

ae-1.人工知能（AI）の基礎と応用： 概要，種類，活用分野，およびプログラ ミング入門

（AI演習）（全15回）

<https://www.kkaneko.jp/ai/ae/index.html>

金子邦彦





金子邦彦 (かねこくにひこ) (福山大学工学部)

【研究領域】

データベース応用、データベース基盤技術、高度データ利用

【実績】

- 学術論文等：27編、査読付き国際会議：76編、その他講演多数
- 教科書等：3
- 授業担当経験：のべ24科目
- 科学研究費：のべ11件 概算のべ数千万円 他大学との共同多数
- 共同研究、受託研究など：のべ10件 概算のべ一億円 国際共同研究あり
- 学部生、大学院生の指導経験多数

詳しくは <http://www.kkaneko.jp/index.html>

人工知能、画像処理、3次元コンピュータグラフィックス (VR含む)、Webシステム、知的システムや社会システムの成功には、データベースが必要 という気持ちで進めています

- **人工知能**（AI, コンピュータに人間のような知的な振る舞いをさせる技術）に関する**基礎知識と応用**について学ぶ.
- 今回は, **AIの基本概念, 種類, 活用分野, プログラミングの基礎**を理解する.
- **AIの基本的な仕組み**を学び, **実践的なスキル**を身につけることを目指す.

人工知能（AI）の学習のための指針



1. **実践重視**： AIツールを実際に使用し、機能に慣れる
2. **エラーを恐れない**： 実行においては、エラーの発生の可能性はある。エラーを恐れず、むしろ学習の一部として捉えるポジティブさが大切。
3. **段階的学習**： 基礎から応用へと段階的に学習を進め、AIの可能性を前向きに捉える

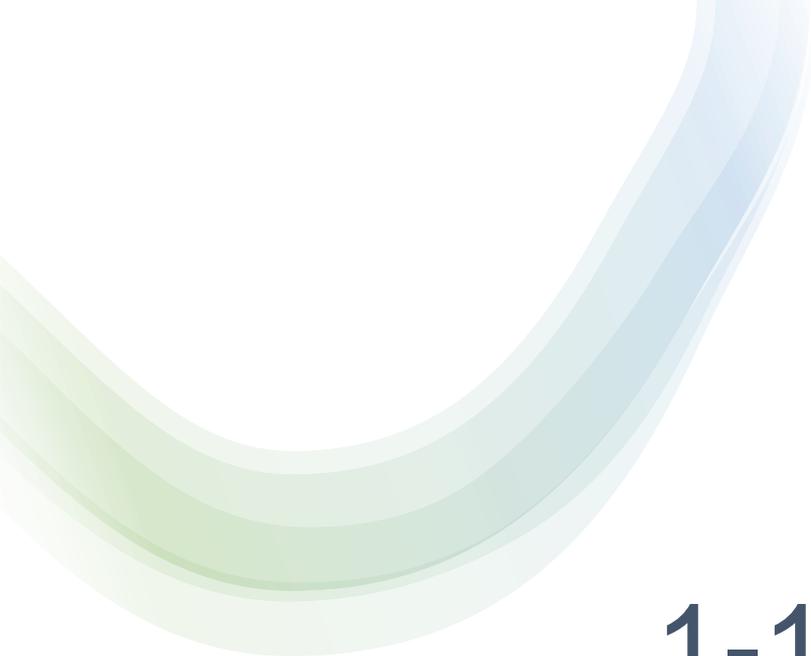
学びへのアドバイス

1. 簡単なことからスタートし、**段階的に複雑な内容へ**
2. **自分自身の成長（さまざまな体験、知識の深まり、スキルの向上、視野の広がり）を感じとる**
3. 大事な**基礎**は**繰り返し**学習する
4. 知識を増やすため、パソコン操作を**辛抱強く**続ける
5. **難しい内容へのチャレンジ精神**を持つ

目標

- 
- ① 多様な技術の理解
 - ② コンピュータとプログラミングの基礎知識
 - ③ AIとプログラミングの深い関連性

これらの要素を総合的に理解し、技術者としての能力を高め、新しい発見やアイデアの創出につなげることを目指す。



1-1. AI入門

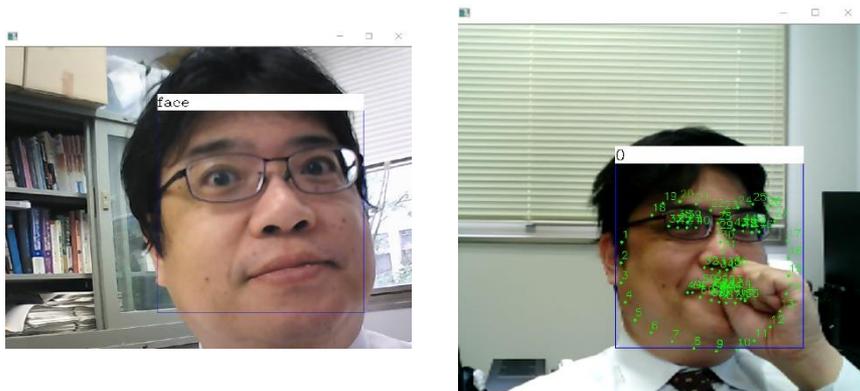
- AI は、コンピュータが人間のような知的能力を持つことを目指す技術

- **AI の 3 要素**

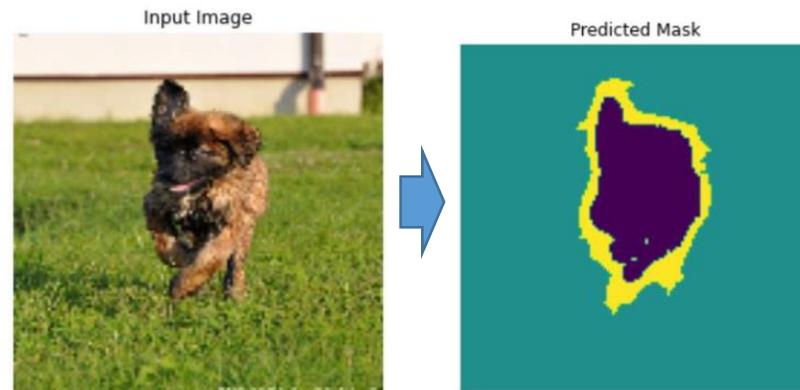
- ① 知能：思考や判断などの能力
- ② 知識：情報を収集・処理する能力
- ③ 学習：知的な能力が上達できる能力

これらの要素が組み合わさることで、高度な処理が可能となる。

人工知能の応用例



顔検知、顔識別



画像のセグメンテーション

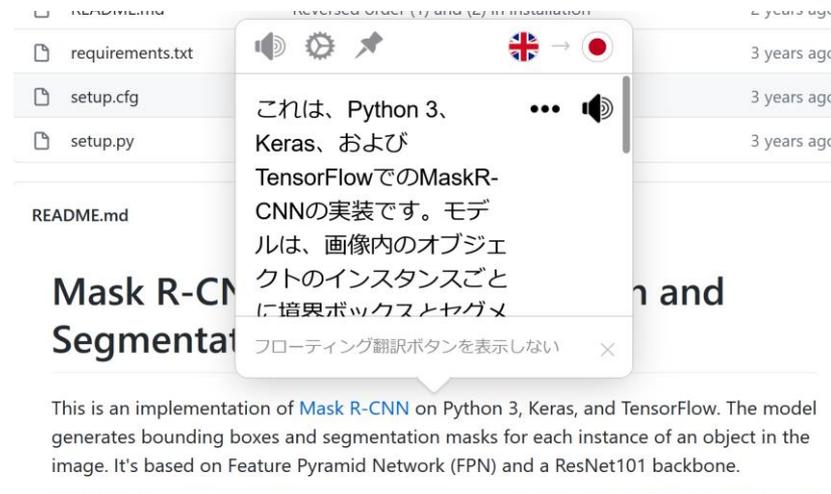


金子の顔

+ 有名人の声、表情、語り

→ 金子がその有名人そっくりで語りだす

合成



Webブラウザで翻訳を行う
Mate Translate (Webブラウザ
Firefox のアドオン)

人工知能（AI）の応用分野の広がり



AIは私たちの生活や仕事を大きく変革する

- **人間の言葉、人間の声の処理**

例：対話型AI（チャットボット）、自動翻訳サービス

- **視覚情報処理**

例：物体検知（自動運転車の障害物検知など）

- **データ分析と予測**

例：顧客行動予測

- **自動化、最適化**

例：工場の最適化、スマートホーム

- **合成**

例：画像生成、画質改善、顔の3次元化

AI の利点



- 24時間365日の稼働
- 大量データの高速処理能力
- 人間が見落とししがちな細かいパターンの検出能力
- 反復作業の効率化

《欠点》

- 創造性や柔軟性の限界
- 予期せぬ状況への対応の難しさ
- 倫理的判断の必要性
- データ品質による結果への影響

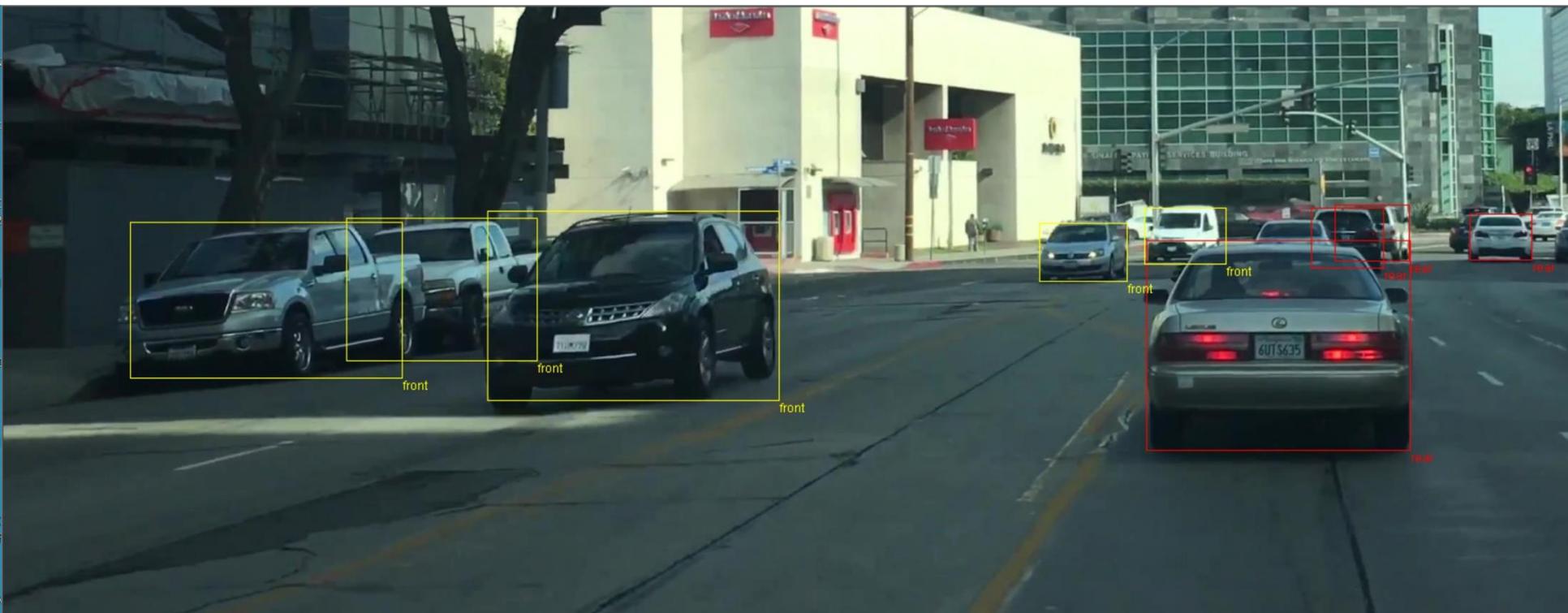


1-2. AIの応用分野と可能性

AIによる物体認識, 人物認識



- 物体認識技術
 - 物体検出
 - セグメンテーション（画素単位の認識）
- 人物認識技術
 - 顔検出
 - 顔のキーポイント（目, 鼻, 口など）の検出
 - 顔識別（顔からの人物特定）
 - 表情の自動判定
 - 目の動きの追跡
- 人体姿勢推定
- これらの技術は, セキュリティシステムやヒューマンインターフェースなどに活用されている



車両の場所と向き（前なのか後ろなのか）の検出

物体認識とセグメンテーション



元画像



人工知能による読み取り結果
(DeepLabv3+ を使用)

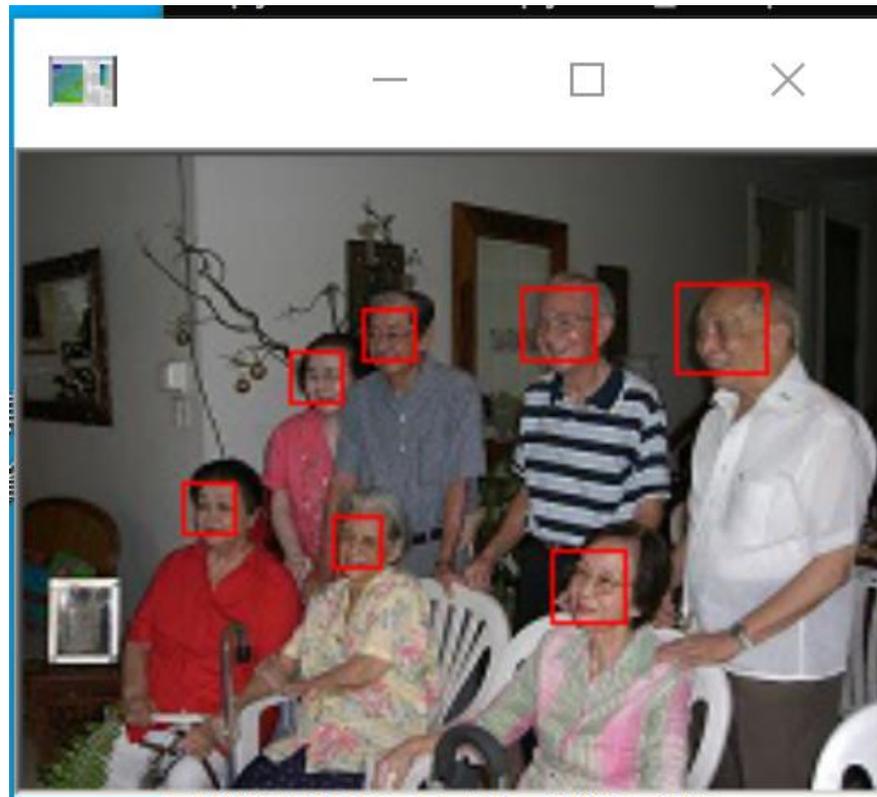
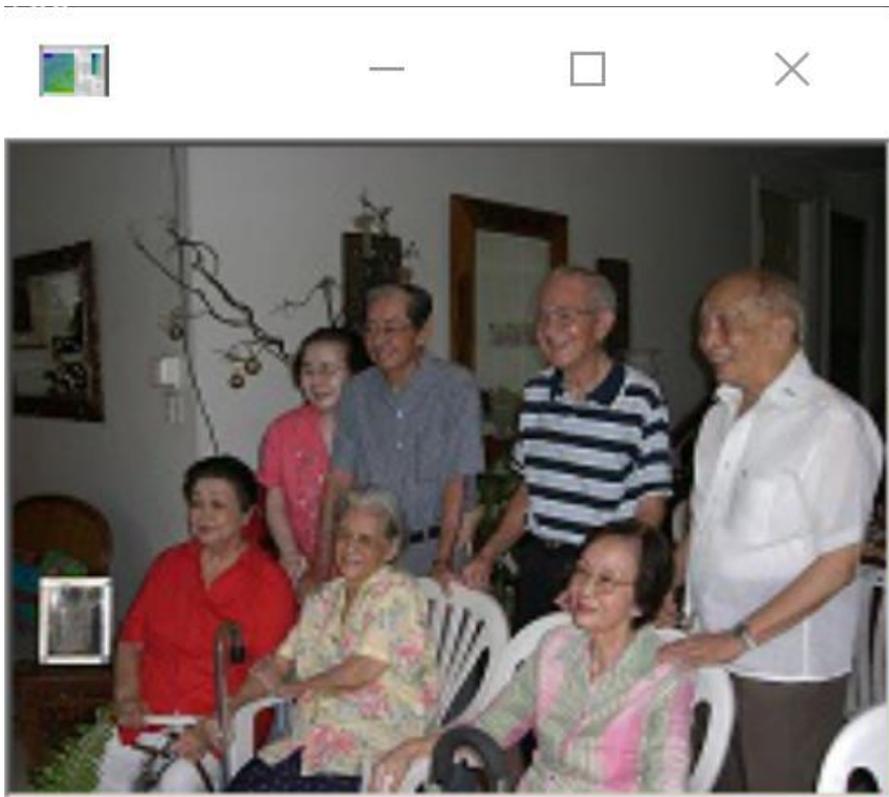
- 人間の「目」の一部機能をコンピュータで再現。
画像の中のオブジェクトを、**人工知能**が発見・検知

自動でのぼかし



人工知能は、手を自動でぼかし、プライバシー保持などに役立てることができるようになってきた
(HypoX64/DeepMosaics を使用)

顔検出



顔検出 (顔の位置, 大きさの情報)

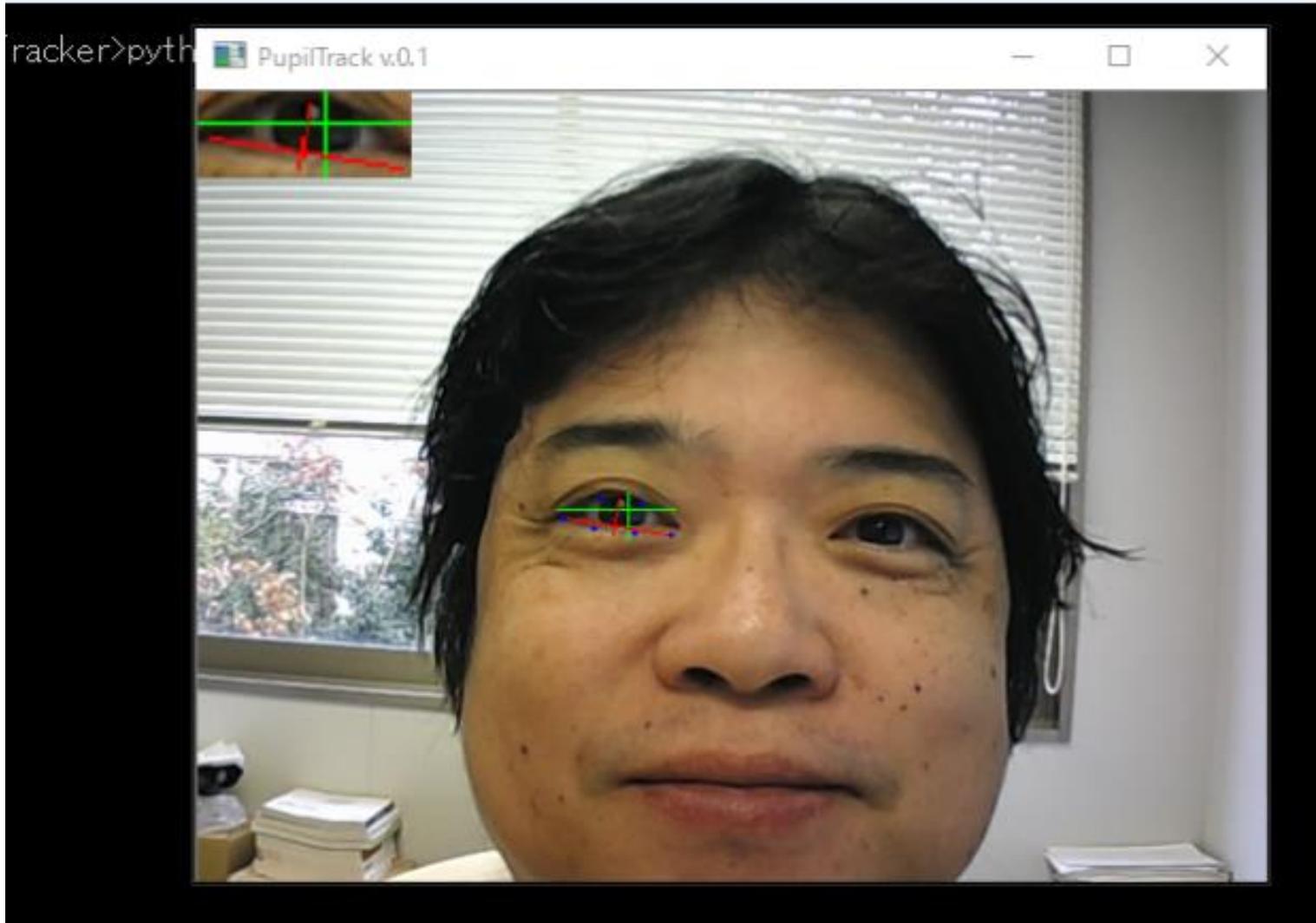


元画像



群衆の数のカウント
(FIDTM を使用)

目の動きの追跡

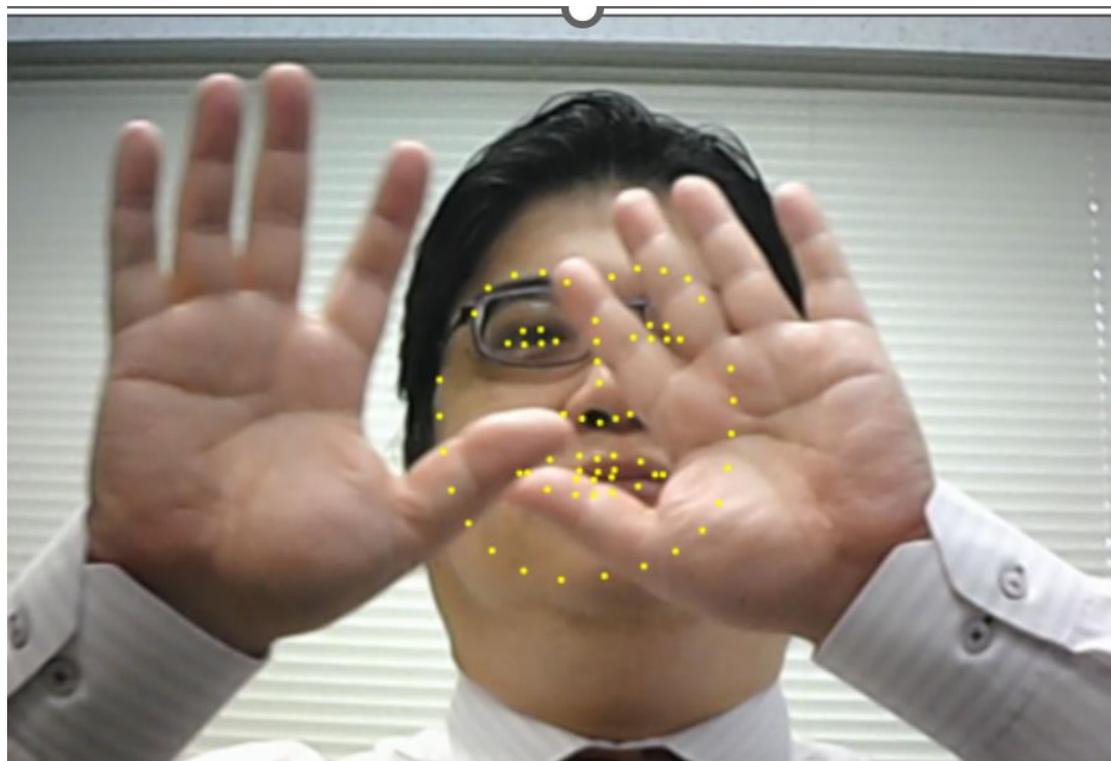


目の動きの読み取り
(Pupil Tracker を使用)

顔のキーポイント

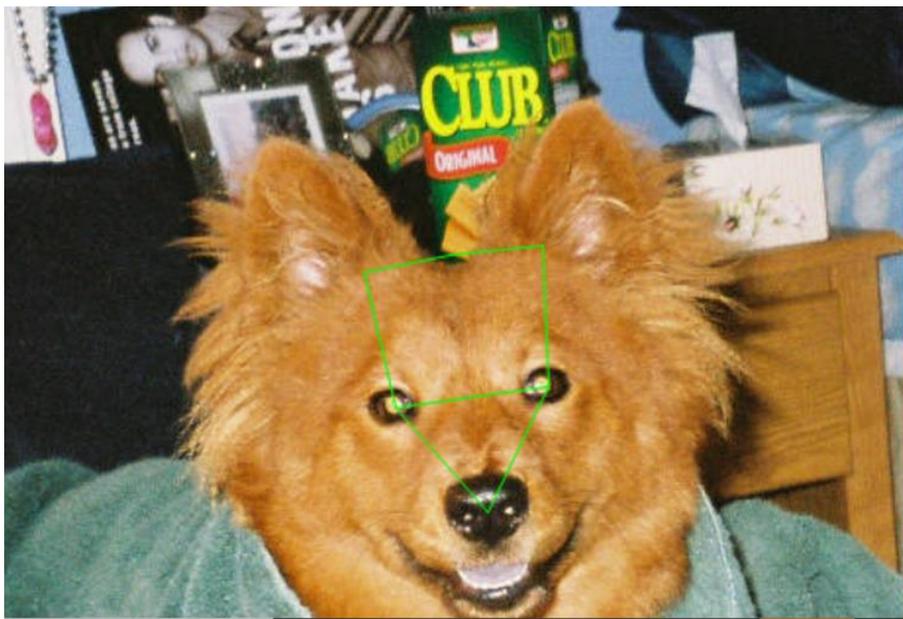


元画像



顔のキーポイント（目、鼻、
口など）の読み取り

顔のキーポイント



キーポイント 5 個
を線で結ぶ



キーポイントを手
掛かりに，眼鏡と
髭をつける

顔識別 (顔からの人物特定)



```
--for danielle, the distance is 0.4635717  
--for younes, the distance is 0.30962762  
--for tian, the distance is 0.48845953  
--for andrew, the distance is 1.0392754  
--for kian, the distance is 0.8913959  
--for dan, the distance is 0.551507  
--for sebastiano, the distance is 0.45932084  
--for bertrand, the distance is 1.0153409  
--for kevin, the distance is 0.80856085  
--for felix, the distance is 0.7121804  
--for benoit, the distance is 0.39749846  
--for arnaud, the distance is 0.7137512  
it's younes, the distance is 0.30962762
```

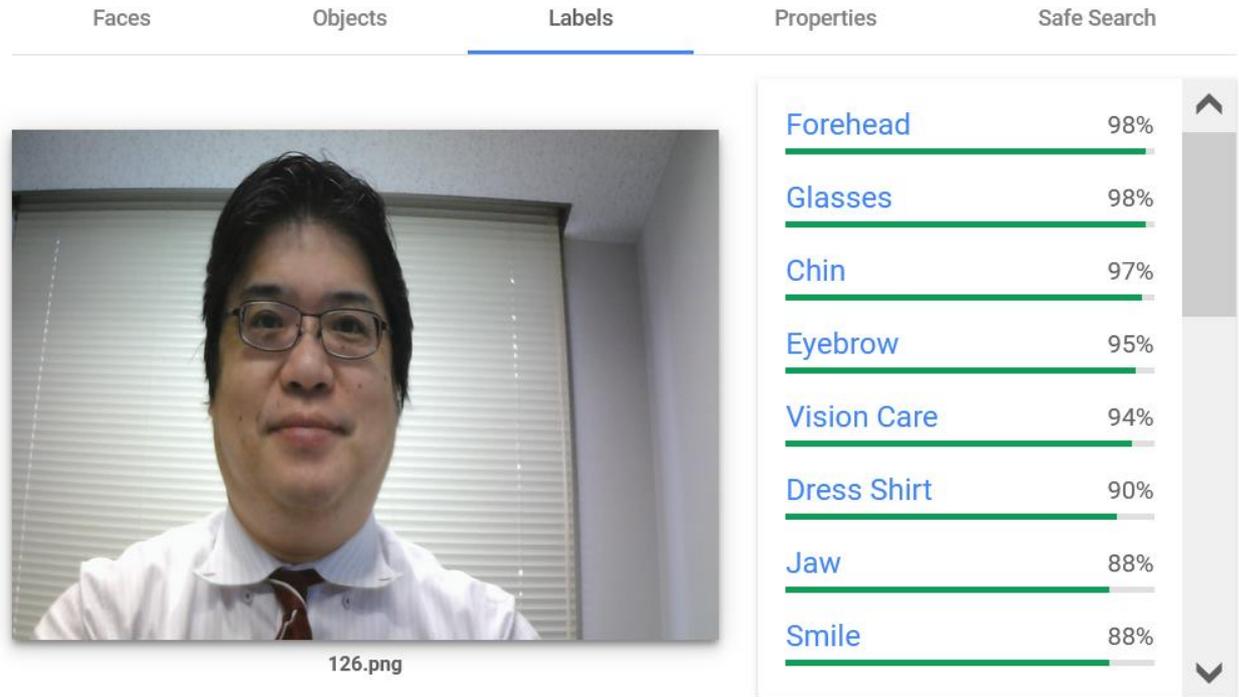
younes

表情の自動判定などを行うオンラインサービス



元画像

Faces Objects **Labels** Properties Safe Search



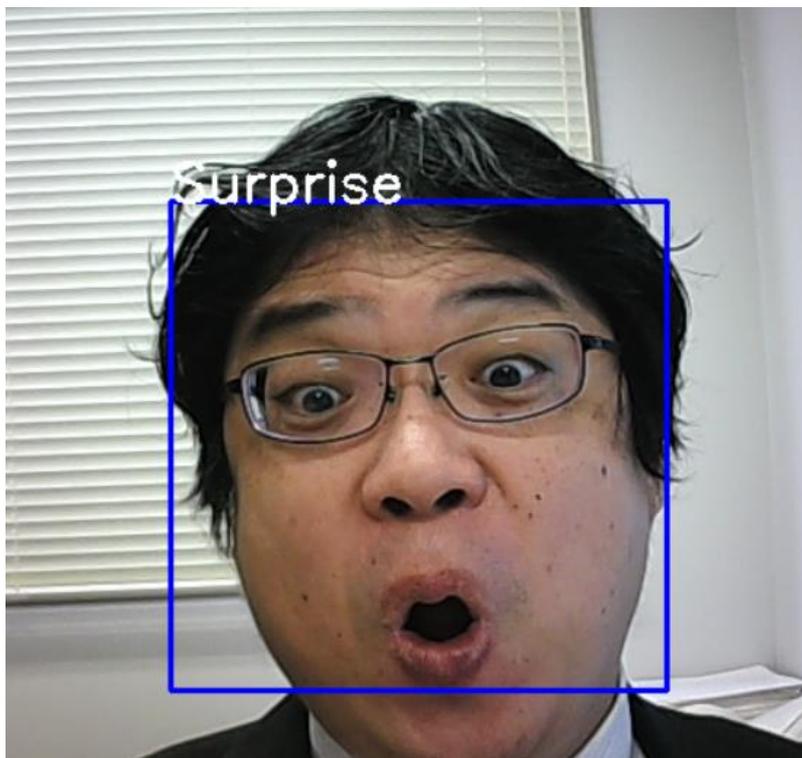
Label	Confidence
Forehead	98%
Glasses	98%
Chin	97%
Eyebrow	95%
Vision Care	94%
Dress Shirt	90%
Jaw	88%
Smile	88%

126.png

画像分類の結果

URL : <https://cloud.google.com/vision/docs/drag-and-drop>

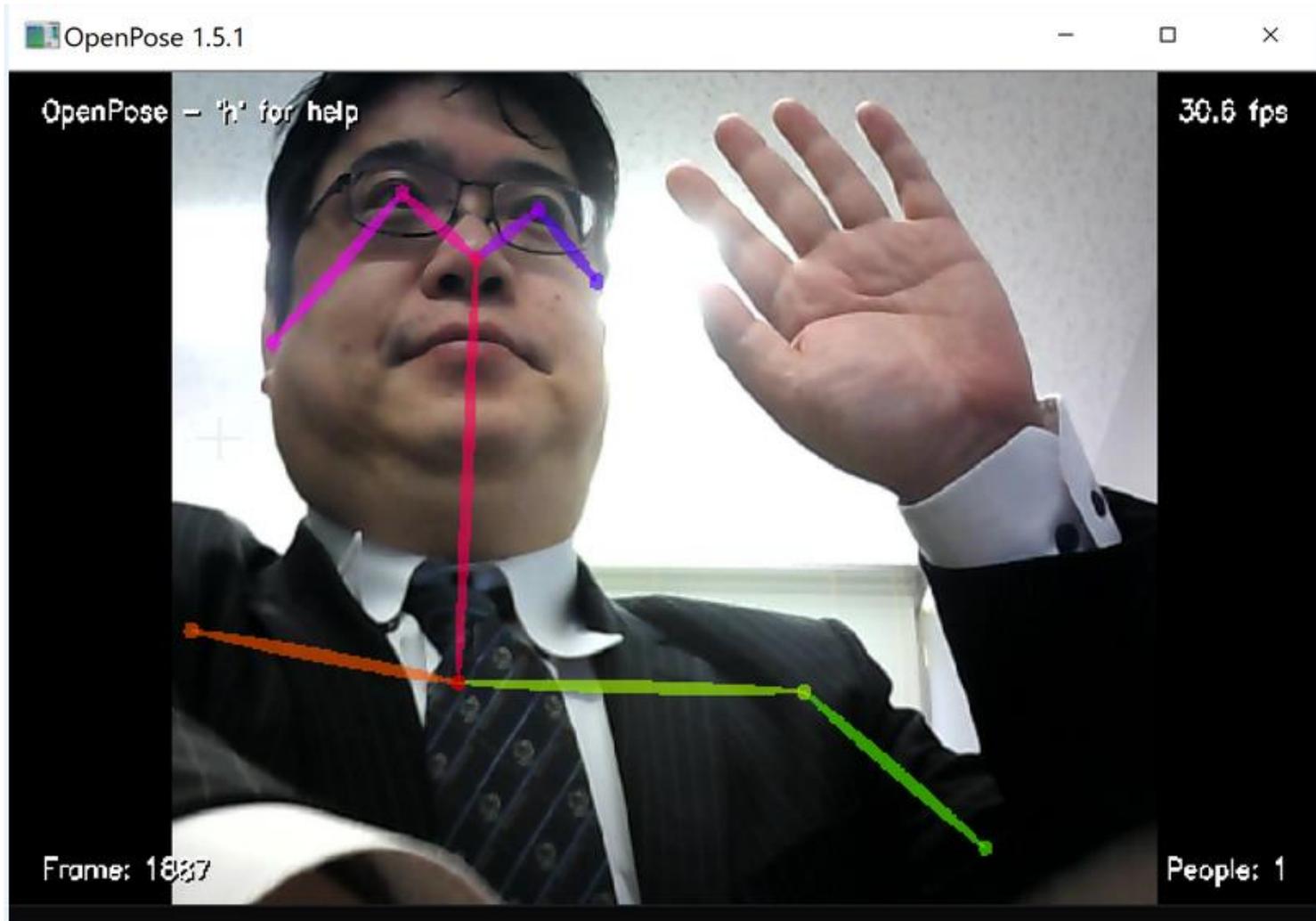
表情の自動判定



```
Angry: % 4.94537390768528  
Disgust: % 7.72874653339386  
Fear: % 2.0912714302539825  
Happy: % 1.1880283243954182  
Neutral: % 30.127882957458496  
Sad: % 1.0293880477547646  
Surprised: % 52.88930535316467
```

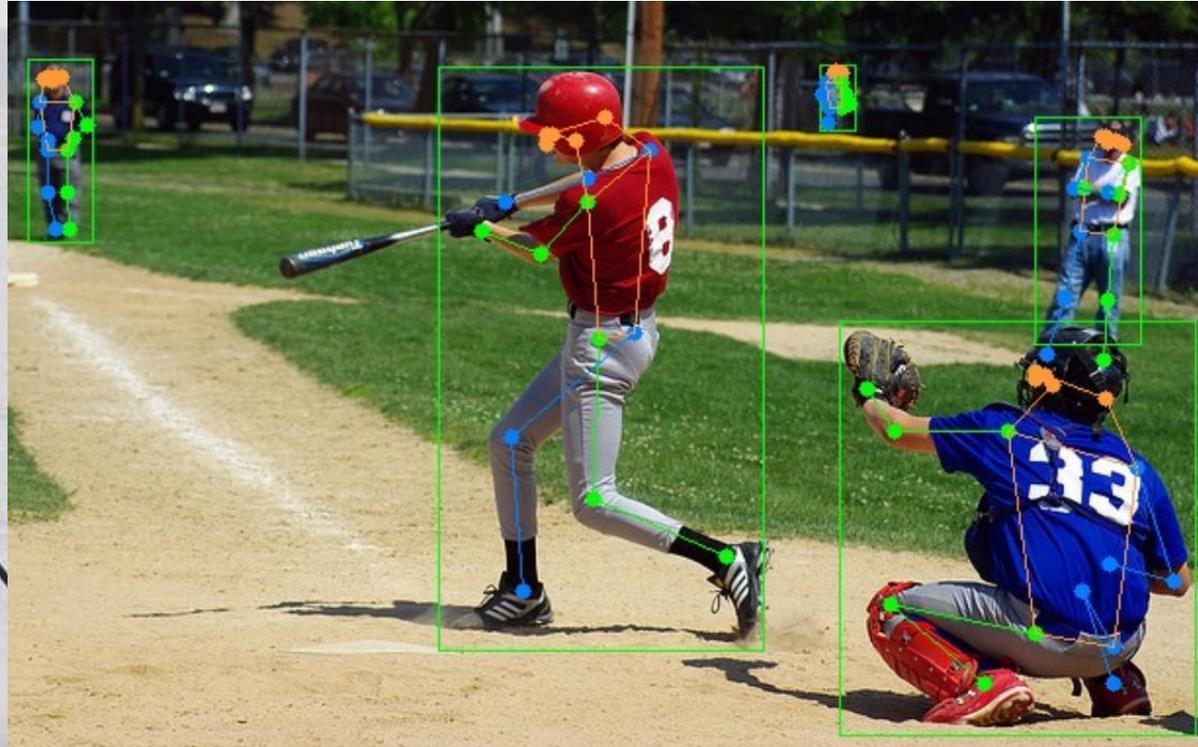
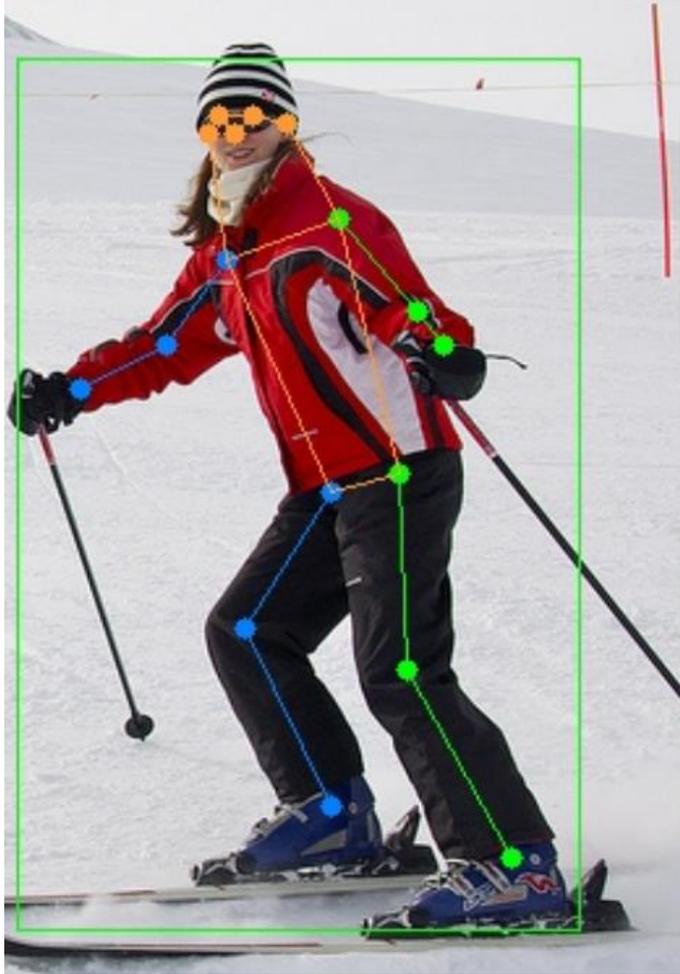
表情の自動判定
「驚き (Surprised)」 と判定
されている

人体姿勢推定



人体の姿勢を読み取り
(OpenPose を使用)

人体姿勢推定



人体の姿勢を読み取り



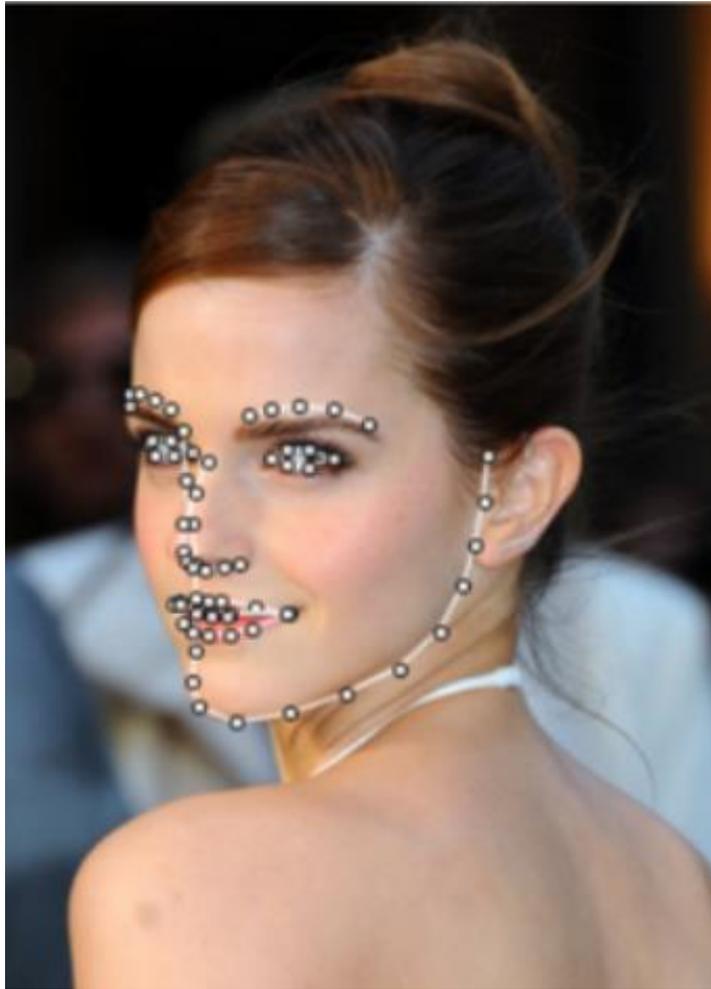


3次元情報処理と合成技術



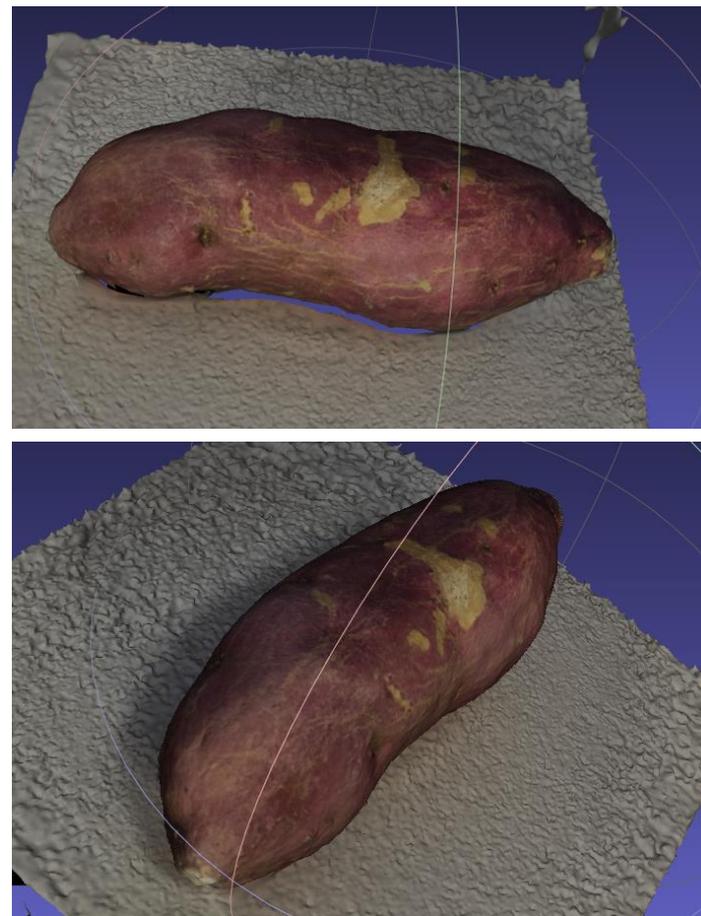
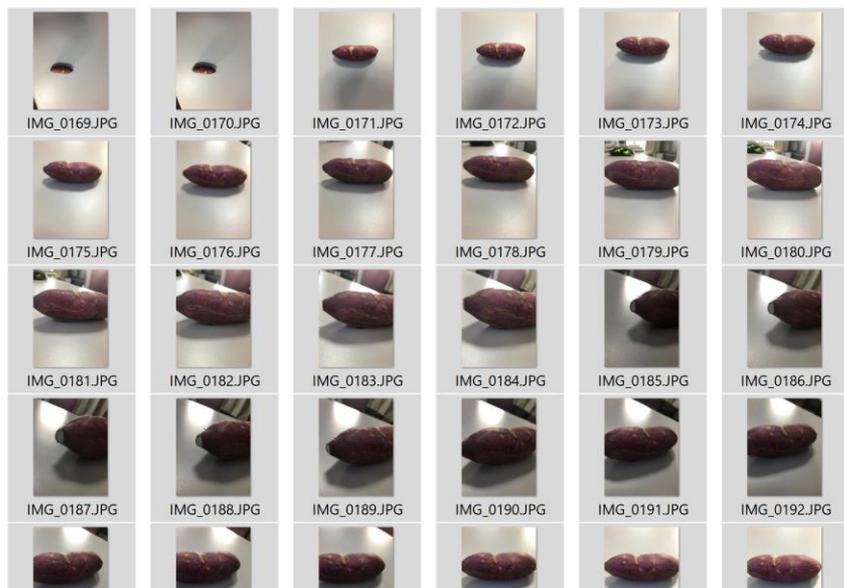
- 3次元情報処理と合成
 - 写真からの顔の3次元化
 - 複数の写真からの3次元データの合成
- 人間の言葉の合成
 - 対話型AI（チャットボット）
 - 翻訳

写真からの顔の3次元化



写真からの顔の3次元化
(3DFFA を使用)

複数の写真からの3次元データの合成



様々な方向から撮影した写真

3次元データをAIで合成

対話型AI（チャットボット）



文章の翻訳、要約のほか、相談や自由なアイデア出しに

KU Java うるう年の計算法 分からない

Javaでうるう年の計算方法は、以下のようになります。

```
if (year % 4 == 0) {
    if (year % 100 == 0) {
        if (year % 400 == 0) {
            // うるう年
        } else {
            // 平年
        }
    } else {
        // うるう年
    }
} else {
    // 平年
}
```

上のJavaのコードは、年が4で割り切れる場合はうるう年の可能性があります、その後100で割り切れる場合は平年となります。それでもうまくいかない場合には、400で割り切れる場合はうるう年となります。

KU 曜日の計算はどうなりますか

曜日の計算には、日付から曜日を求めるためのアルゴリズムがいくつか存在します。中でも一般的に使われるアルゴリズムに、「Zellerの公式」と呼ばれるものがあります。

Zellerの公式は、以下の式で日曜日から土曜日までの7日間の曜日を求めることができます。

```
w = (d + floor(2.6 * m - 0.2) + y + floor(y/4) + floor(c/4) - 2 * c) mod 7
```

ただし、

- w: 曜日 (0:日曜日, 1:月曜日, ..., 6:土曜日)
- d: 日
- m: 月(3月から12月を1月から8月に変換)
- y: 年(世紀から19世紀)
- c: 世紀

例えば、2022年3月1日の曜日を求める場合、以下のように計算します。

```
w = (1 + floor(2.62-0.2) + 22 + floor(22/4) + floor(20/4) - 220) mod 7
w = (1 + 2 + 22 + 5 + 5 - 40) mod 7
w = (35) mod 7
w = 1
```

結果的にw = 1となり月曜日となります。

Javaで実装する場合には、上記の式をそのまま記述することもできますが、Javaには日付や時間に関連するクラスが用意されており、それらを使用することも簡単に曜日を求めることができます。

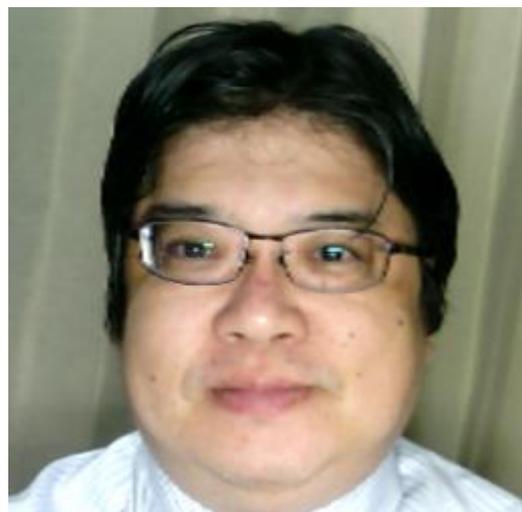
自動翻訳サービス



A screenshot of the DeepL translation interface. At the top, there are two tabs: "テキスト" (Text) and "ドキュメント" (Document). Below the tabs, there are language selection dropdowns for the source language (currently "日本語 - 自動検出") and the target language (currently "英語"). The main area shows a translation of the Japanese sentence "白い雲と青い空が美しい" (Shiroi kumo to aoi sora ga utsukushi) into the English sentence "Beautiful white clouds and blue sky". The interface includes a character count "11/5000", a microphone icon, a speaker icon, and a "フィードバックを送信" (Send feedback) link at the bottom right.

DeepL の URL: <https://www.deepl.com/ja/translator>

現実には存在しないビデオの合成



写真

+



ビデオ

→



フェイクビデオ



- **生産性の向上**：人間がより創造的な仕事に集中できるように
- **科学技術の発展**：膨大なデータから人間には見つけにくいパターンを発見、新薬の開発や疾病の早期発見、農業の発展など
- **コミュニケーションの壁の除去**：言語の壁などを超えたコミュニケーションの支援

AIの社会的影響と責任ある活用

- **人間主導の判断**

AIはあくまで人間を支援する道具。 **最終的な判断は人間。**

- **個人情報の適切な取り扱い**

AIは人間が運用している。 AIが危険というよりも、 AIの運営者が危険な場合がある。

- **偽情報への対策**

AIの悪用により偽情報の生成が極めて容易に。 **複数の信頼できる情報源を参照する習慣が大切。**



ここまでのまとめ



• AIの主要応用分野

言語・音声処理（翻訳、チャットボット）、視覚情報処理（物体識別、顔認識）など

• AIがもたらす可能性

生産性向上、科学技術の飛躍的發展、コミュニケーションバリアの低減

• AIの責任ある活用

個人情報への慎重な取り扱い、偽情報への警戒と情報確認

演習 1. さまざまなAI

ページ 36 ~ 40

【トピックス】

- 翻訳サイト DeepL による翻訳
- AIと将棋対戦
- AIを用いて言葉を絵に

1. 翻訳サイト DeepL による翻訳



① 翻訳サイト DeepL

<https://www.deepl.com/>

② 右側に日本語の文章を
入れると
左側に翻訳結果が出る

2. AIと将棋対戦

① 「ぴよ将棋」のURLをWEBブラウザで開く

<https://www.studiok-i.net/ps/>

② 対局設定し、対戦開始

※ 本格的に楽しみたい場合には「ぴよ将棋」のスマホアプリをお薦めします

対局設定

先手:  後手: 
R0 15級 Lv1 R30 15級

プレイヤー vs コンピューター
 コンピューター vs プレイヤー
 プレイヤー vs プレイヤー
 コンピューター vs コンピューター

先手 レベル: Lv1 ひよこ (R30 15級) ▾
玉囲い: 指定なし ▾

後手 レベル: Lv1 ひよこ (R30 15級) ▾
玉囲い: 指定なし ▾

手合割: 平手 ▾
持ち時間: 指定なし ▾

振り駒 ランダム定跡 レーティング対局

キャンセル 対局開始



さまざまなAIとの対戦ゲーム



- **ぴよ将棋** <https://www.studiok-i.net/ps/>

ブラウザ上で動作する無料の将棋ゲーム。スマホアプリ版もある。様々な難易度のAIと対戦可能で、初心者から上級者まで楽しめる

- **将棋ウォーズ** <https://shogiwars.heroz.jp/>

ブラウザ版とスマホアプリ版があり、AIとの対戦や他のプレイヤーとの対戦が可能。登録必要。初心者から上級者まで楽しめる。

- **Egaroucid for WEB (オセロ)** <https://reversi.simaenaga.net/>

ブラウザで動作するオセロ。Windows版もある。シンプルなインターフェース。登録不要でプレイ可能。

- **lichess.org (チェス)** <https://lichess.org/>

無料でオープンソースのチェスプラットフォームです。AIとの対戦、オンライン対戦、パズルなど様々な機能。

- **囲碁 - Online-Go.com** <https://online-go.com/>

ブラウザベースの囲碁プラットフォームで、AIとの対戦やオンライン対戦が可能。登録必要。初心者向けのチュートリアルもある。

3. AI を用いて言葉を絵に (Craiyon を使用)



① Craiyonのウェブサイトアクセス

<https://www.craiyon.com/>

② ページの中央にあるテキストボックス (プロンプト入力欄) を見つける

③ 英語でプロンプトを入れる

内容を英語の単語や文章で。例: A cat riding a bicycle in space

④ Draw をクリック

しばらく待つ (1 から 3 分以上)

⑤ ③、④を繰り返す

※ プロンプトの作成、
翻訳を ChatGPT に頼むこと
も可能



人工知能に期待される役割



- **人間の仕事の補助や代行。人間との協働。**

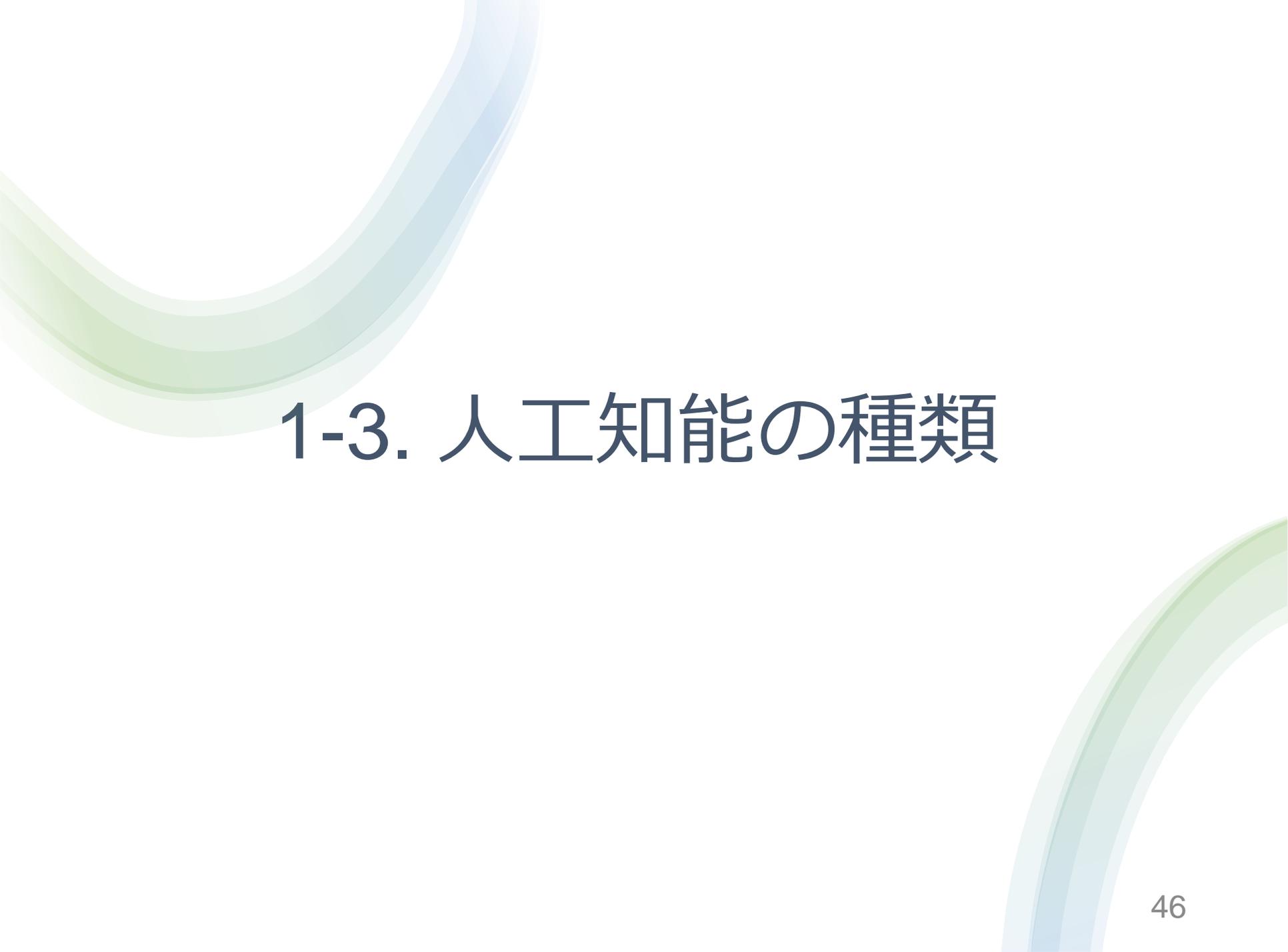
人工知能は、単純なタスクや反復的な作業など、人間にとって退屈な仕事を代行できる。

- **迅速な判断**

膨大な量のデータの処理、高速な計算が得意。人間のミスを減らすことにつながる。

- **新たな創造性や価値の創出の可能性**

人間の知覚を超えたセンサー、情報ネットワークの能力を組み合わせ、新たな分野の開拓や価値の創出が可能に。



1-3. 人工知能の種類

人工知能
(知的なITシステム)

機械学習

コンピュータがデータを使用して**学習**することより**知的能力を向上**させる技術

ルール, 知識による人工知能

人間が**ルール, 知識**をコンピュータに**与える**ことによる人工知能

機械学習の基本



機械学習は、**コンピュータ**が**データ**を使用して**学習**することにより**知的能力を向上**させる技術

- **情報の抽出**：データからパターンや関係性を自動で見つけ出す能力を持つ
- **知的なタスクの実行**：予測，分類などの知的なタスクを実行



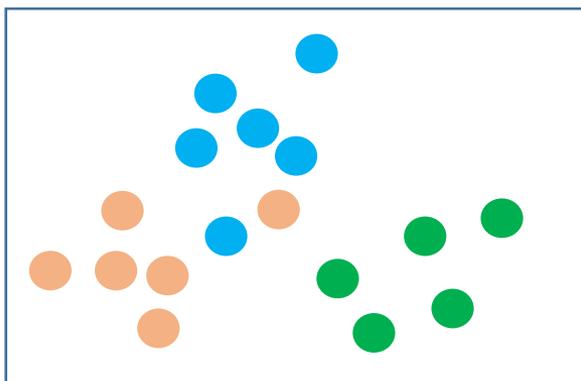
機械学習における訓練データの役割



- 訓練データは、機械学習の学習に使用されるデータである。
- データを用いた学習により、コンピュータは分類や予測などの知的な能力を獲得する。

例：画像分類では分類済みの大量データを使用する。

訓練データ



3種類に分類済み



学習

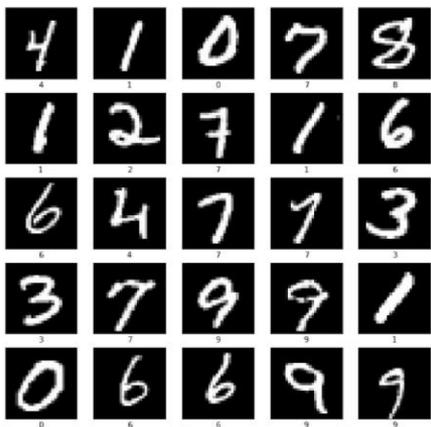


学習者

機械学習の学習プロセス



①データの準備： 目的に応じた訓練データを準備



4 1 0 7 8
1 2 7 1 6
6 4 7 7 3
3 7 9 9 1
0 6 6 9 9

画像 60000枚
(うち一部)

正解 60000個
(うち一部)

②学習の実行： データを用いてパターンを学習

プログラム

```
[4] In[ ]: install -q -uikit-learn matplotlib
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# データの取得と読み込み
iris = datasets.load_iris()
X = iris.data
y = iris.target

# データの標準化
scaler = StandardScaler()
X = scaler.fit_transform(X)

# 訓練データとテストデータの分割
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

X_train = torch.tensor(X_train, dtype=torch.float32)
y_train = torch.tensor(y_train, dtype=torch.long)
X_test = torch.tensor(X_test, dtype=torch.float32)
y_test = torch.tensor(y_test, dtype=torch.long)

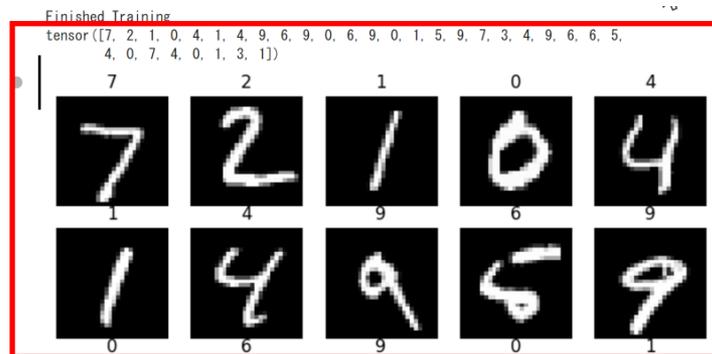
# ニューラルネットワークの定義
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(4, 10) # 入力は4次元 (irisの特徴量)
        self.fc2 = nn.Linear(10, 3) # 出力は3クラス

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

net = Net()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.01)
```

データを用いて学習を行う。学習の結果、文字認識の能力を獲得。

③タスクの実行： 新しいデータを処理



機械学習まとめ



機械学習の特徴

- データを用いて知的能力を向上
- 自動でデータのパターンを抽出
- さまざまなタスクを自動実行

応用事例

画像理解、自然言語処理、予測
など多数

機械学習の定義：

- **訓練データ**を用いて**学習**し、その結果として知的能力が向上
- **訓練データの追加**により、さらに**知的能力が向上**する可能性



1-4. プログラミングの基本 と意義

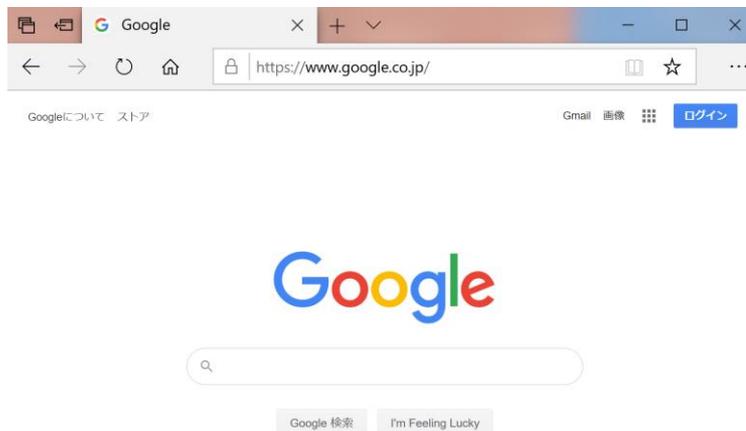
プログラミング



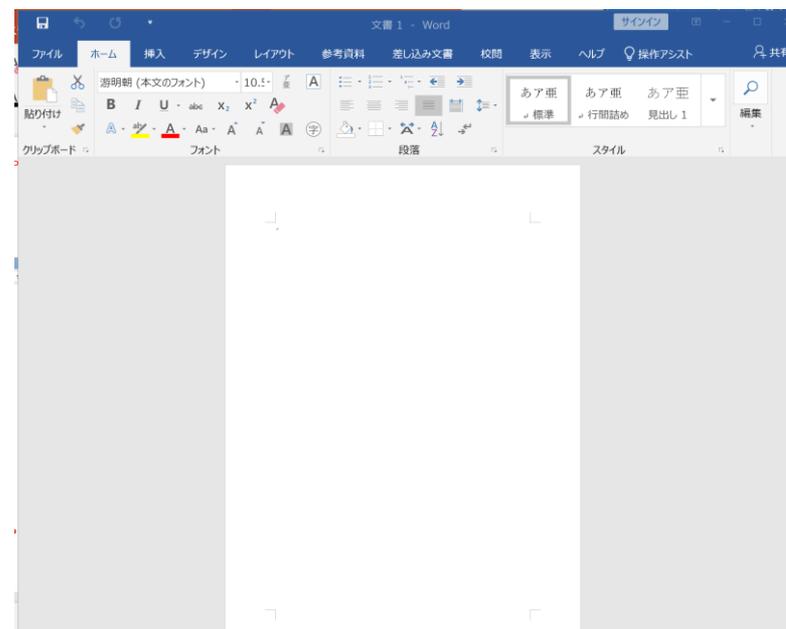
- **プログラム**を設計し作成するプロセス（プログラミング）は、**創造的な活動**
- アイデアを形にできることが、**プログラミング**の魅力



① プログラムとアプリケーション



Web ブラウザ



ワープロ
(マイクロソフト・ワード)

プログラムが動作し、**アプリケーション**の機能を実現

② プログラムは、コンピュータの動作をコントロール



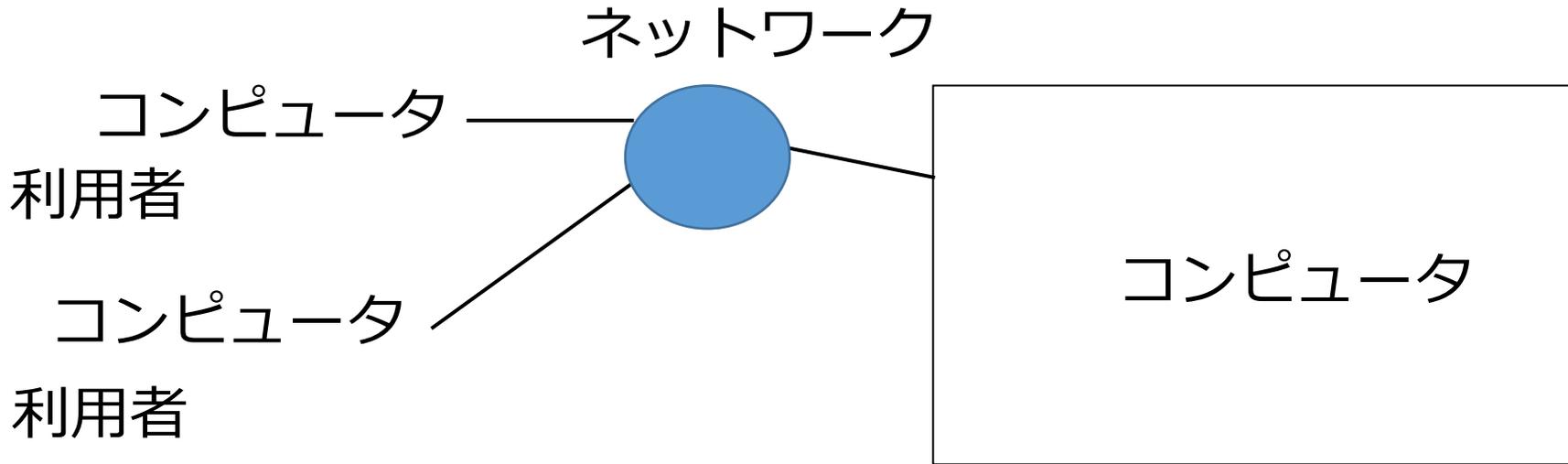
```
In [7]: from keras.models import Sequential
...: model = Sequential()
...: from keras.layers import Dense, Activation
...:
...: model.add(Dense(units=64, input_dim=len(x_train[0])))
...: model.add(Activation('relu'))
...: model.add(Dense(units=max(set(y_train)) - min(set(y_train)) + 1))
...: model.add(Activation('softmax'))
...: model.compile(loss='sparse_categorical_crossentropy',
...:               optimizer='sgd',
...:               metrics=['accuracy'])
...: model.fit(x_train, y_train, epochs=200)
...: score=model.evaluate(x_test, y_test, batch_size=1)
...: print(score)
...: model.predict(x_test)
...: model.summary()

Epoch 1/200
3/3 [=====] - 0s 5ms/step - loss: 1.0583 - accuracy:
0.3200
Epoch 2/200
3/3 [=====] - 0s 0s/step - loss: 1.0530 - accuracy:
0.3200
Epoch 3/200
3/3 [=====] - 0s 0s/step - loss: 1.0485 - accuracy:
0.3200
```

Python 言語を使って
ニューラルネットワーク
を作成. AIシステムを構築

プログラムは、**コンピュータ**の動作を細かくコントロール

③ プログラムは、コンピュータ間の連携にも役立つ



サーバは、サービスを提供するITシステム

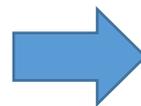
コンピュータ同士の接続でも**プログラム**が必要.

プログラミングの目的



- **プログラム**は、**コンピュータ**に指示を出し、所定の作業を遂行させる
- 複雑な作業も**自動化**し、効率化することが可能

```
a = [200, 400, 300]
for i in a:
    print (i * 1.08)
```



```
216.0
432.0
324.0
```

Python プログラムの
ソースコード

プログラムの
実行結果

プログラミングの利点

- ① **プログラム**の内容によって、**コンピュータ**はさまざまな作業を実行できる
- ② **プログラム**を利用することで、多くの作業を自動化できる
- ③ **プログラム**で行った作業をいつでも再現できる。
- ④ **プログラム**は柔軟性がある。変更により、プログラムの動作を簡単に調整できる



プログラミングの基本と意義まとめ



- コンピュータは、**プログラムの指示に従って動作する**ものである。
- プログラミングは、**創造的な活動としてプログラムを設計・作成**することである。
- プログラムは、**コンピュータの動作を詳細にコントロール**する。
- プログラムを活用することで、**複雑な作業も自動化**が可能である。
- プログラムは、**作業の再現性と柔軟な変更が可能**という利点を持つ。

1-5. Python言語の特徴とプログラミングの可能性

Python

- **Python** は多くの
人々に利用されてい
る**プログラミング言
語**の1つ
- **読みやすさ, 書きや
すさ, 幅広い応用範
囲**が特徴

```
from keras.models import Sequential
: model = Sequential()
.: from keras.layers import Dense, Ac
.:
... model.add(Dense(units=64, input_di
... model.add(Activation('relu'))
... model.add(Dense(units=max(set(y_tr
... model.add(Activation('softmax'))
... model.compile(loss='sparse_categor
... optimizer='sgd',
... metrics=['accuracy'])
... model.fit(x_train, y_train, epochs
... score=model.evaluate(x_test, y_tes
... print(score)
... model.predict(x_test)
... model.summary()
epoch 1/200
3 [=====] - 0s
3200
epoch 2/200
3 [=====] - 0s
3200
epoch 3/200
[=====] - 0s
0
```

Python 言語が広く使用されている理由



文法のシンプルさ

- Python は、**直感的で読みやすい文法**
- 例えば、**print** で簡単に**出力**できる、**if** や **else** で**条件分岐**、**for** や **while** で**繰り返し**（ループ）

拡張性

- **多岐にわたる分野で利用が可能**
- 例えば、**関数やクラスを定義**するための **def** や **class**、**継承**や**オブジェクトの属性名と値**を操作するための **super** や **vars** などがある。

柔軟性

- シンプルなスクリプトも、高度なプログラムも作成可能
- **オブジェクト指向**の機能を持ち、**__init__** や **self** のようなキーワードを使用して**クラス**を利用できる。

- Trinket は**オンライン**の Python、HTML 等の**学習サイト**
- ブラウザで動作
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）
- **Python の標準機能**を登載、その他、次の外部ライブラリがインストール済み

matplotlib.pyplot, numpy, processing, pygal

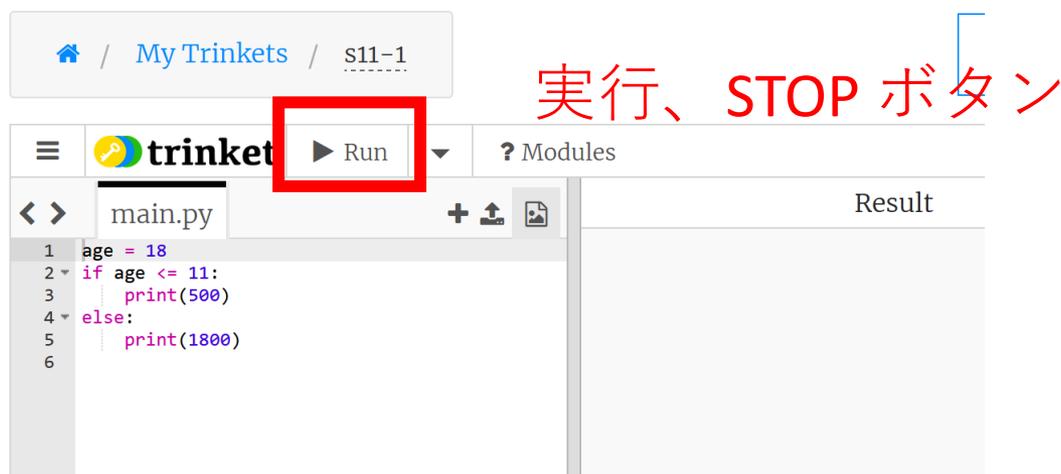


trinket でのプログラム実行

- trinket は Python, HTML などのプログラムを書き実行できるサイト

- <https://trinket.io/python/0fd59392c8>

のように、違うプログラムには違う URL が割り当てられる



ソースコードの
メイン画面

実行結果

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを書き替えて再度実行することも可能

ソースコード

- ソースコードは、プログラミング言語で書かれたプログラムのもの
- 人間も読み書き，編集できる
- ソースコードにより，プログラムの動作を理解し，必要に応じて改変できる



```
function() {
  //is the element hidden?
  if (!t.is(':visible')) {
    //it became hidden
    t.appeared = false;
    return;
  }

  //is the element inside the visible window?
  var a = w.scrollLeft();
  var b = w.scrollTop();
  var o = t.offset();
  var x = o.left;
  var y = o.top;

  var ax = settings.accX;
  var ay = settings.accY;
  var th = t.height();
  var wh = w.height();
  var tw = t.width();
  var ww = w.width();

  if (y + th + ay >= b &&
      y <= b + wh + ay &&
      x + tw + ax >= a &&
      x <= a + ww + ax) {

    //trigger the custom event
    if (!t.appeared) t.trigger('appear', settings.data);

  } else {

    //it scrolled out of view
    t.appeared = false;
  }
};

//create a modified fn with some additional logic
var modifiedFn = function() {

  //mark the element as visible
  t.appeared = true;

  //is this supposed to happen only once?
  if (settings.one) {

    //remove the check
    w.unbind('scroll', check);
    var i = $.inArray(check, $.fn.appear.checks);
    if (i >= 0) $.fn.appear.checks.splice(i, 1);

  }

  //trigger the original fn
  fn.apply(this, arguments);
};
```

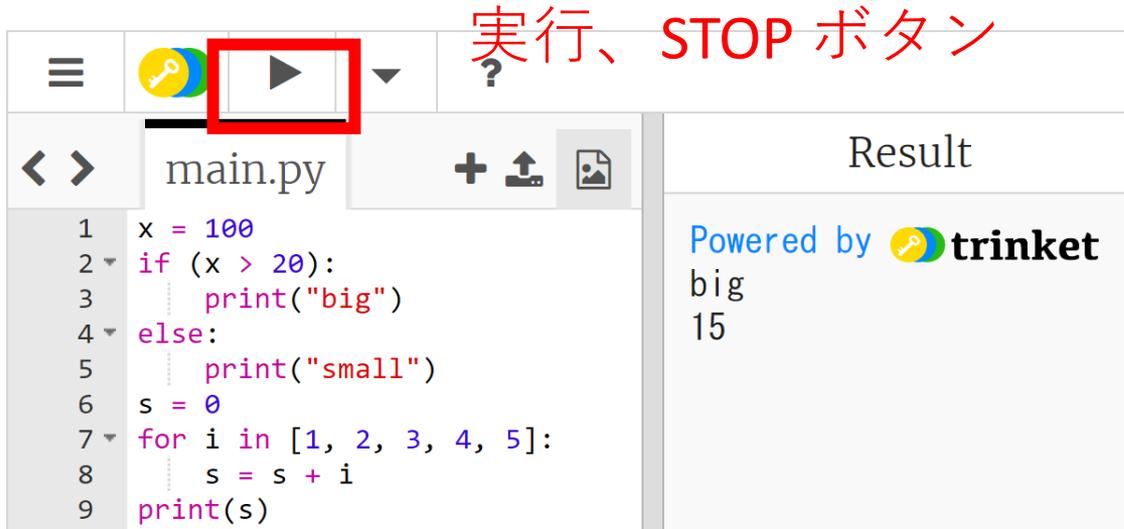
演習. Python プログラムの実行

① trinket の次のページを開く

<https://trinket.io/python/6c652f1c2f>

② 実行結果が、次のように表示されることを確認

実行、STOP ボタン



```
1 x = 100
2 if (x > 20):
3     print("big")
4 else:
5     print("small")
6 s = 0
7 for i in [1, 2, 3, 4, 5]:
8     s = s + i
9 print(s)
```

Result

Powered by  trinket

big

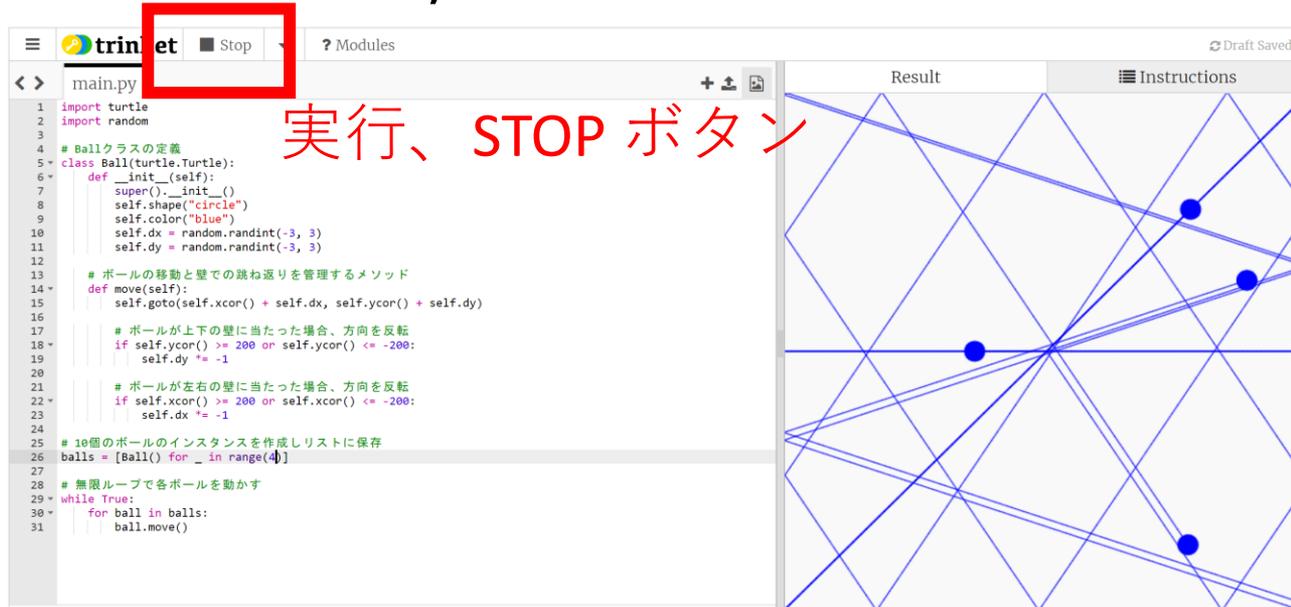
15

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

③ trinket の次のページを開く

<https://trinket.io/python/94d1563844>

④ 実行結果が，次のように表示されることを確認



実行、STOP ボタン

ボールが
壁に当たったら
反射する。

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを書き替えて再度実行することも可能

演習.
簡単なプログラムでも
さまざまなことが可能

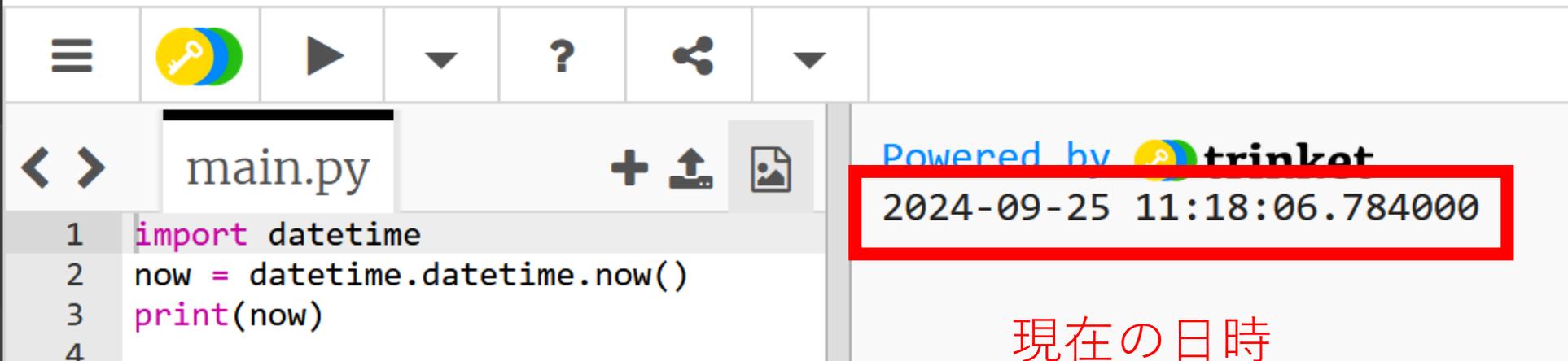
オペレーティングシステム（コンピュータ）のタイマー
を利用. **現在の日時が表示される**

① trinket の次のページを開く

<https://trinket.io/python/2b804ab19a>

② 実行結果が、次のように表示されることを確認

```
import datetime
now = datetime.datetime.now()
print(now)
```



The screenshot shows the Trinket IDE interface. On the left, a code editor displays a Python script named 'main.py' with the following code:

```
1 import datetime
2 now = datetime.datetime.now()
3 print(now)
4
```

On the right, the execution output is displayed, showing the current date and time: '2024-09-25 11:18:06.784000'. This output is highlighted with a red rectangular box. Above the output, it says 'Powered by trinket'.

現在の日時

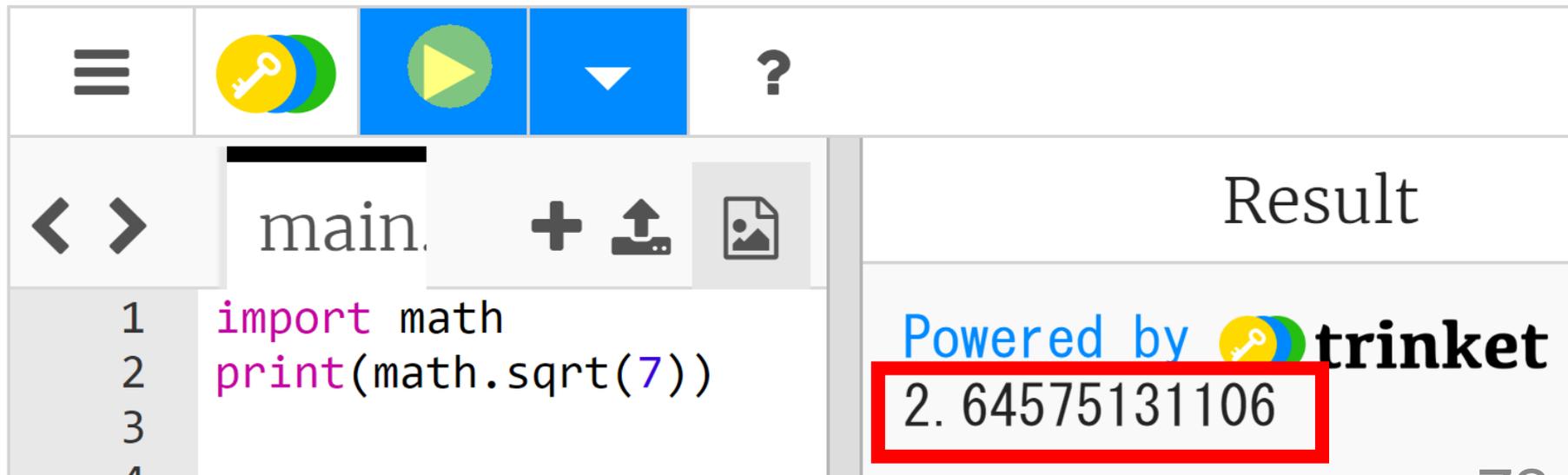
面積が **7** の正方形の一辺の長さ

③ trinket の次のページを開く

<https://trinket.io/python/597e5771ff>

④ 実行結果が、次のように表示されることを確認

```
import math
print(math.sqrt(7))
```



Result

Powered by  trinket

2.64575131106

半径 **3** の円の面積は？



⑤ trinket の次のページを開く

<https://trinket.io/python/4e3559f879>

⑥ 実行結果が、次のように表示されることを確認

```
import math
print(3 * 3 * math.pi)
```

A screenshot of the Trinket.io Python IDE interface. The top bar contains navigation icons: a hamburger menu, a key icon, a play button, a dropdown arrow, and a question mark. Below this, the file name "main.py" is displayed. The code editor shows the following Python code:

```
1 import math
2 print(3 * 3 * math.pi)
3
4
```

To the right of the code editor, there are icons for adding a new file, uploading a file, and inserting an image. The output area on the right is titled "Result" and displays the text "Powered by trinket" with the key icon. Below this, the numerical result "28.2743338823" is shown and highlighted with a red rectangular box.

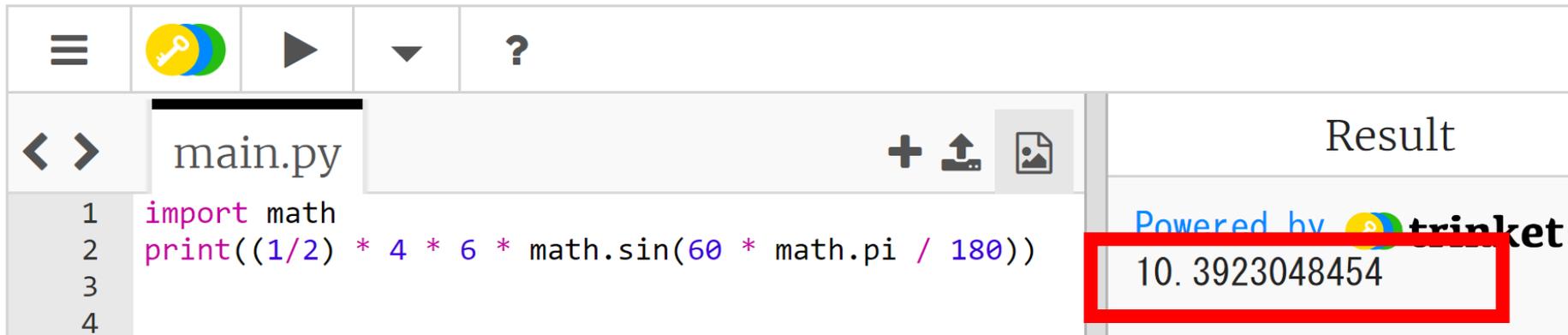
三角形の2辺の長さが、**4**と**6**で、その間の角度が**60**度のとき、面積は $(1/2) \times 4 \times 6 \times \sin(60)$

⑦ trinket の次のページを開く

<https://trinket.io/python/bdcce27488>

⑧ 実行結果が、次のように表示されることを確認

```
import math
print((1/2) * 4 * 6 * math.sin(60 * math.pi / 180))
```



The screenshot shows the Trinket.io Python IDE interface. The code editor displays the following code:

```
1 import math
2 print((1/2) * 4 * 6 * math.sin(60 * math.pi / 180))
3
4
```

The output window on the right shows the result: **10.3923048454**. The result is highlighted with a red box.

プログラミングの可能性



- **人間の力を増幅し、私たちができることを大幅に広げる**
- **シミュレーション、大量データ処理、AI連携、ITシステム制作など、さまざまな活動で役立つ**
- **プログラミングはクリエイティブな行為**
- **さまざまな作業を自動化したいとき、問題解決したいときにも役立つ**
- **論理的思考力の向上**
- **問題解決能力の育成**
- **デジタル社会での必須スキル**

プログラミングの応用分野



- **Web開発**

フロントエンド (HTML, CSS, JavaScript) , バックエンド (Python, Django, Flask)

- **データ分析**

ビッグデータ処理, 統計分析, データビジュアライゼーション

- **人工知能**

自然言語処理, コンピュータビジョン, 予測モデリング

- **ゲーム開発**

2Dゲーム, 3Dゲーム, モバイルゲーム

- **IoT (Internet of Things)**

センサーデータの収集と分析, スマートホームシステム

- **サイバーセキュリティ**

ネットワークセキュリティ, 暗号化技術

Python 言語の特徴とプログラミングの可能性まとめ



- Pythonは**直感的で読みやすい**文法を持つプログラミング言語.
- **文法のシンプルさ, 拡張性, 柔軟性**が特徴的である.
- Pythonは**基本的な計算から高度なプログラミング**まで幅広く対応可能である.
- **Web開発, データ分析, AI, ゲーム開発**など応用分野が多岐にわたる.
- プログラミングは**人間の能力を増幅し, 創造的な活動を支援**する.

プログラミングにおける注意点 ①

1. コンピュータにも、できないことがある。全ての問題を解決できるわけではないことを理解しよう、
2. コンピュータを使用するからといって、計算が常に完全に正確であるとは限らない。特に複雑な計算を行う場合には、精度に注意が必要。
3. 人間がプログラムを作る際には、書き間違い、勘違い、思い込みなどによるミスが起こり得る。

プログラミングにおける注意点 ②



4. ミスがあり得るため、「プログラムが期待通りに動いているか」を確認するテストは非常に重要.
5. ミスの回避のため、抽象化、モジュール、標準ライブラリの活用などの様々な手段を知っておく.
6. 性能や精度を追求し、問題を解決するために、既存のアルゴリズムを知っておく