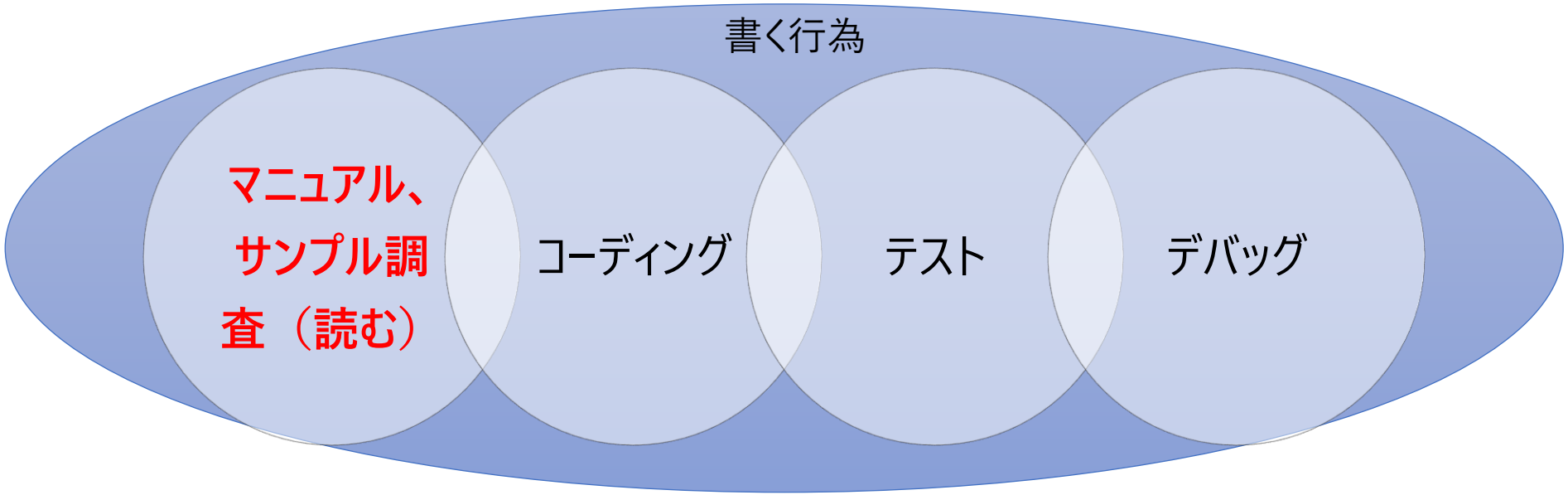
導入ヒヤリング２

Pythonの読み書きに自信がありますか？

この書き方、習ってないから知らないもん？

プログラミング言語の習得

- プログラムを書くとき、たいてい、書き方がわからないケースに遭遇します。新しい知識を習得するには、以下の二つの調査を行い、慣れていくしかありません。
  - **オンラインマニュアルで調べたり、AIに聴く。**
  - **検索して、先輩の書いた解説やサンプルコードを調べる。まず見つかります。**



プログラミング言語の習得

知らないとネガティブにとらえず、知らないことを一つ見つけて（無知の知）、一つ新しいことを学ぶ機会を得たとポジティブにとらえてください。

プログラミング言語の習得

調べたときに、たとえきっちり理解できなくても、使い込んでいけば、赤ちゃんが言葉を覚えるのと同じで、自然と身についてくるので、直ちに悲観する必要は全くありません。

- print文やデバッガのステップ追跡で、動的に結果を確認すると、理解できることがあります。

新しいものに慣れるやり方

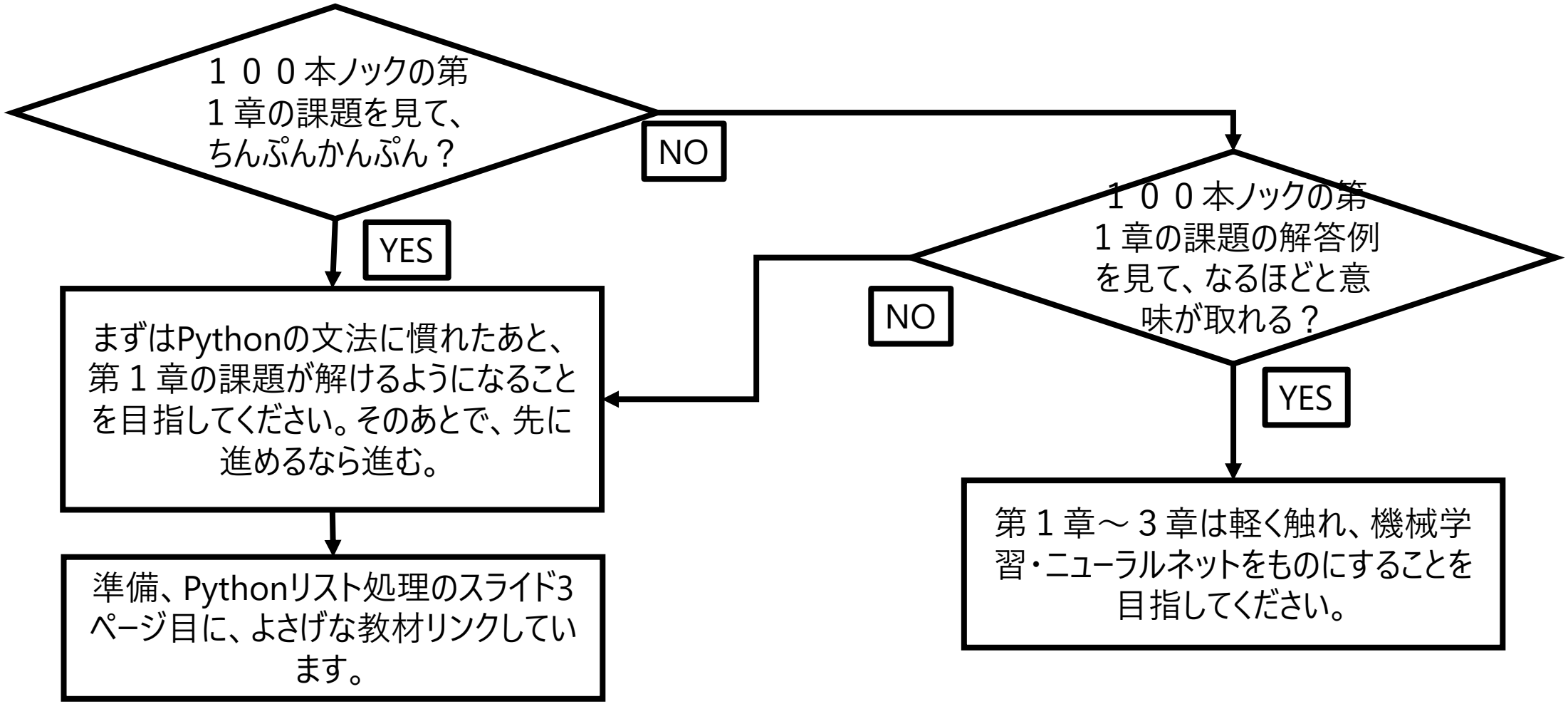
大枠(Python)に慣れていて、その中の特殊なパッケージ（numpy、OpenCVとか）などの場合

- まず使ってみる。必要に応じて、調べて、理解する。そうやって、具体事例の理解を積み上げて、理解を広げていく。

枠組み自体が新しい場合：初めてAndroid プログラミングをやるとか、初めてPyTorchでDLを書くとか

- ある程度、ドキュメントで全体の概要・骨格を把握してから、サンプルを手掛かりにして、個々の問題に取り組む。

Pythonに集中することを目標にするか？



導入ヒヤリング３

数式が苦手？

数学の前提？

ネットワーク構成を理解するためには、各パーツの動作や意味も理解していると、全体の動きがより理解しやすくなります。

- コードの中身にきちんと潜るには、線形代数と微分の知識が必要になります。しかし、この授業では、ネットワークの構成を概略把握することに重点を置き、数学の話は必要最低限だけに絞ります。

数学の前提？

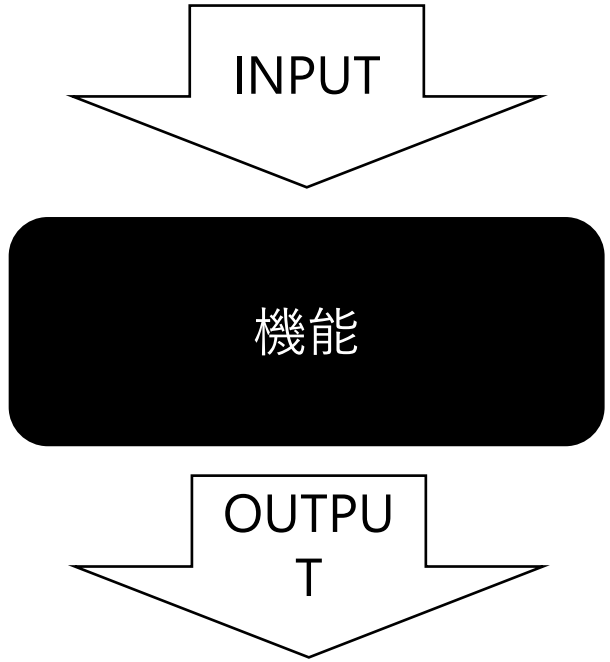
- 意欲のある人は、高校数学の学びなおしのために、無料の https://edupa.org/　の『高校数学標準講義』をお勧めします。その対数・指数、ベクトル・行列、微分のところ。

- また、大学の教養課程の数学は、YouTubeなどで探せば講義の記録が見つかります。偏微分の考え方だけ。

数式が苦手？

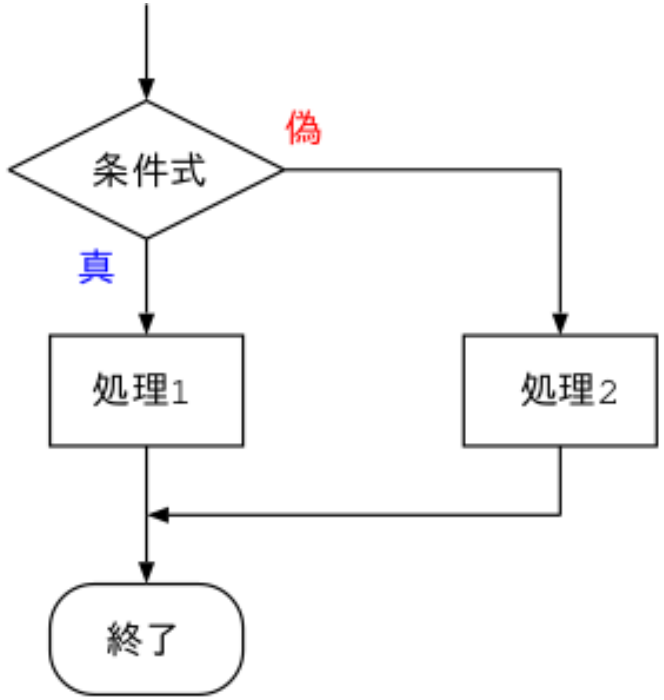
数式は、メソッド。まず機能（何をするためなのか）、入出力はなにで、入出力はどう対応するか、というメソッドの仕様を把握し、腑に落ちたら、中身を知らなくても、もう使えます。

例：Softmaxの機能は、任意の個数の数値を、合計で１となる数値に変換すること。それによって確率値と解釈できる量にする。



数式が苦手？

- 式の中の関数の組み合わせは、いわばメソッドの実装。中の処理フローを理解するのは、アルゴリズムの理解と同じ。具体的なテストデータを使って、処理ステップをトレースすると、理解できる。



数式が苦手？

- 余裕があれば、いろいろな設計の中でなぜその式なのかを調べてみるとよい。
  - 例：softmaxで、なぜネイピア数あるいはオイラー数 e の冪を使うのかは、学習（逆伝播）のための微分結果が扱いやすいから。

強化学習？

- 今どきの強化学習は、深層学習ニューラルネットを関数近似として実装に利用しています。

- 強化学習に取り組むには、まずは第 8 章あたりまで把握してからにしてください。

環境

言語：Python3
ランタイム：Google Colaboratory

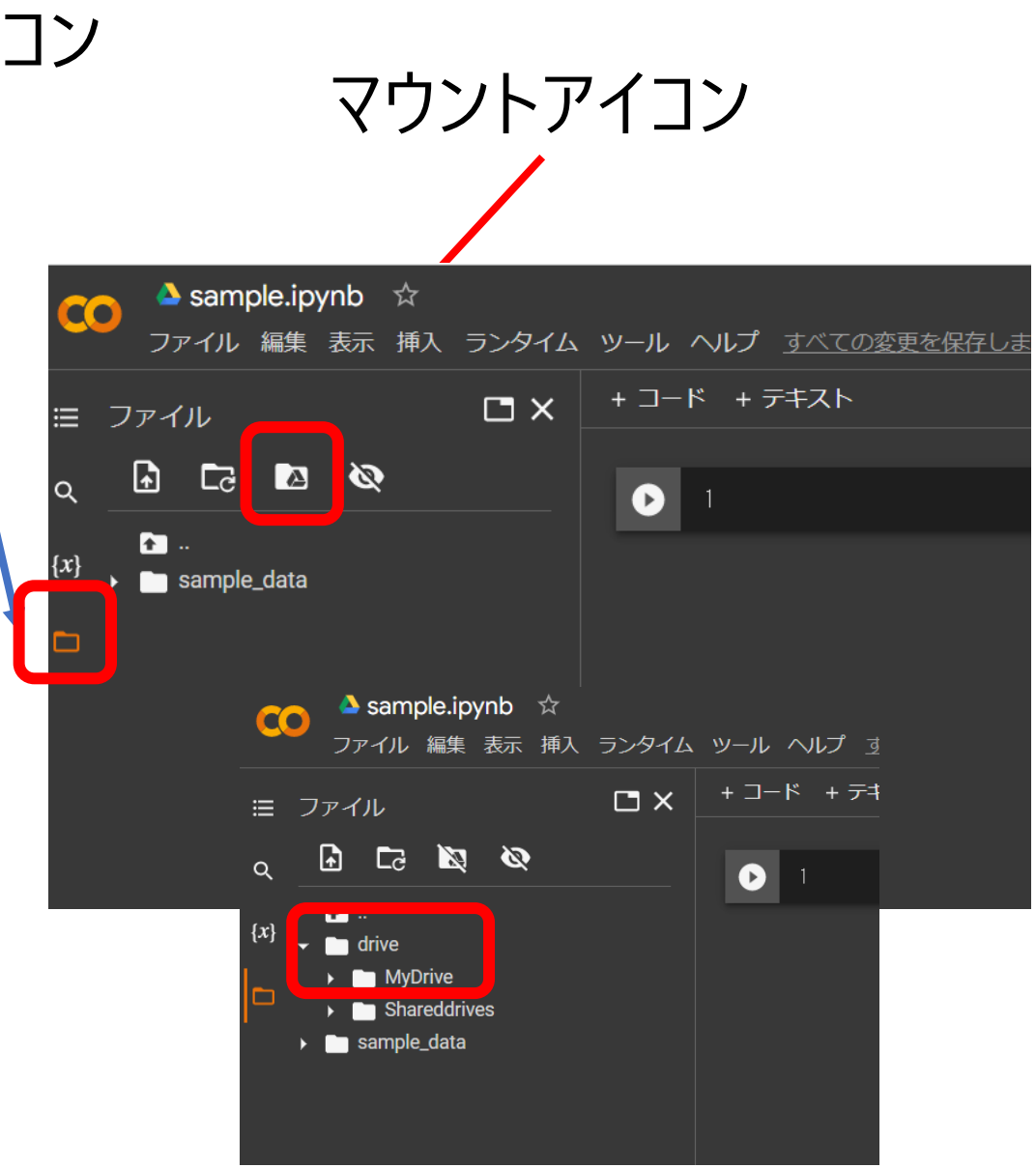
Google Colaboratory

導入：作業１

- ブラウザで　Googleのアカウントに入ります。
- https://colab.research.google.com/notebooks/intro.ipynb?hl=ja　を開いてください。
- 「はじめに」の下、最初のコードブロックの上にカーソルを持っていき、三角アイコンをクリックして、seconds_in_a_day = 24 * 60 * 60　seconds_in_a_day を実行してみましょう。
- ファイルメニューから「新規ノートブックを作成」を選び、左上のファイル名Untitled0.ipynb を sample.ipynbにしてください。
- 各ノートブックには、Python込みのランタイム環境が張り付きます。メモリ上の処理だけなら、このままで動きます。一方、ファイルIOを行う際、Google Driveをマウントする必要があります。カレントディレクトリは、/content/drive/My Drive/Colab Notebooksで、このパスを意識します。
- また、ランタイムはクラウド上の資源で動いているので、ローカルのGdriveと同期するように設定してある場合、作業中に関連ファイルがクラウドと同期ずみかどうかを意識します。

Gdriveのマウント

- 左のファイルアイコンをクリック。
- 左に展開したタブの上部にあるマウントアイコンをクリック。
- 数秒待つと、driveというフォルダーが現れます。自分の drive>MyDrive> Colab Notebooksフォルダー下に 先に作ったsample.jpynbが置かれていることがわかります。
- ーーーーー
- これから授業で提供したノートブックは、ここにローカルにコピーを保存して、課題を書き込み編集してから、スタログへアップロードして提出となります。



導入：作業２

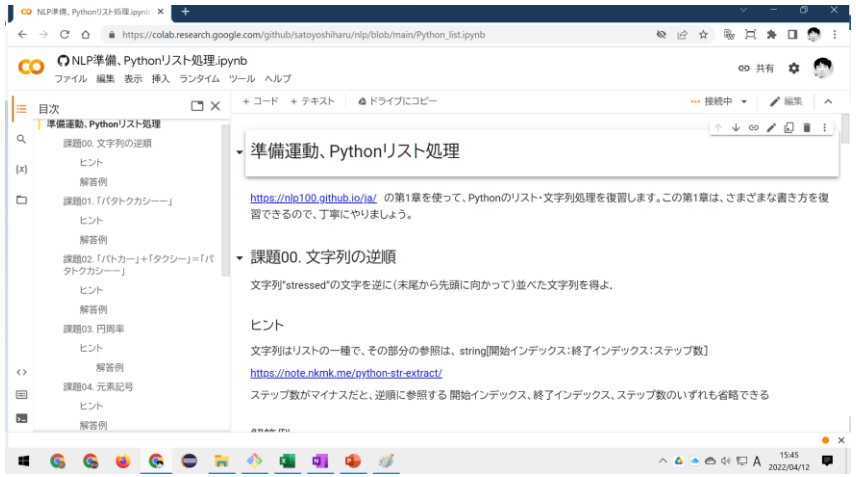
- Google Collaboratoryの操作に慣れます。
  - 目次表示
  - ファイル表示
  - 「+コード」で、コード挿入。コードブロック上にカーソル持って行って右向き三角のクリックで、コード実行。
  - 「＋テキスト」で、テキストブロック挿入。簡単な成形（マークダウン）構文が使える。
  - ランタイムのGPUへの変更

課題提出

- スタログ、「提出、確認クイズ」フォルダーの提出箇所へ提出する
- 100本ノックの課題は、解答（もしくは穴埋めた）コードを入れた、Jupyterノートブック (.ipynb) のファイル提出。
- 確認クイズは、コードの場合と選択式クイズの2種類あります。
  - コードの場合は、ipynbのファイル提出。
  - 選択式クイズは、スタログのクイズとして登録しています。GitHubには.txtでクイズ原稿を載せていますが、それでなくスタログのほうへ直接解答してください。

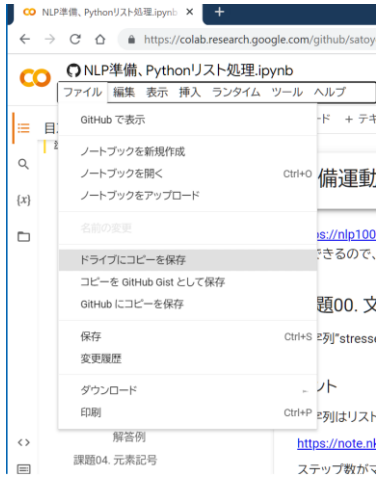
ノートブックの提出

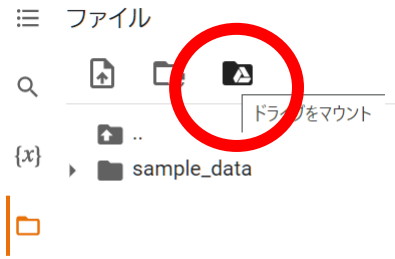
1. https://satoyoshiharu.github.io/nlp/ へアクセスする。
2. 自分の取り組むトピックに応じ、「Pythonリスト処理課題ノート」などをクリックします。
3. すると、Google Collaboratory経由でノートブックが開きます。
   - Google CollaboratoryとGitHubは統合していて、URLに細工することで、CollaboratoryにGitHubのファイルのパスを引き渡しています。



ノートブックの提出

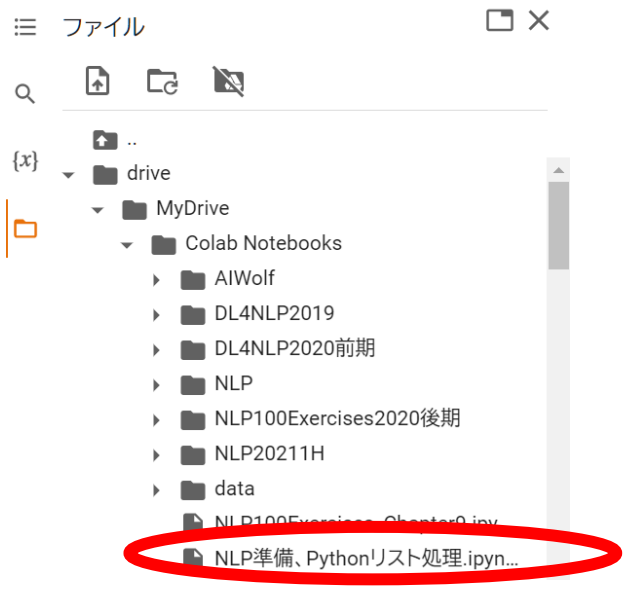
4. ここで、そのままソースを上書きすることを心配せず編集・変更・実行できます。が、提出のため、最初に、ローカルファイルを自分用に作ってしまいます。「ファイル」メニューから「**ドライブにコピーを保存**」を選択してください。
5. 別タブにコピーファイルが開きます。これを自分のドライブに保存します。
6. 左端のファイルアイコンをクリック、開いた左ペインの上のほうにあるドライブの**マウントアイコン**をクリック。
7. このコード実行せよときたら、三角をクリックで実行。Google Driveへのアクセスを許すかというダイアローグが出たら、許す。

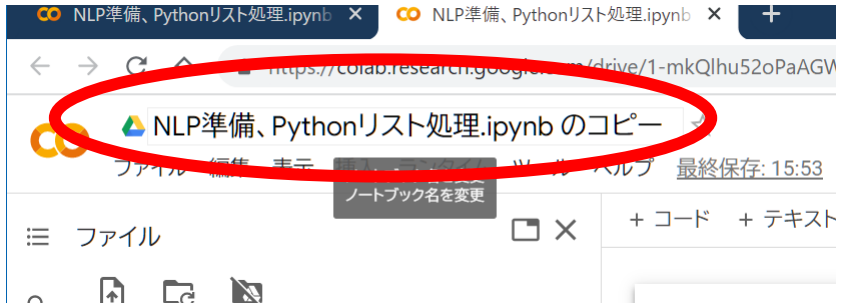




ノートブックの提出

8. １，２秒すると、左のフォルダー改装に、「drive」が登場します。それを展開して、**drive/MyDrive/Colab Notebooks/**　の下に、先ほど作ったローカルなコピーを見つけます。「Pythonリスト処理課題ノート」でしたら、「NLP準備、Pythonリスト処理.ipynbのコピー」とかいうやつです。
9. 左上のファイル名表示のところで、「Pythonリスト処理.ipynb」などと適当な名前に変更します。以降、Pythonリスト処理.ipynb を編集しては、「ファイル」メニューの「保存」。

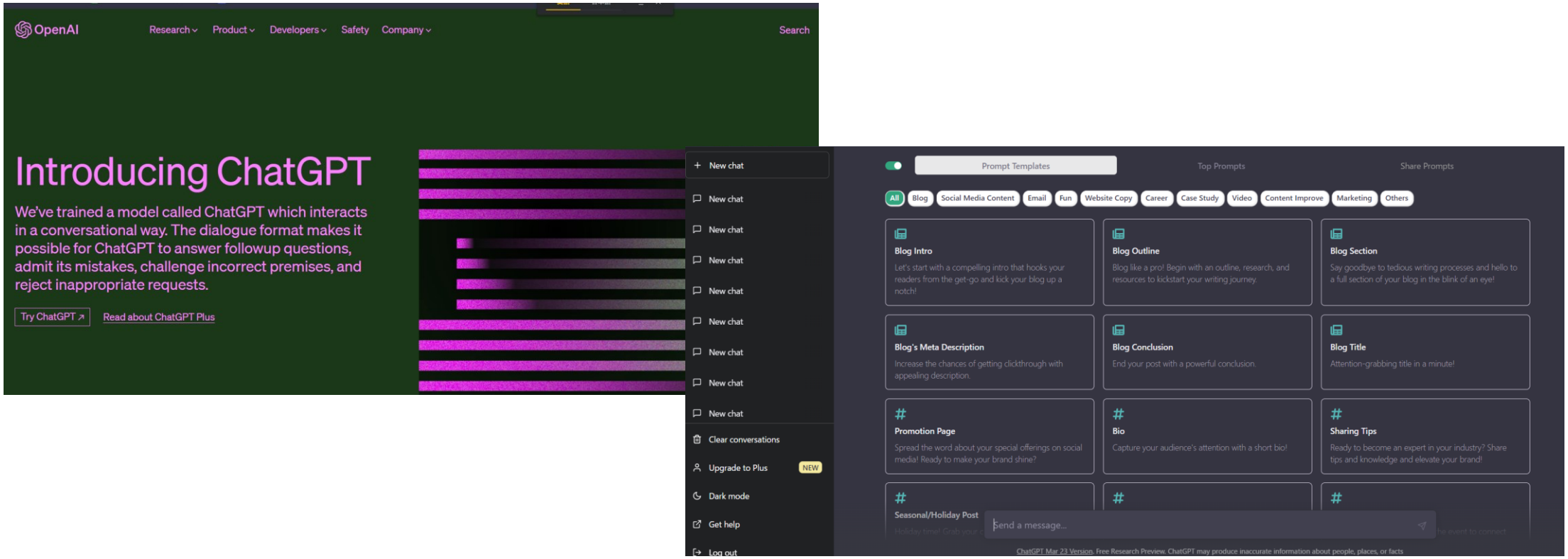




ノートブックの提出

10. ファイル編集が終了したら、
    - もしもGoogle Driveをローカルなディスクと同期設定している場合は、そのローカルなフォルダーのファイルを直接、スタログにアップロードします。その時ファイルのタイムスタンプを見て、クラウドと同期済みかどうか確認してください。
    - そうでない場合、「ファイル」メニューの「ダウンロード」で適応なところにいったん置いてから、それをスタログにアップロードします。

それと、ChatGPTにアカウント登録しておいてください



導入まとめレポート

- 以下を文章にして提出してください。
  - 自分がこの授業でやりたいこと
  - 自分の思ったことや意見
  - 理解できなかったことや残った疑問