ようこそアトリエ秋葉原へ

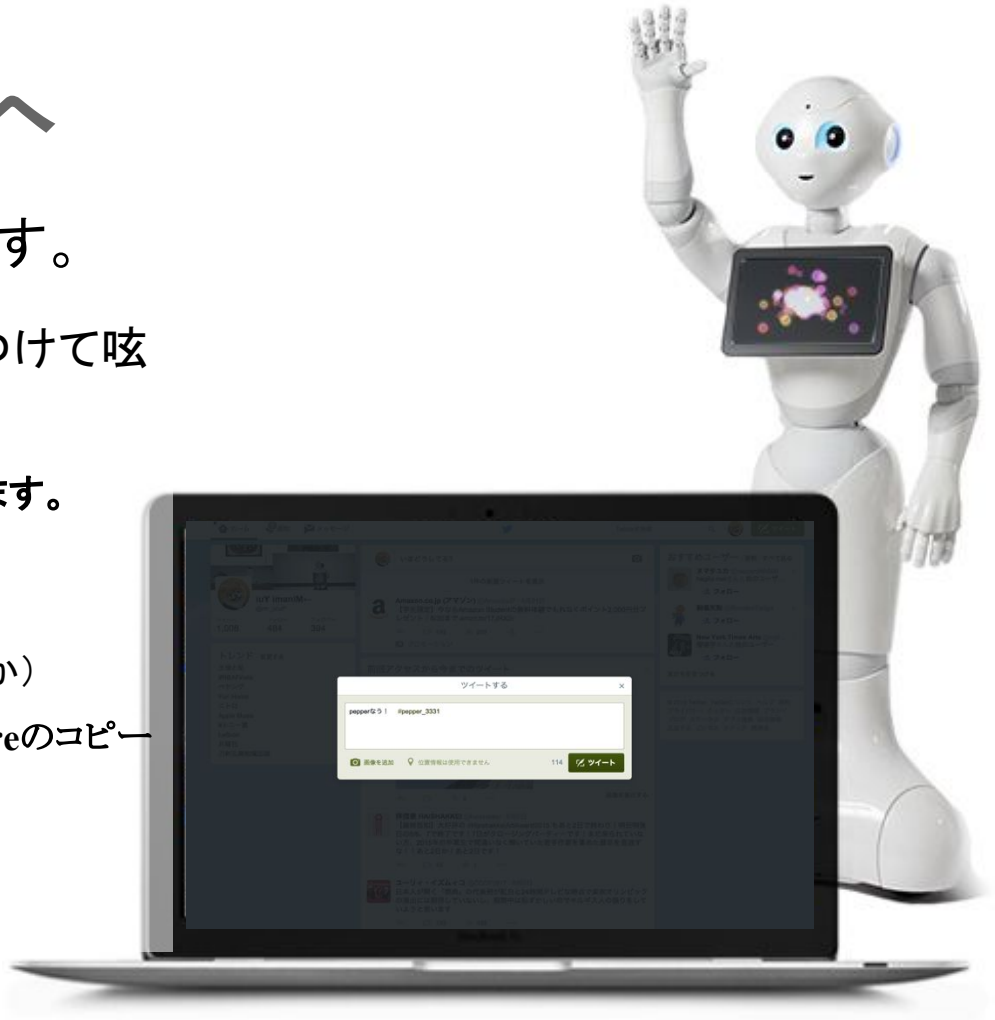
ここのスペースは写真OKです。

ぜひハッシュタグ#pepper_3331をつけて呟いてください

事前にMicrosoftAzureへの登録をお願いします。

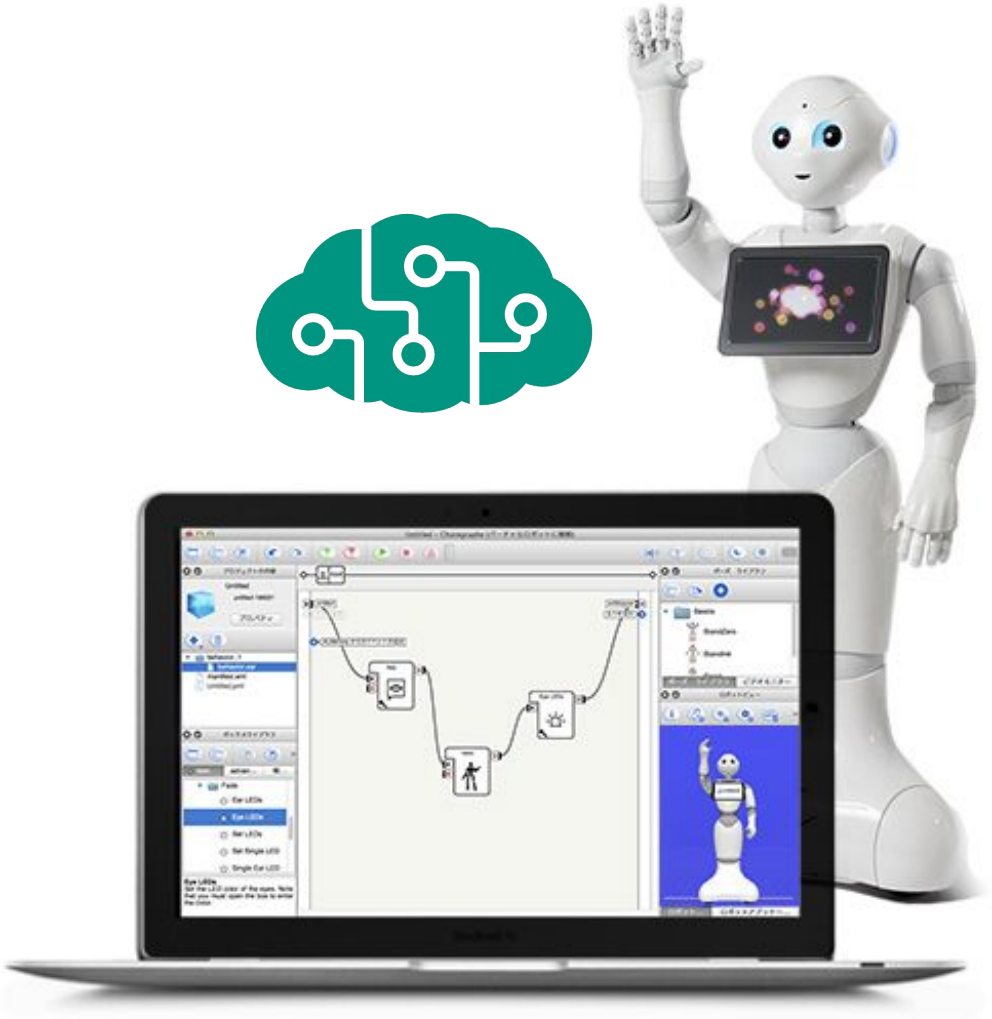
以下データのご用意のご協力をお願いします。

- **顔写真データ3枚程度**（gif .jpg. pngのどれか）
- アトリエのUSBからワークショップ番外編 Azureのコピーをお願いします



Atelier Akihabara

ワークショップ 番外編：Cognitive Service FaceAPI



Microsoft Azure

2018/5/27

免責事項

このワークショップは
アトリエのスタッフが作成したものであり
ソフトバンク公式のものではないことを
ご承知ください。

アトリエについて

実体験とコミュニティーで開発を促進する

アトリエ

- Pepperのアプリ開発を実体験

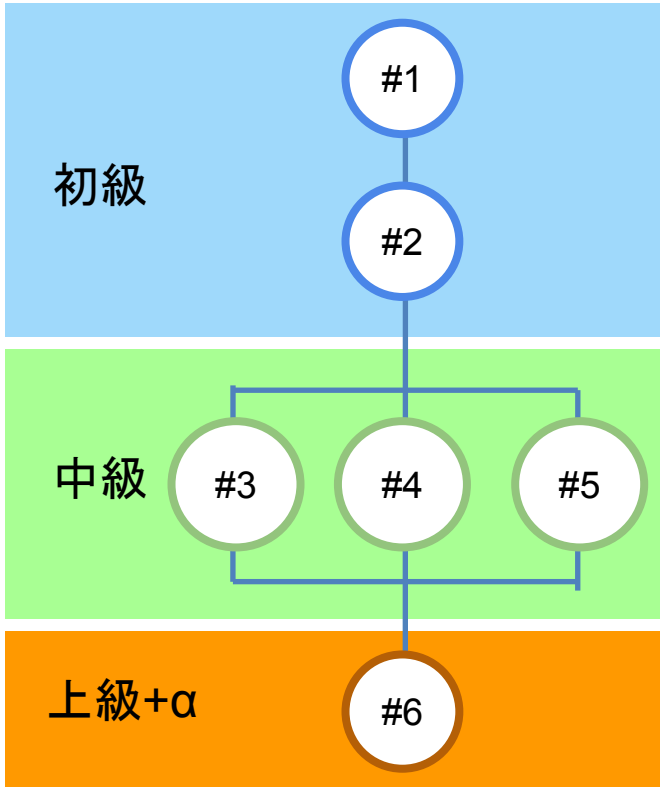
相互
促進

コミュニティー

- 経験や知見を共有

アトリエ秋葉原について

ワークショップ



タッチアンドトライ

自由に開発

質問はスタッフに

お客様同士の交流
検証や
打ち合わせの利用も可

1週間の予定

月	タッチアンドトライ
火	貸し切り(有料)
水	Pepper for Biz説明会 ＆タッチアンドトライ
木	貸し切り(有料)
金	タッチアンドトライ ＆ワークショップ
土日	タッチアンドトライ ＆ワークショップ

アトリエについて

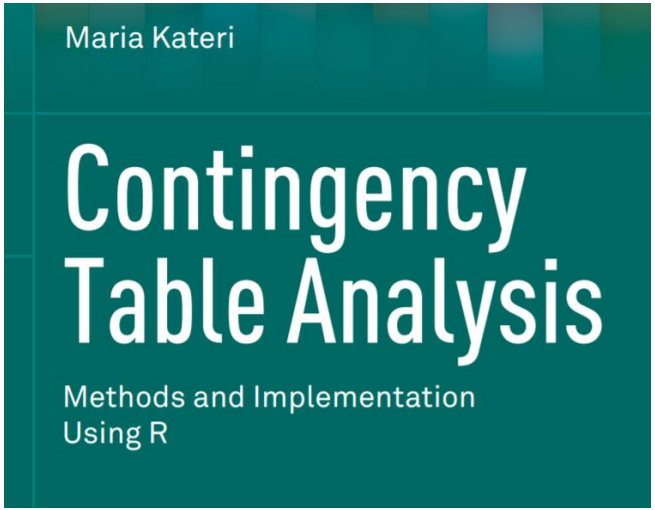
実体験とコミュニティーで開発を促進する

アトリエサテライト

有志でPepperと開発スペースを
提供している
企業、大学、コミュニティスペース

秋葉原で回答できない質問は
各サテライトへ

講師紹介



- 落合 達也(Tatsuya Ochiai)
- 現在、大学院で統計学を専攻中

はじめに

- **お名前**
- **所属**
- **本日の意気込み**
- **プログラミング経験など**

例:

本日の案内を勤めさせていただきます、

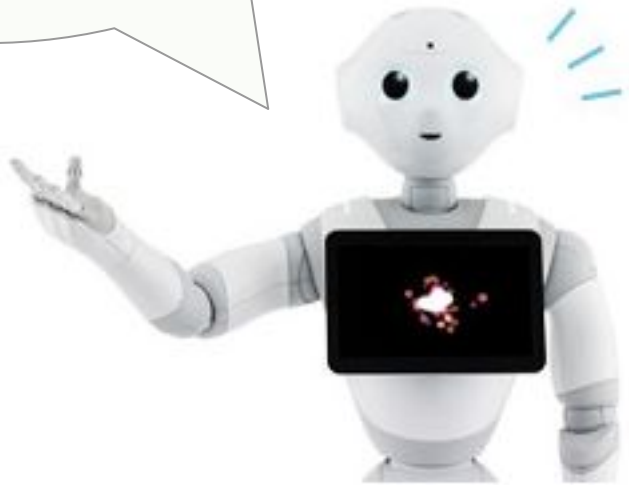
落合　達也(Tatsuya Ochiai)と申します。

本日の内容

1. Microsoft Azure Cognitive Serviseとは
2. Face API について
3. 顔検出させてみよう
4. Requests 処理について
5. 画像を登録して転移学習を行おう
6. Pepper から顔を判別しよう

Microsoft Azure ??

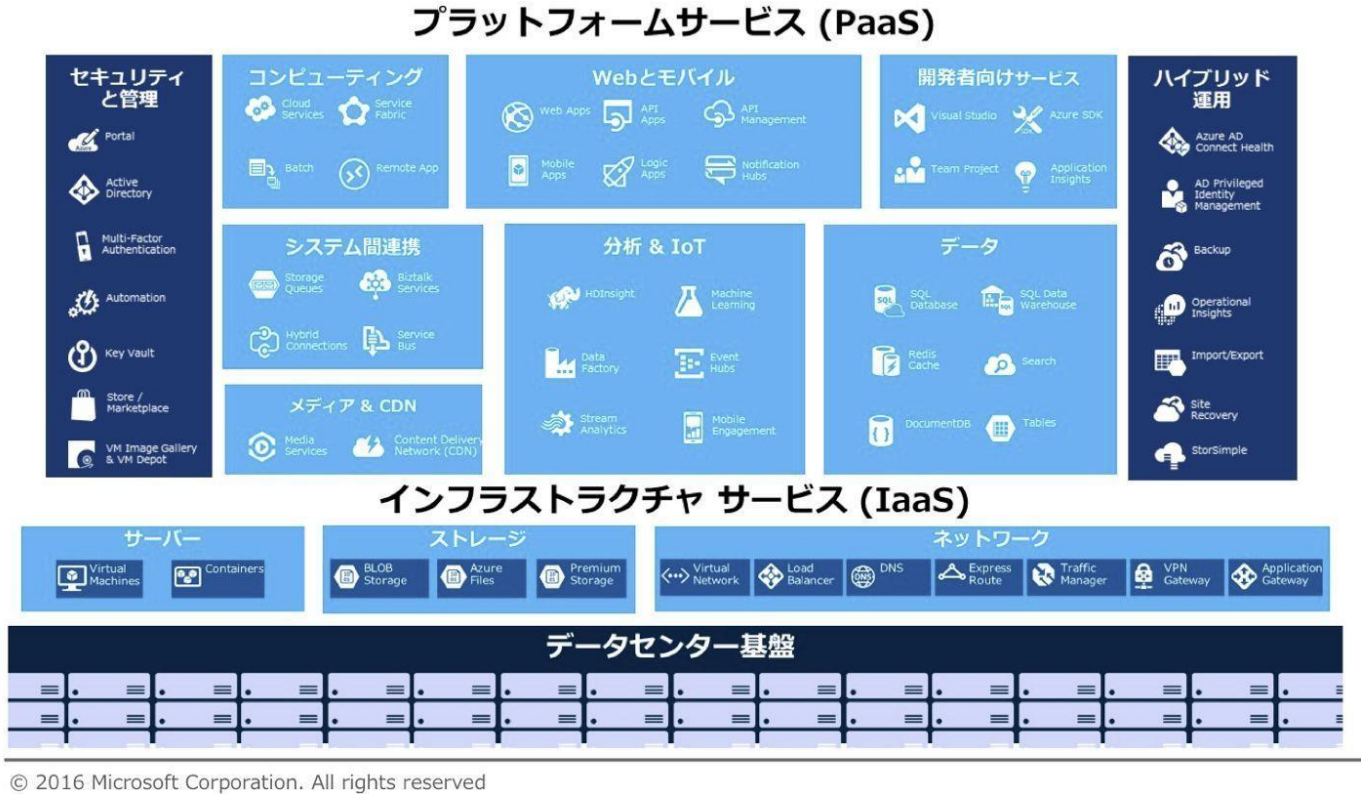
Cognitive Service ???



Microsoft Azure Cognitive Serviceとは？

Azure とは?

Microsoft Azure（マイクロソフト・アジュール）は、マイクロソフトのクラウド プラットフォーム (PaaS/IaaS) である



Microsoft Azure Cognitive Serviceとは？

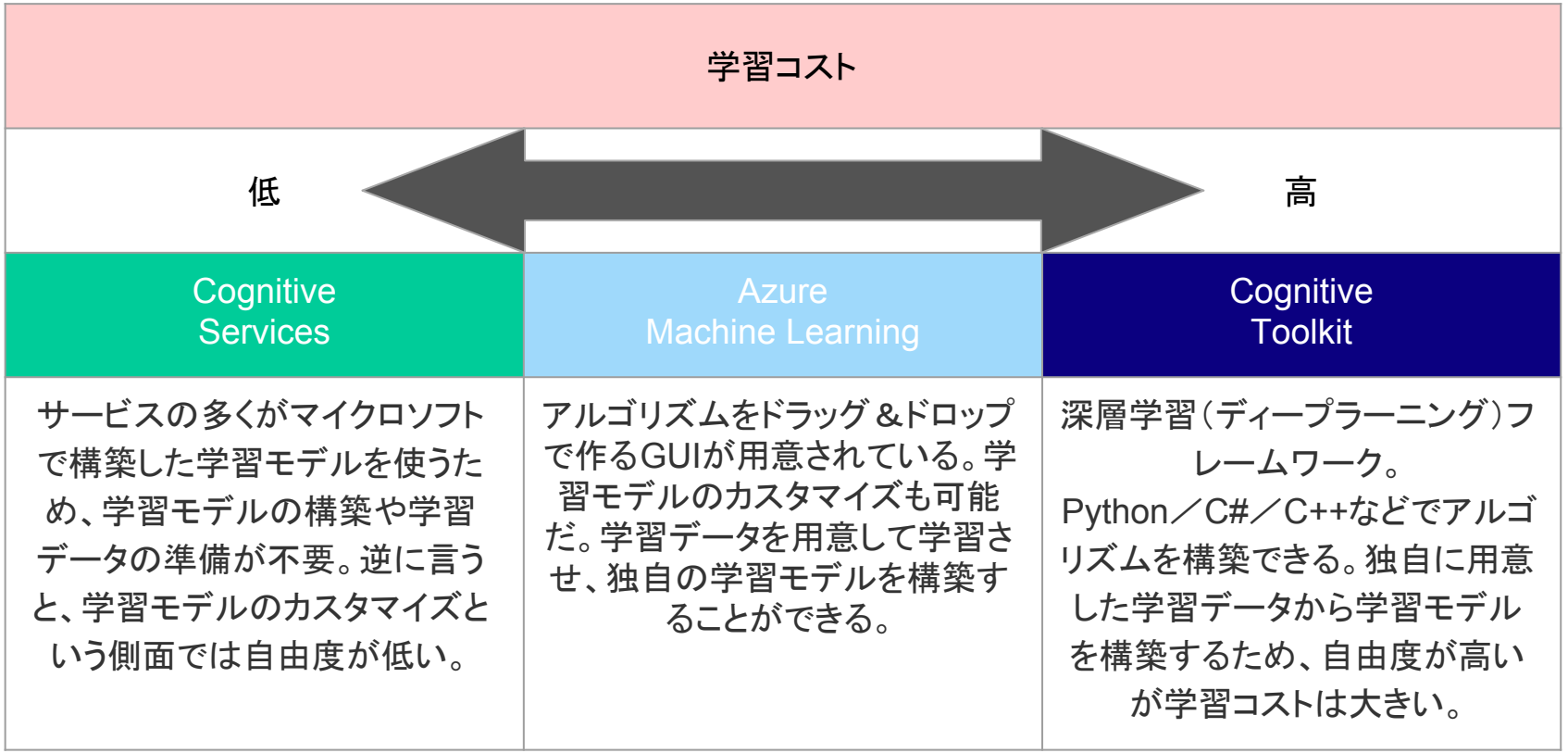
Cognitive Service とは?

- Microsoft Azure内の製品の一つとして提供されている。最先端の深層学習（ディープラーニング）のアルゴリズムを用いた評価済みの学習結果をAPIとして活用できる。つまりAPIを呼び出すだけで、AIのサービスを利用できる。

- REST形式のWeb APIの呼び出しが可能で多くのプログラミング言語で（基本的に）利用できる。また、C#、Python、Androidなど、さまざまな環境向けのSDK／クライアントライブラリ／サンプルも用意されている。

- 多くのサービスが一定の無料枠付きの従量課金スタイルになっている。料金体系については、「Cognitive Servicesの料金に関するページ」で確認していただきたい。実際にプロダクションとして運用をするのであれば、要件や設計の段階から料金体系を考慮することは重要。

Microsoft Azure Cognitive Serviceとは？



参照: https://www.buildinsider.net/small/cognitiveservices/01

Microsoft Azure Cognitive Serviceとは？

主な機能は5つ！！

AI を使用してビジネスの問題を解決しましょう

視覚

イメージ処理アルゴリズムで、写真をスマートに識別したり、キャプションを追加したり、 モデレートしたりできます。

音声

音声をテキストに変換したり、確認に音声を使用したり、 アプリに話者の認識機能を追加したりできます。

知識

インテリジェントなお勧め機能やセマンティック サーチといったタスクを解決するために、 複雑な情報やデータをマッピングします。

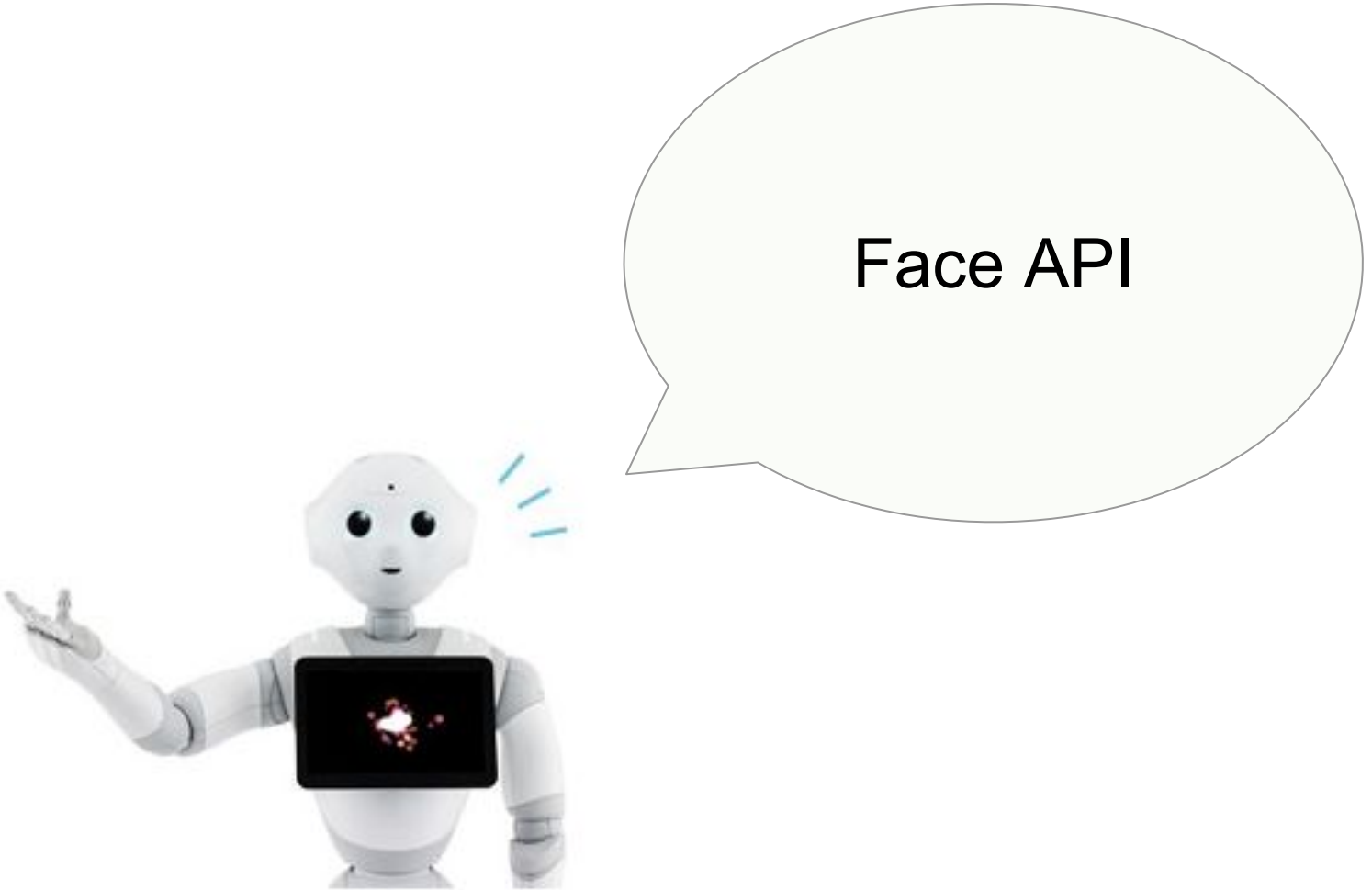
Search

Bing Search APIs をアプリに追加して、1 つの API 呼び出しで 何十億という Web ページやイメージ、ビデオ、ニュースをくまなく調べる機能を実装しましょう。

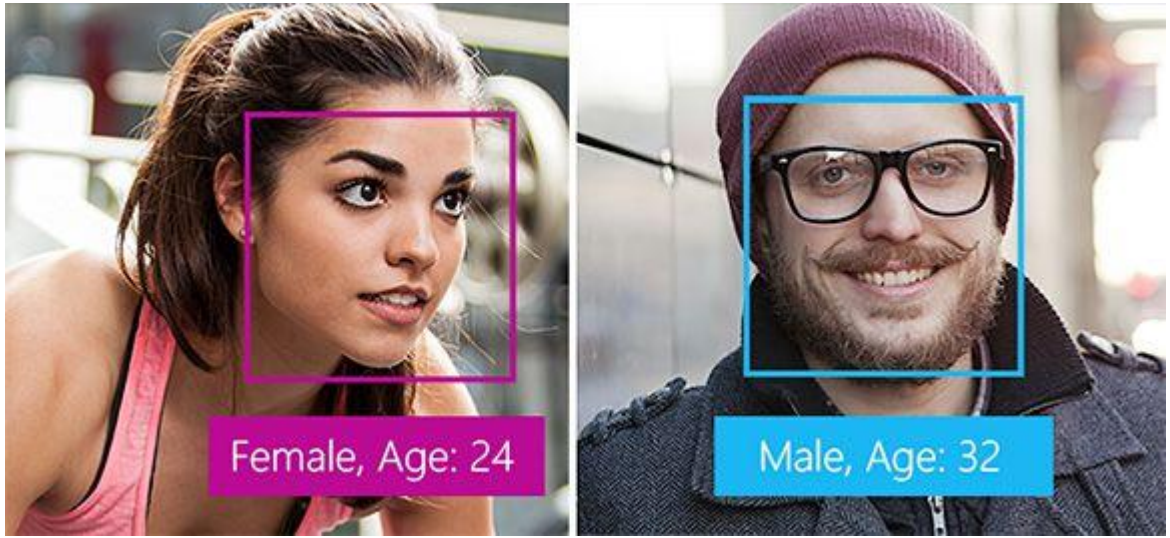
言語

アプリが、事前に構築済みのスクリプトで自然言語を処理したり、センチメントを評価したり、ユーザーが望んでいることの認識方法を学習したりできます。

詳しくは「https://www.buildinsider.net/small/cognitiveservices/01」



Face API (顔検出)



- 画像内の高精度の顔の位置を持つ最大64人の人間の顔を検出。イメージはバイト単位または有効なURLでファイルを指定できる。
- 任意選択で、[性別] [年齢] [笑顔] [髭] [顔の回転] [眼鏡] [感情] [髪色] [メイク] [アクセサリー] [画像のぼやけ具合] [顔の露出具合] [ノイズ]

Face API (顔認識)

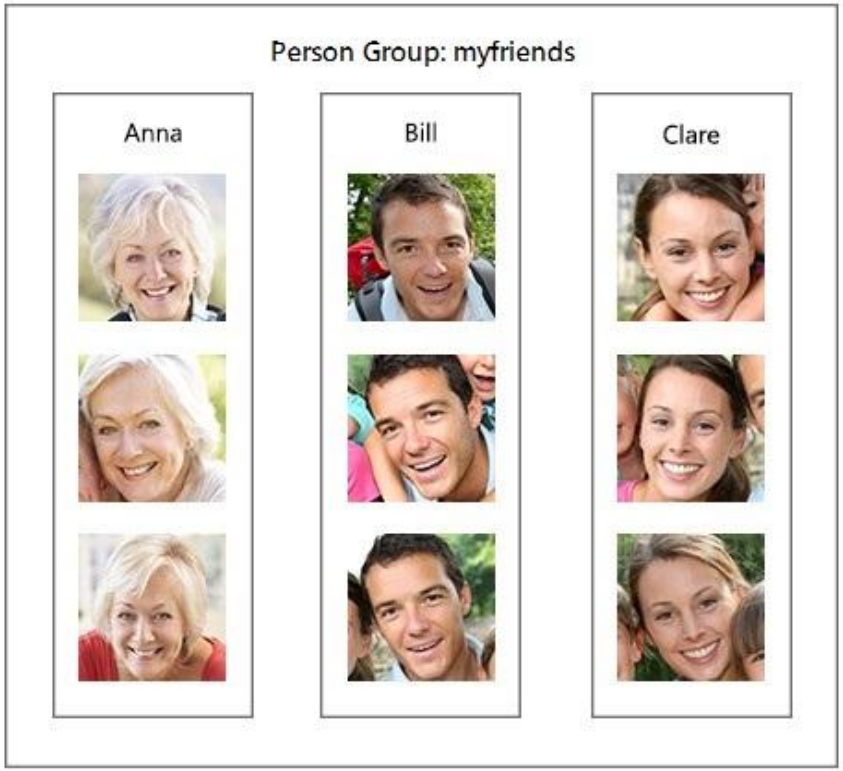
顔認識は、セキュリティ、自然なユーザーインターフェイス、画像コンテンツの分析と管理、モバイルアプリ、ロボットなど多くのシナリオで広く使用されています。

主に4つの顔認識機能が用意されています:
[顔の確認] [似た顔の検索] [顔のグループ分け] [人物の識別]

Face API (顔認識)

Face APIは、検出された顔とpeopleデータベース（LargePersonGroup / PersonGroupとして定義）に基づいて人を識別する。このデータベースは、あらかじめ作成しておくことができます。

(右図は、"myfriends"という名前のLargePersonGroup / PersonGroupの例です。各グループには、最大1,000,000人/ 10,000人のオブジェクトが含まれます。一方、各人物オブジェクトは、最大248の顔を登録することができます)



Face API (ストレージ)

LargePersonGroup / PersonGroup Person オブジェクトまたは LargeFaceList / FaceLists を使用して、Face APIに 保存された画像は1000人の顔につき$ 0.5で毎日比で課金される。

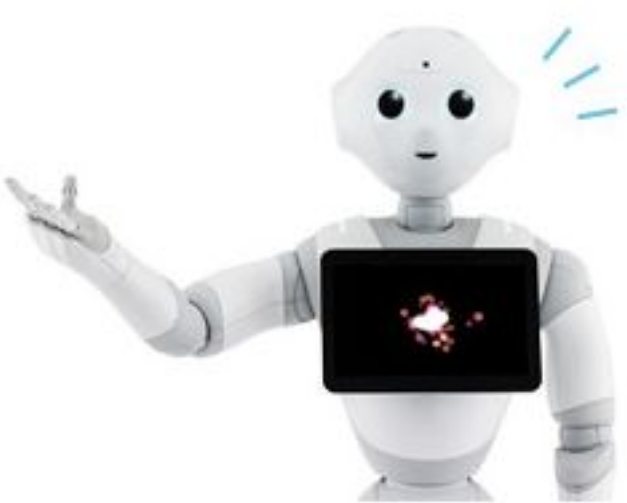
無料のサブスクリプションは無料なので、1,000人に制限されている。

フェイスストレージの価格は毎日比で計算されるとは？？

例えば、月の前半に1万人の顔を登録して、後半に1人も使用されていない場合は、保管された日の10,000人の顔だけが請求される。また、毎日 1,000人の顔を数時間だけ登録してそのあと削除すると、毎日1000人分の課金がされること。

参照: https://docs.microsoft.com/ja-jp/azure/cognitive-services/face/overview

顔を検出
させてみよう



顔を検出しよう（デモ）

デモページ: https://azure.microsoft.com/ja-jp/services/cognitive-services/face/#detection

検出結果:
JSON:


```
[
 {
  "faceId": "65cd0685-44c2-4b74-ab89-ea2c9b682f5e",
  "faceRectangle": {
   "top": 128,
   "left": 459,
   "width": 224,
   "height": 224
  },
  "faceAttributes": {
   "hair": {
    "bald": 0.0,
    "invisible": false,
    "hairColor": [
     {
      "color": "brown",
      "confidence": 1.0
     },
     {
      "color": "blond"
```

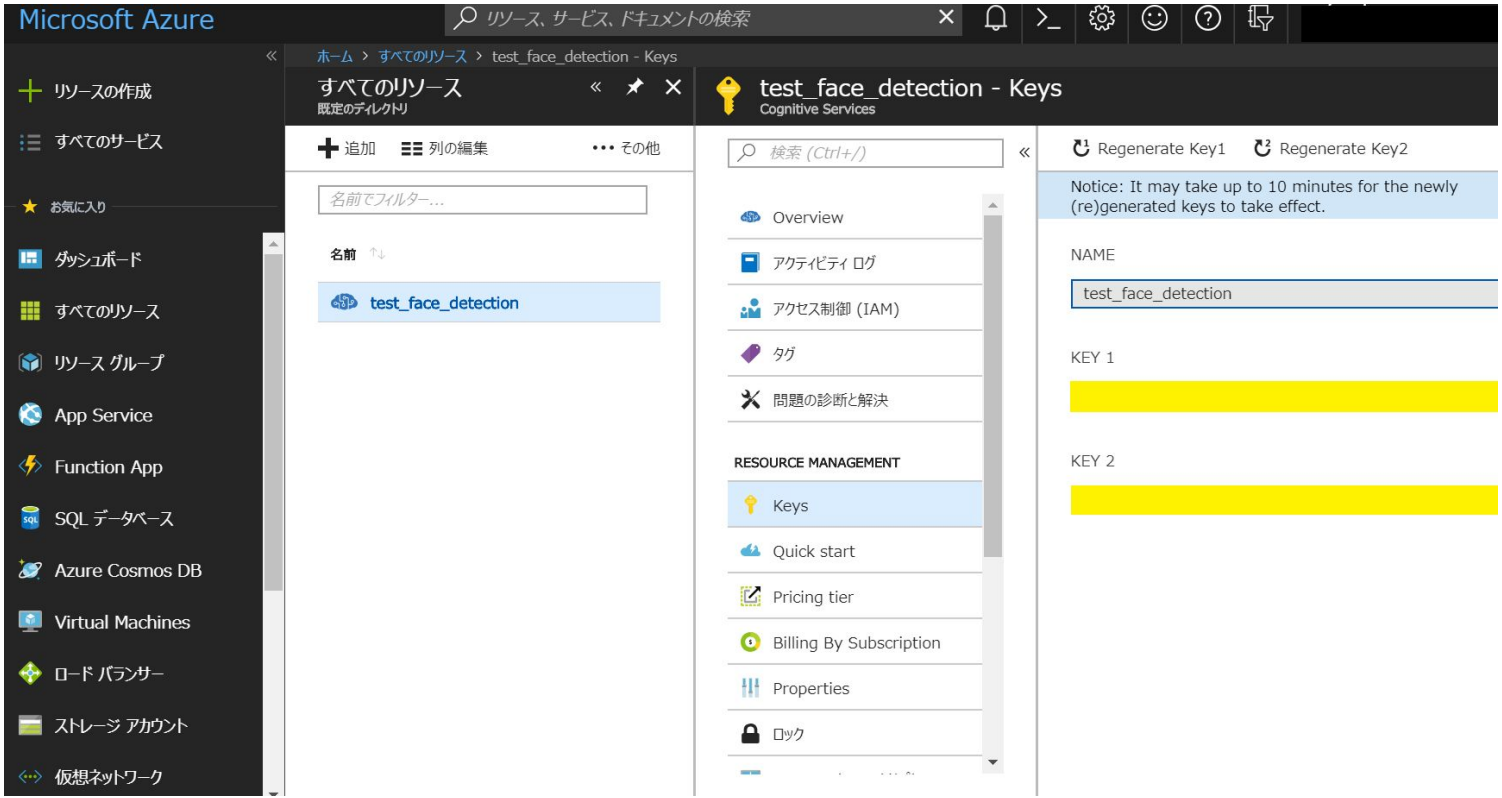
顔を検出しよう

※注意…横向き，画像が小さすぎる場合は失敗する恐れがある

```
検出結果:
JSON:
 []
```

- JPEG, PNG, GIF (the first frame), BMP format 形式をサポート
- ファイルサイズは 1KB から 4MB

顔を検出しよう（下準備）

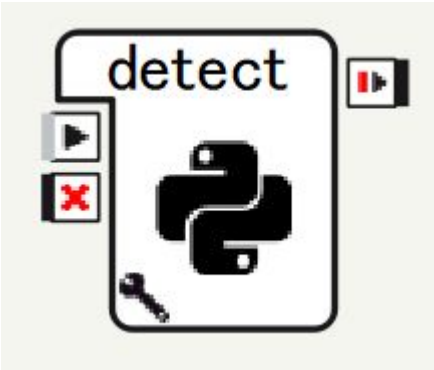


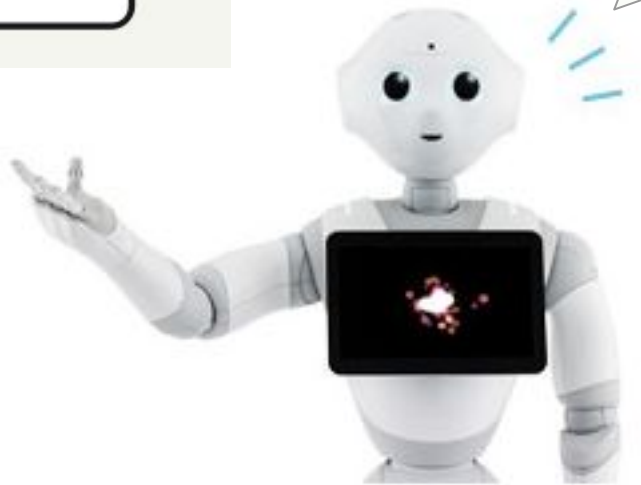
1: Azure に登録

2: アプリケーションの作成

3: Keyの取得

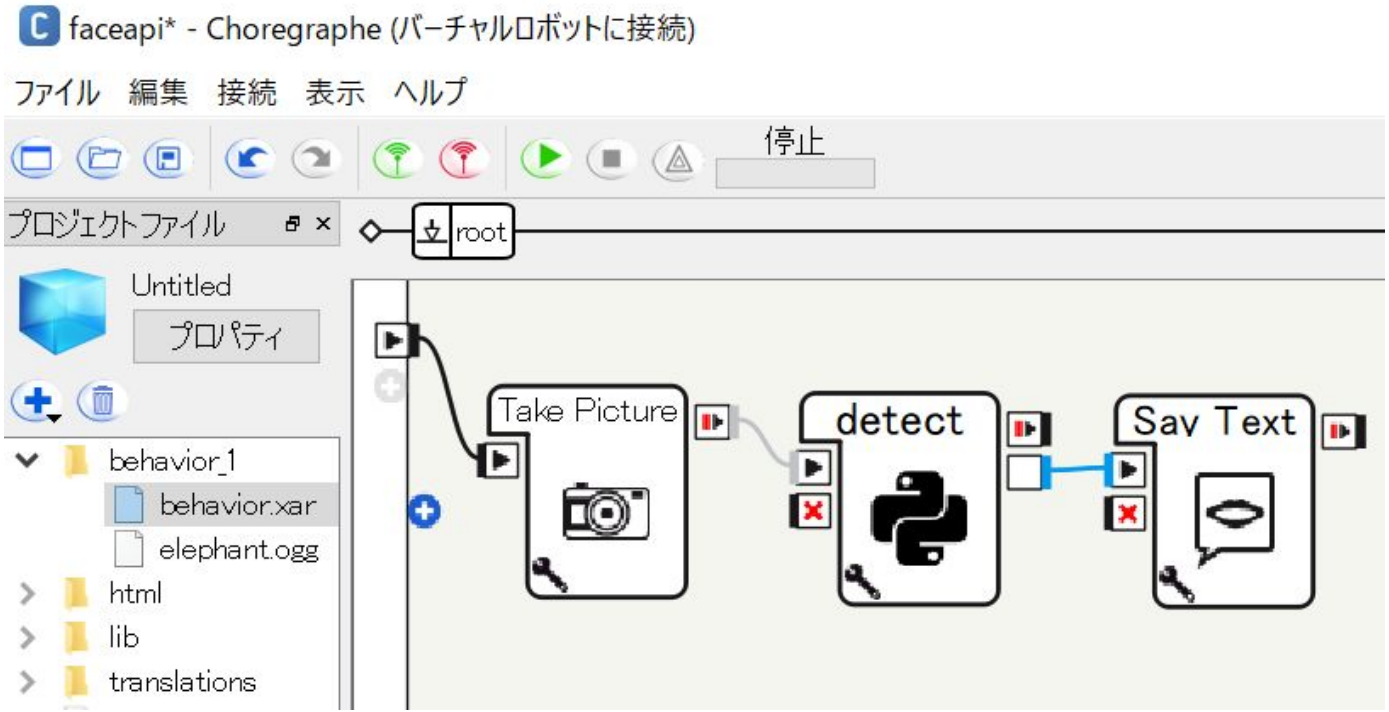
Azure ポータルサイト(https://portal.azure.com/)





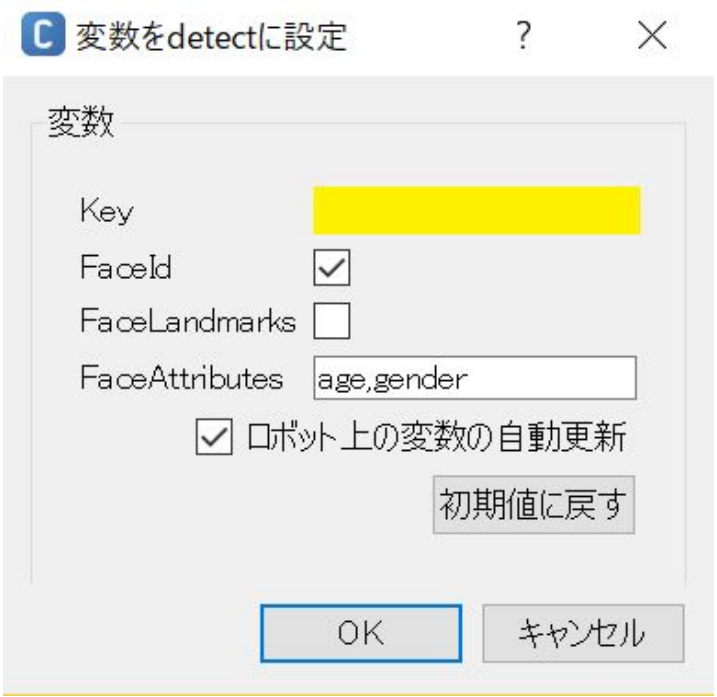
顔を検出しよう

ボックスを以下のようにつないで下さい



顔を検出しよう

パラメータを設定しよう

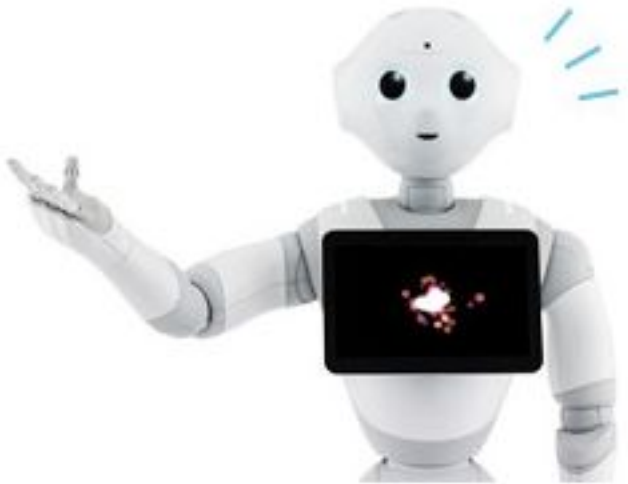


[Key]...サブリプションキー

[FaceId]...detected face APIによって返されるID、呼び出されてから24時間後に自動的に消滅

[FaceLandmarks]...顔の特徴を27の特徴点で詳細を出す。

[FaceAttributes]... [age, gender, headPose, smile, facialHair, glasses, emotion, hair, makeup, occlusion, accessories, blur, exposure, noise] などオプションを付けられる



Requests
について

Requests について

- requestsはPythonモジュールの1つ
- HTTP(s)通信を行うためのライブラリ
- 標準のurllibよりもかなり使い勝手良くHTTPリクエストを発行することができる。
- APIにアクセスするにはHTTP(s)通信は必須！！！

参照: http://docs.python-requests.org/en/v2.3.0/

```
atelier43 [0]   $ pip show requests
---
Name: requests
Version: 2.3.0
Location: /usr/lib/python2.7/site-packages
Requires:
```

Naoqi2.5.5ではrequests==2.3.0が標準搭載されている！
※一般的なrequestsの最新verは2.19.1(2018/6/15現在)

Requests の使い方について

r = requests.[ ](url, headers, params, data)

[.get] ...リソースの取得
[.post] ...リソースへのデータ追加
[.put] ...リソースの更新、作成
[.delete] ...リソースの削除
[.head]...リソースのヘッダ
[.option]...リソースがサポートしているメソッドの取得
[.trace]...プロキシの動作確認
[.connect]...プロキシ動作のトンネル接続への変更

参照: https://qiita.com/Ryutaro/items/a9e8d18467fe3e04068e

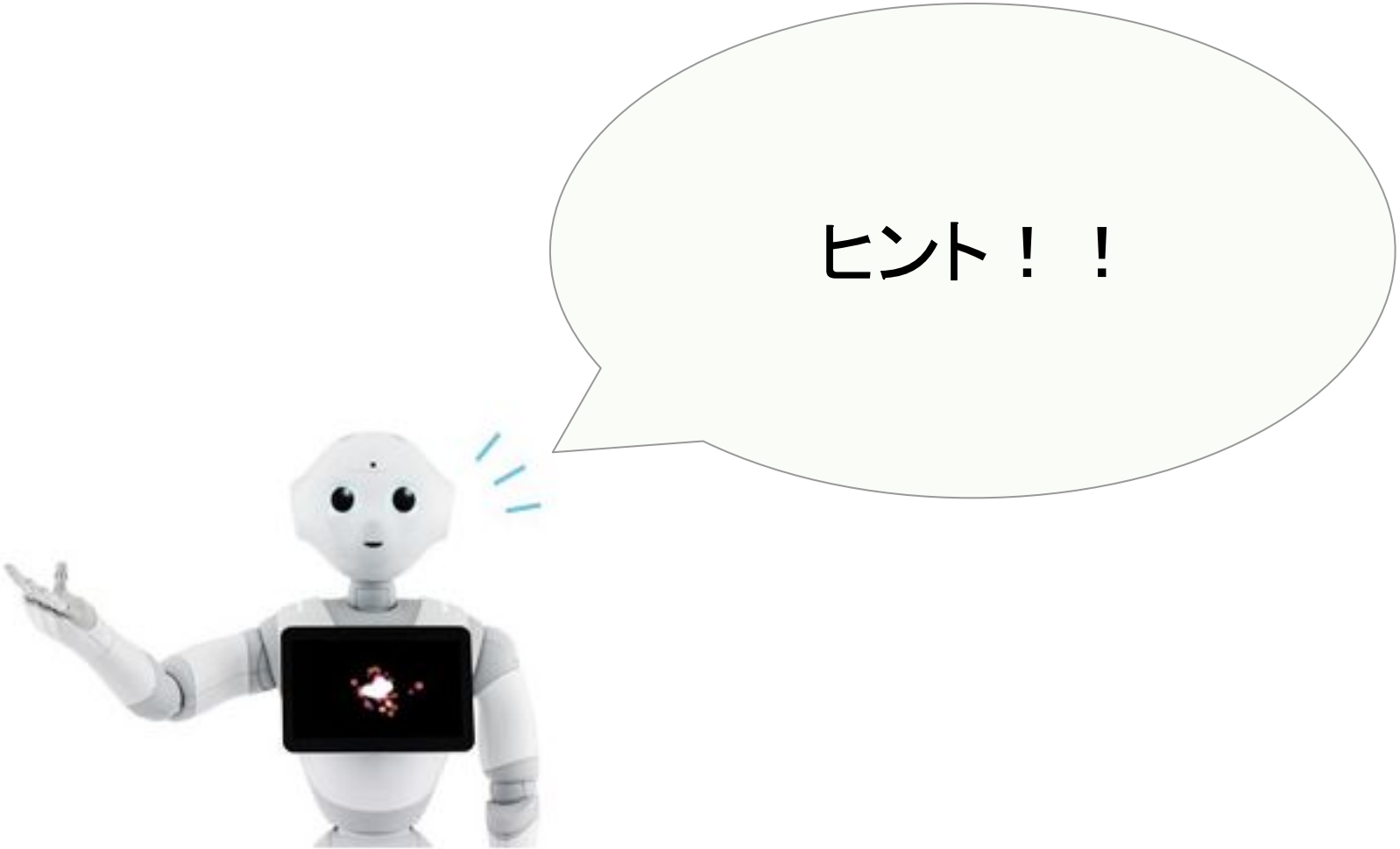
Requests のサンプルコード

```
>>> import requests
>>> r = requests.get("http://www.google.com/")
>>> r.status_code
200
>>> r.headers['content-type']
'text/html; charset=ISO-8859-1'
>>> r.encoding
'ISO-8859-1'
>>> r.text
'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ja"><head><meta cont
;&#12422;&#12427;&#24773;&#22577;&#12434;&#26908;&#32034;&#12377;&#12427;&#12383;&#12417;&#12398;&
75;&#12390;&#12356;&#12414;&#12377;&#12290;&#12373;&#12414;&#12374;&#12414;&#12394;&#26908;&#32034
2390;&#12289;&#12362;&#25506;&#12375;&#12398;&#24773;&#22577;&#12434;&#35211;&#12388;&#12369;&#123
escription"><meta content="noodp" name="robots"><meta content="text/html; charset=UTF-8" http-equi
world-cup-2018-day-11-5692104616443904-5663741160980480-ssw.png" itemprop="image"><meta content="&
03; - Day 11" property="twitter:title"><meta content="&#12373;&#12354;&#35430;&#21512;&#38283;&#22
#GoogleDoodle &#12391; &#19990;&#30028;&#20013; &#127758;&#127757;&#127759;&#12398; &#9917;&#12434
roperty="twitter:description"><meta content="&#12373;&#12354;&#35430;&#21512;&#38283;&#22987;&#123
odle &#12391; &#19990;&#30028;&#20013; &#127758;&#127757;&#127759;&#12398; &#9917;&#12434;&#24540
og:description"><meta content="summary_large_image" property="twitter:card"><meta content="@Google
```


detect Boxを完成させよう！

```
19
20      def onInput_onStart(self, p):#「Take Picture」boxで撮った写真を引数pで受け取っている
21          #アクセスするurl
22          url = 'https://australiaeast.api.cognitive.microsoft.com/face/v1.0/detect'
23          #ヘッダー情報
24          headers = {
25              'Content-Type': 'application/octet-stream',
26              'Ocp-Apim-Subscription-Key': key,
27          }
28          #パラメータ情報
29          params = {
30              'returnFaceId': self.getParameter("FaceId"),  # The default value is true.
31              'returnFaceLandmarks': self.getParameter("FaceLandmarks"), # The default value is false.
32              'returnFaceAttributes': self.getParameter("FaceAttributes"), # age, gender, headPose, smile, facialHair, and
    glasses.
33          }
34          #パス指定の場合
35          #img_url = '/home/nao/.local/share/PackageManager/apps/.lastUploadedChoregrapheBehavior'
36          try:
37              [redacted]
38          except Exception as e:
39              self.logger.info(e)  # 必要であれば失敗時の処理
```

※ requests 処理を付け足してください！



detect Boxを完成させよう！（ヒント）

```
try:
    r = requests.      (      ,headers =          ,params =          ,data = open(    ,"rb"))
except Exception as e:
    self.logger.info(e)   # 必要であれば失敗時の処理
```

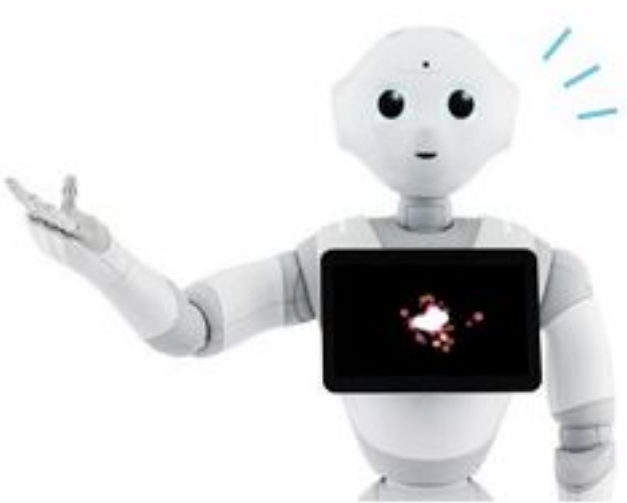
Boxを埋めよう！！

detect Boxを完成させよう！（答え）

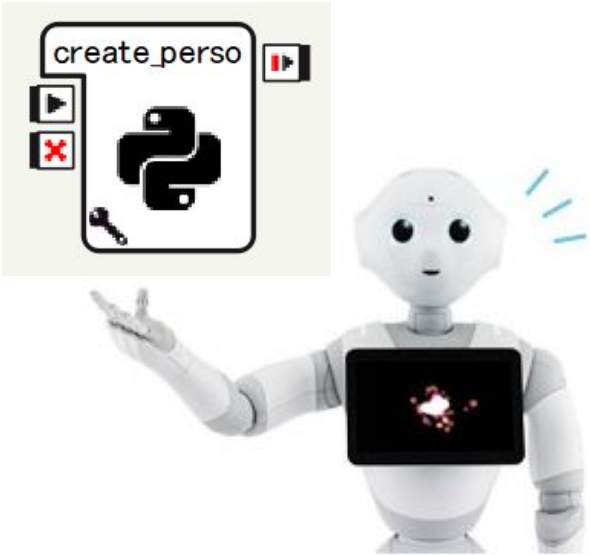
```
try:
    r = requests.post(url ,headers = headers,params = params,data = open(p,"rb"))
except Exception as e:
    self.logger.info(e)   # 必要であれば失敗時の処理
```

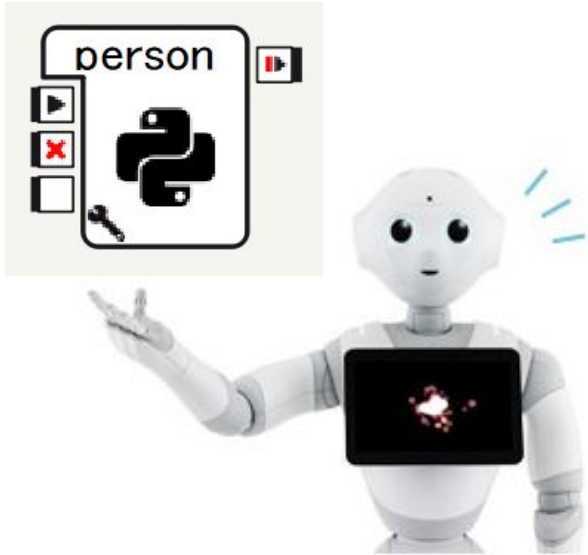
この後扱うボックスにも同様にAPIにアクセスするためのREST処理を行っている！！

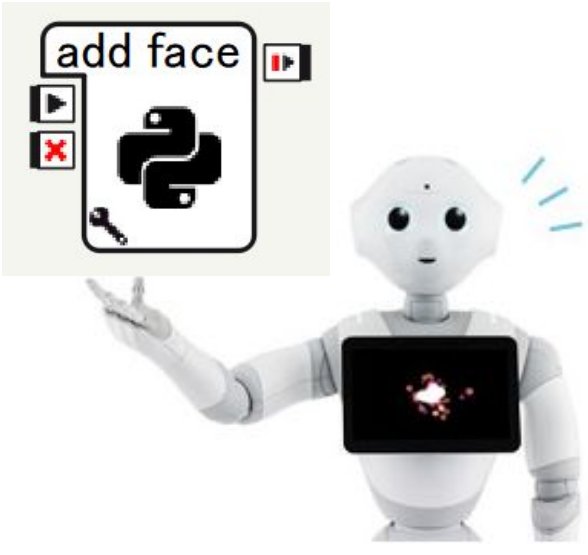
転移学習を行おう！
～下準備編～



その前にちょっと下準備！
使うボックスはコレ！！

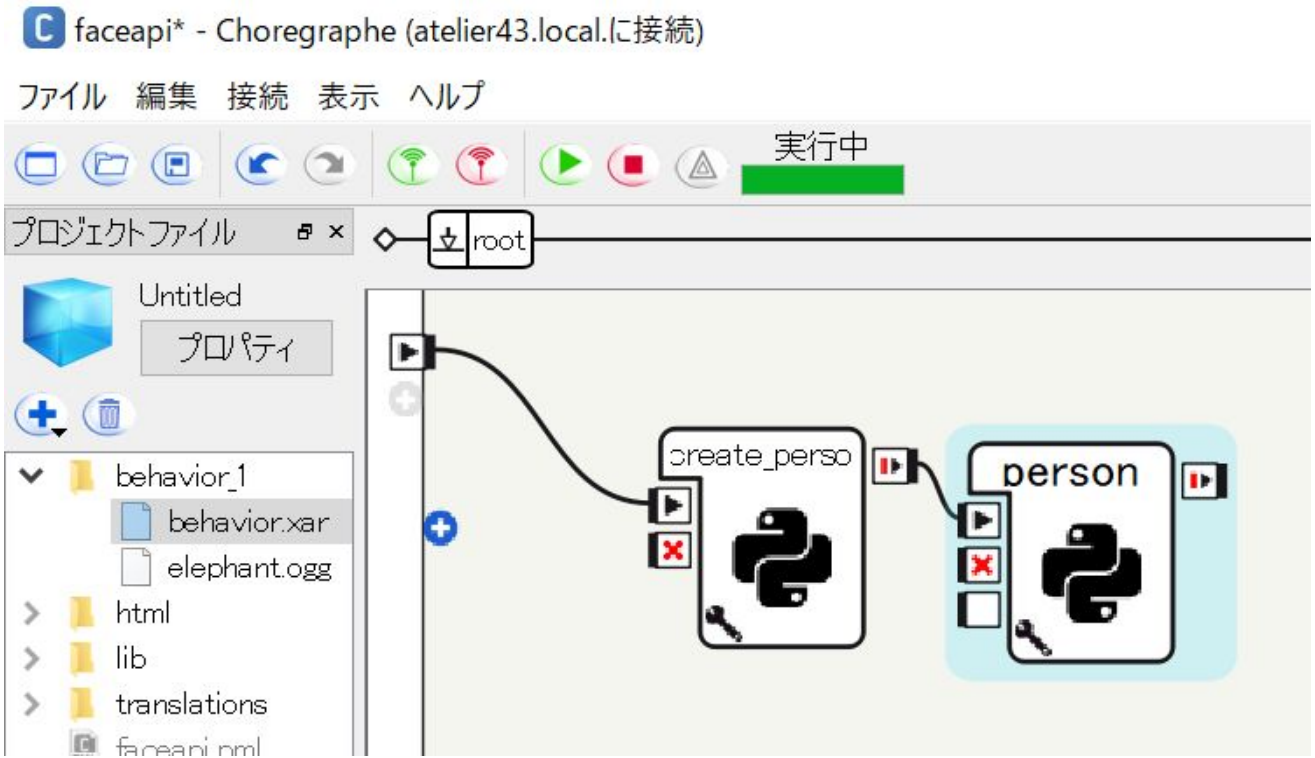




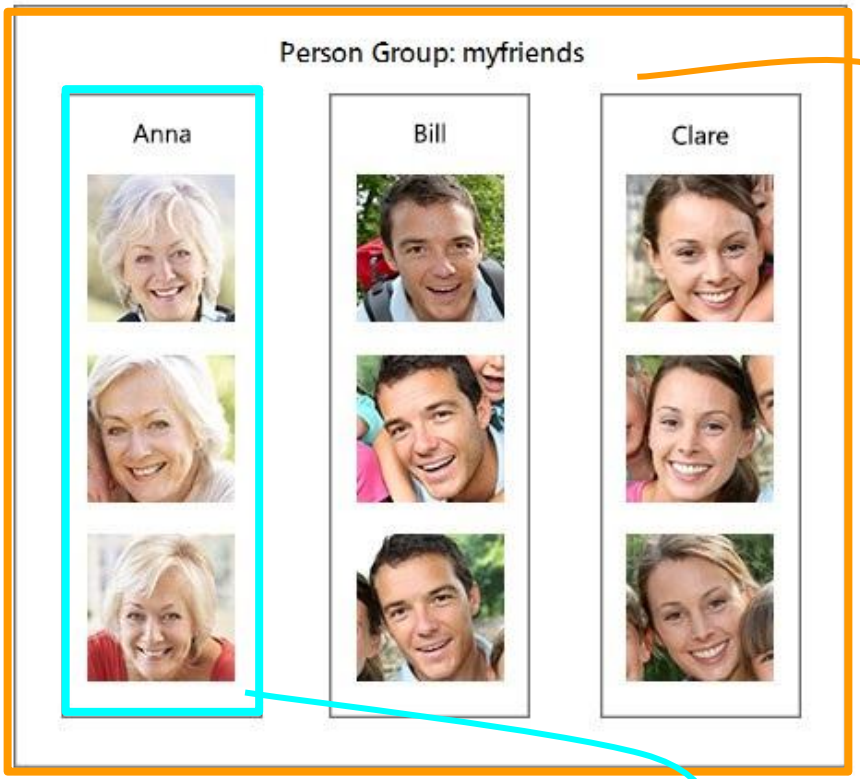


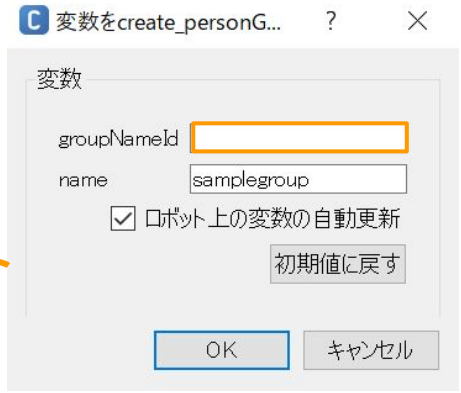
画像を登録して転移学習を行おう

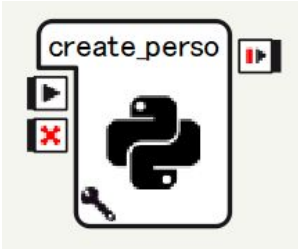
ボックスを以下のようにつないで下さい

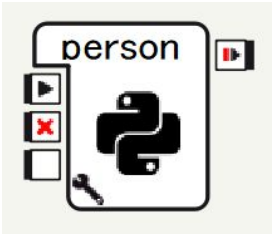


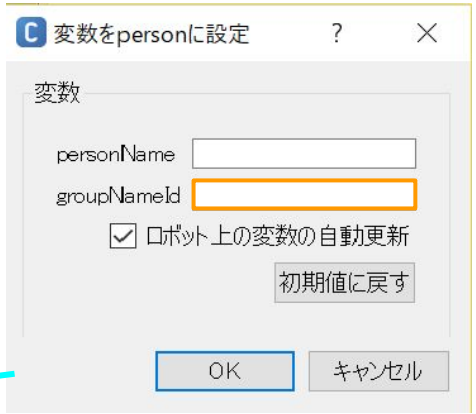
画像を登録して転移学習を行おう





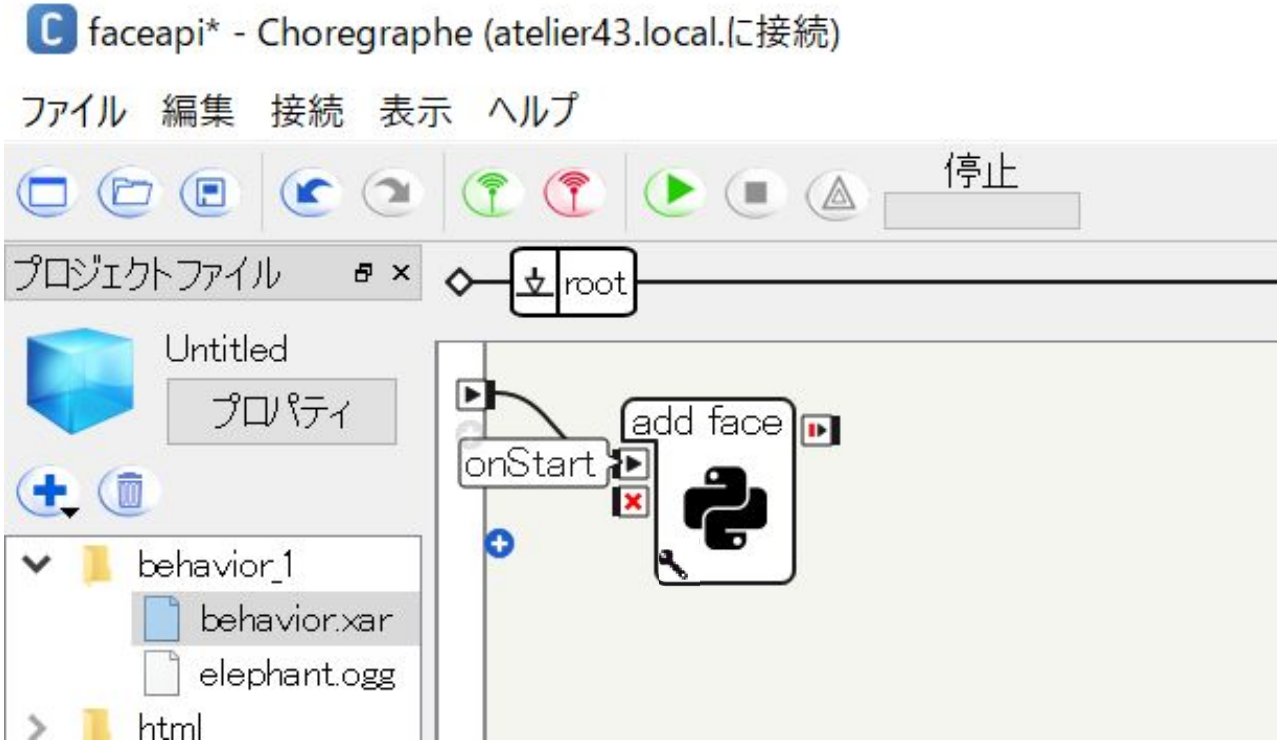




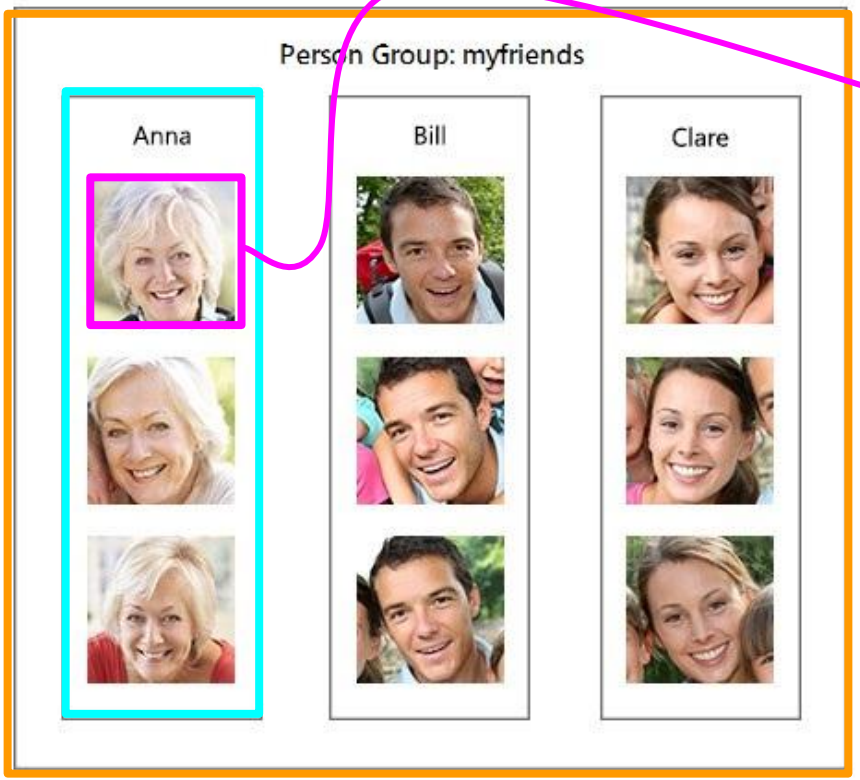


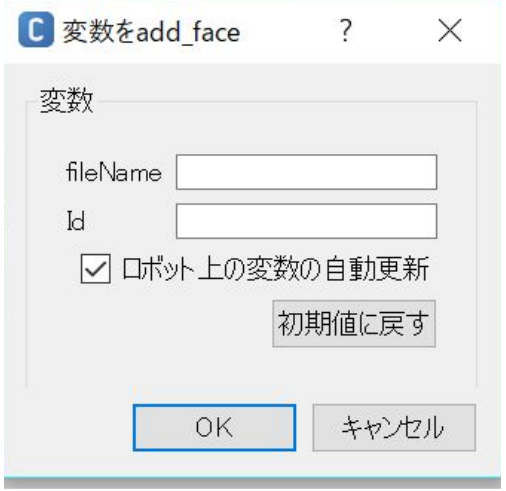
画像を登録して転移学習を行おう

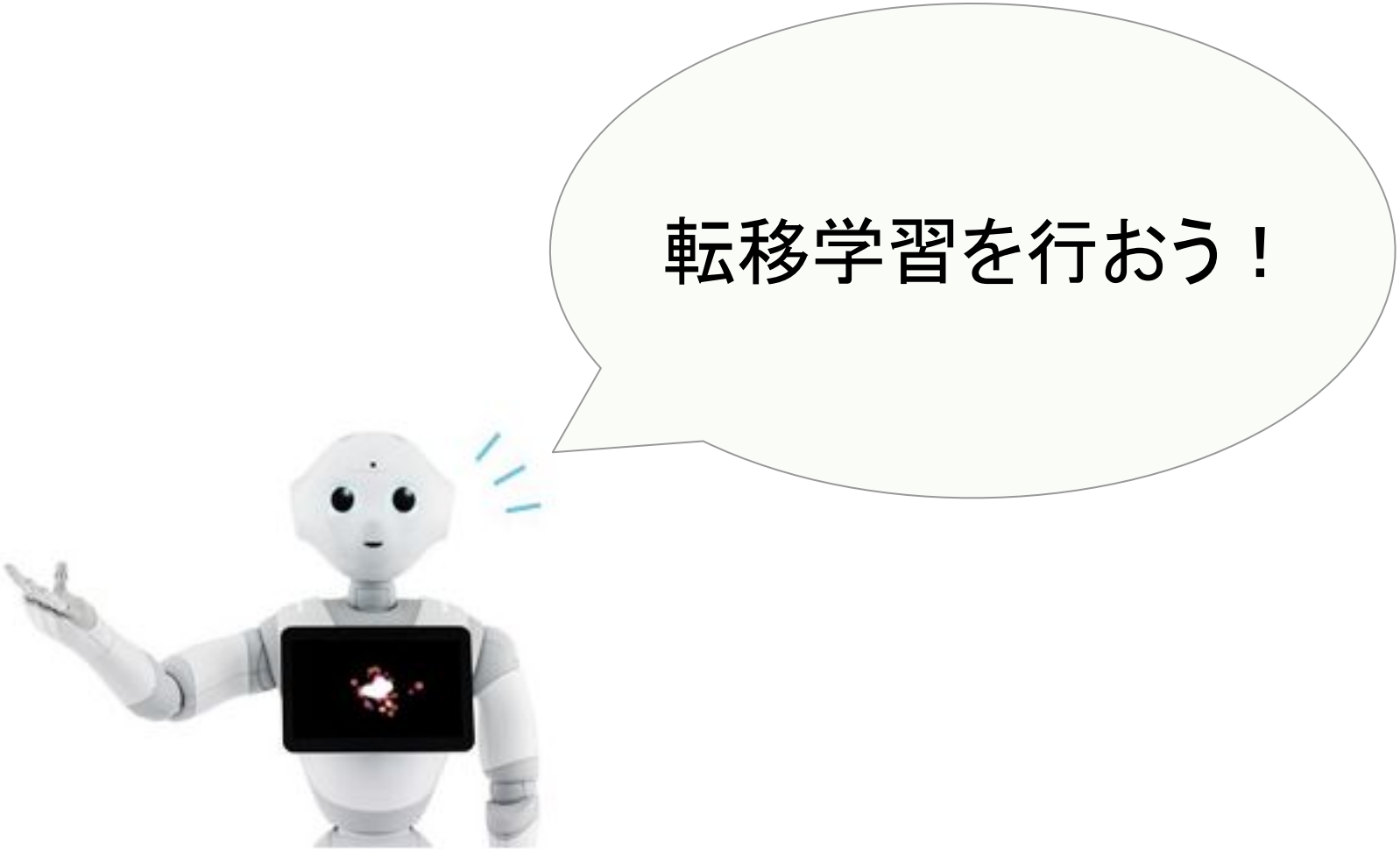
ボックスを以下のようにつないで下さい



画像を登録して転移学習を行おう





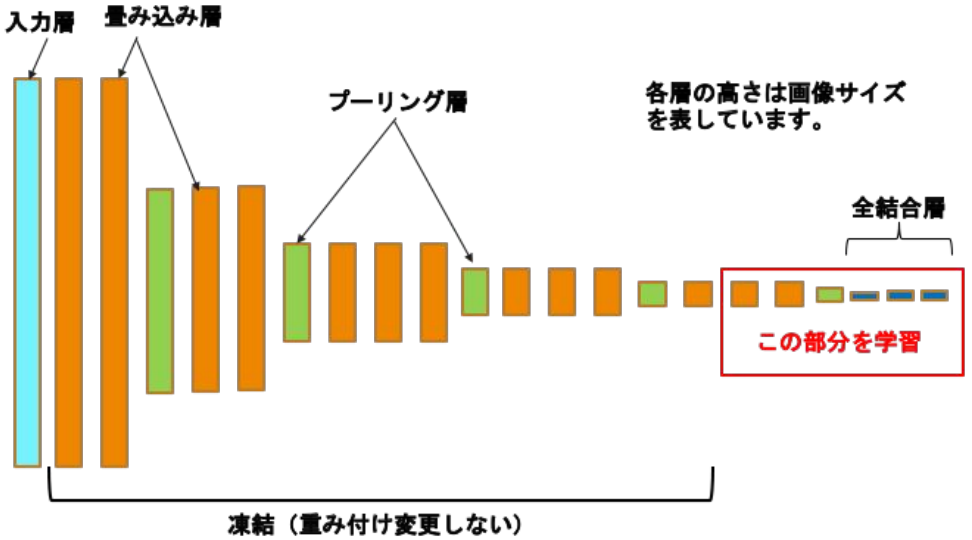


画像を登録して転移学習を行おう

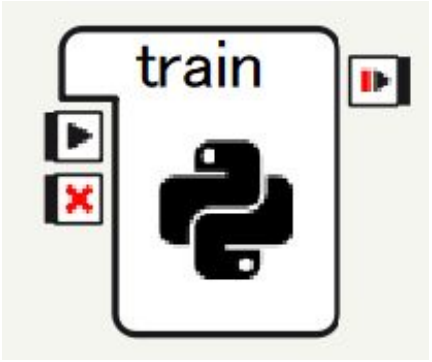
転移学習とは

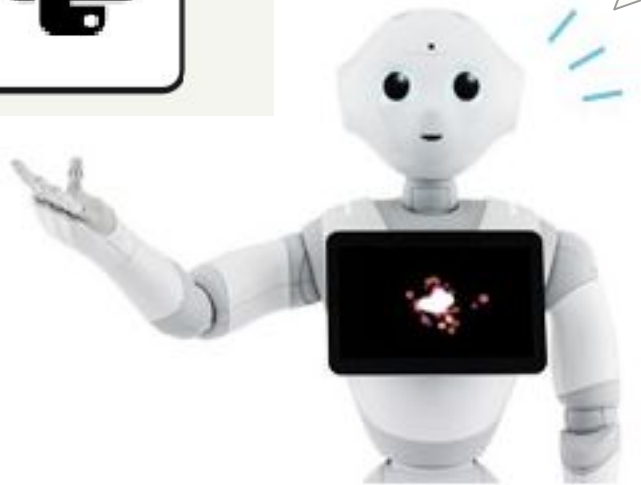
転移学習とは，あらかじめ別の訓練データを用いて学習を行わせたモデルに対して，新たに別の学習をさせることをいう。

ゼロから学習する場合に比べて学習のコストを削減することができる。



参照:
https://wba-initiative.org/wiki/%E6%A9%9F%E6%A2%B0%E5%AD%A6%E7%BF%92/%E8%BB%A2%E7%A7%BB%E5%AD%A6%E7%BF%92
https://products.sint.co.jp/aisia/blog/vol1-7

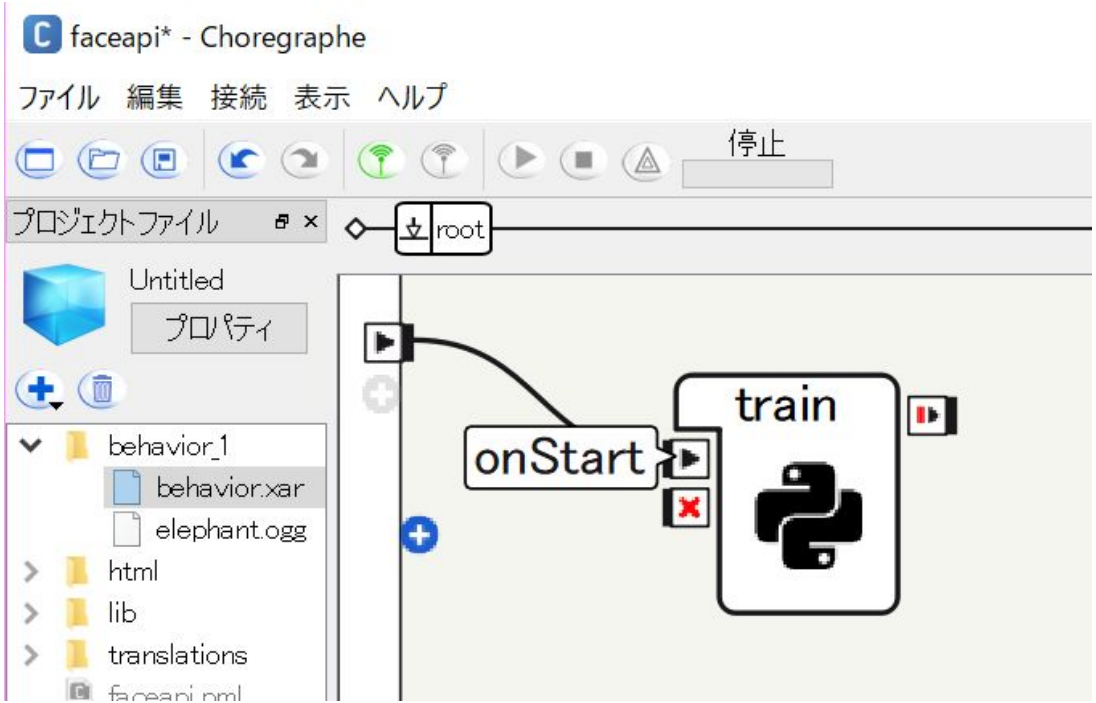




使うボックスはコレ！！

画像を登録して転移学習を行おう

ボックスを以下のようにつないで下さい



画像を登録して転移学習を行おう

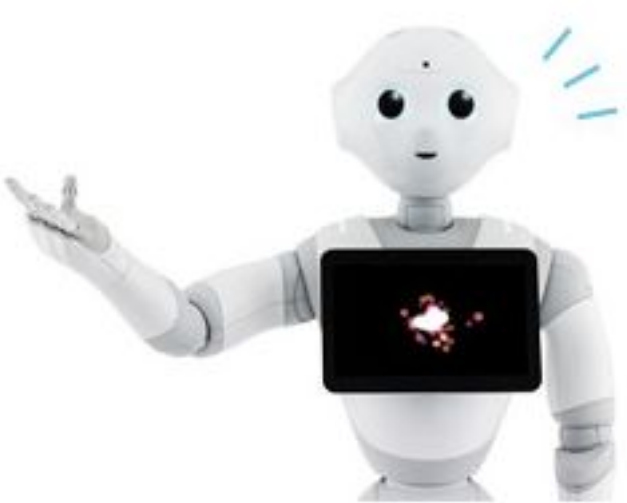
スクリプトエディタ

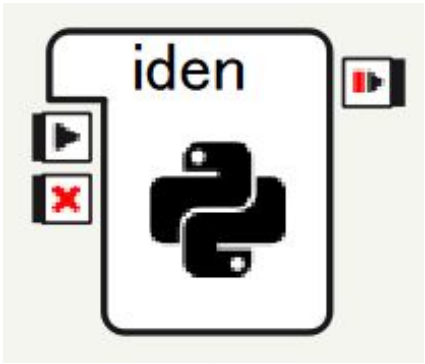
train

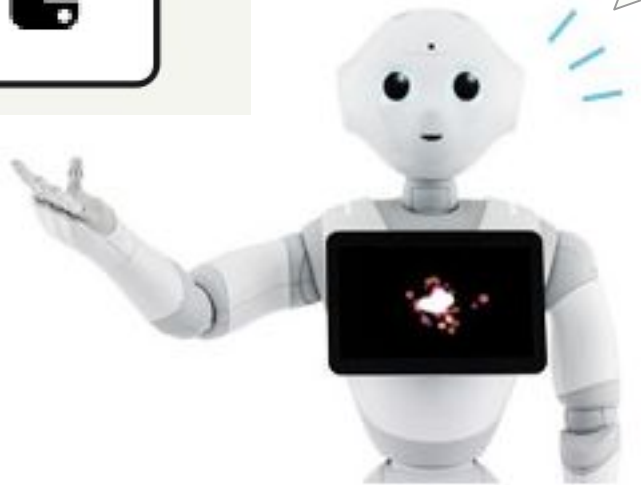
```
    def onInput_onStart(self):

        def train():
            url =   self.url + gnameId + '/train'
            headers = {
                'Ocp-Apim-Subscription-Key': key,
            }
            params = {
                'personGroupId': gnameId,
            }
            try:
                r = requests.post(url, headers = headers, params = params)
            except Exception as e:
                self.logger.info(e)  # 必要であれば失敗時の処理
            else:
                if str(r)=="<Response [202]>":
                    self.logger.info(r)
                    get_train()
                else:
                    self.logger.info(r)

        def get_train():
            url =  self.url + gnameId + '/training'
            params = {
                'personGroupId': gnameId,
            }
            headers = {
                'Ocp-Apim-Subscription-Key': key,
            }
            try:
                r = requests.get(url, headers = headers, params = params)
            except Exception as e:
                self.logger.info(e)  # 必要であれば失敗時の処理
            else:
                if str(r)=="<Response [200]>":
                    self.logger.info(r.json())
                else:
                    self.logger.info(r)
        train()
```

実際に行っているのはAPIへのアクセスのみ

Pepperから
顔を判別しよう

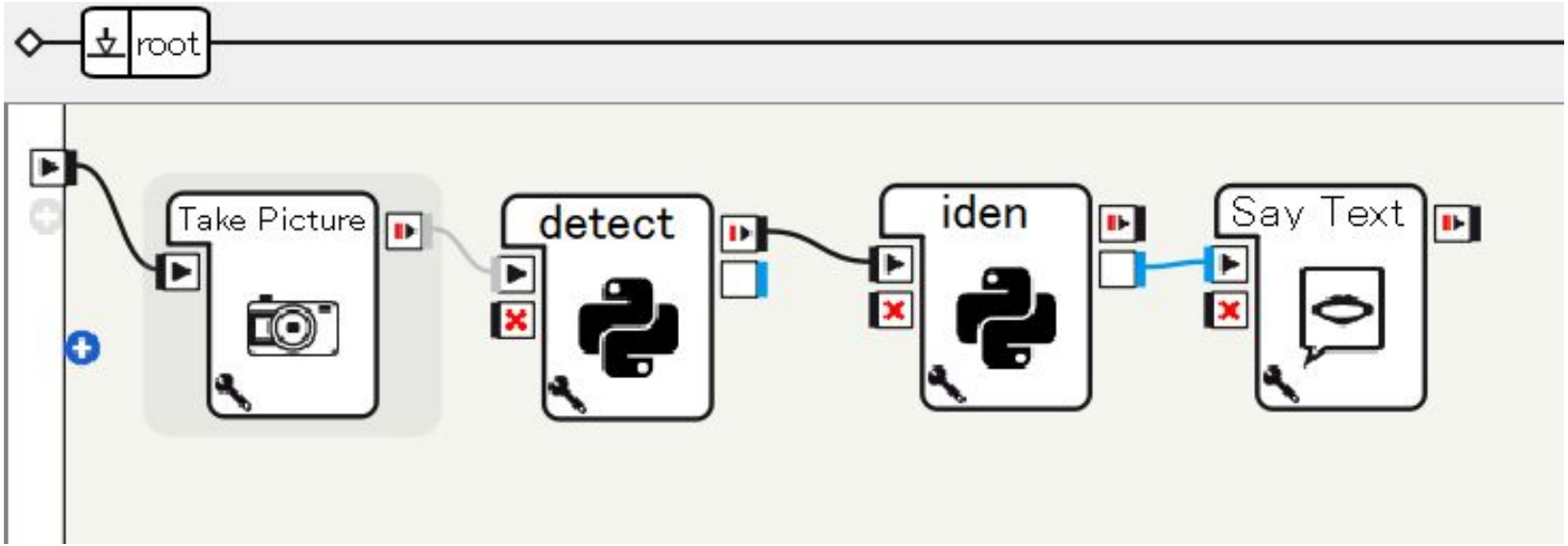






Pepper から顔を判別しよう

ボックスを以下のようにつないで下さい



iden Boxについて

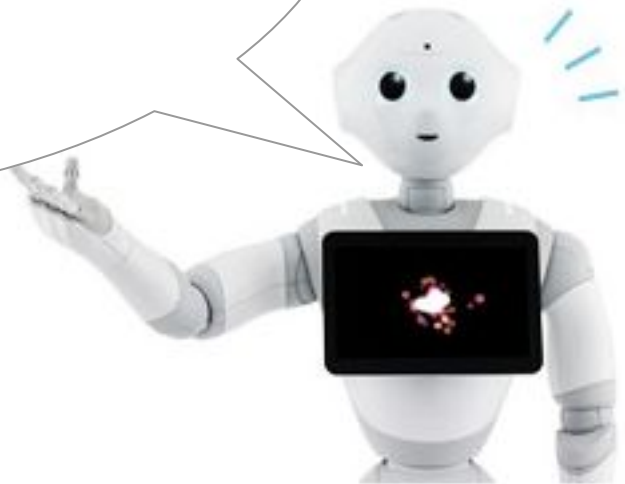
スクリプトエディタ

iden | detect

```
        #put clean-up code here
        pass

    def onInput_onStart(self):
        def iden():
            global faceId
            url = 'https://australiaeast.api.cognitive.microsoft.com/face/v1.0/identify'

            headers = {
                'Content-Type': 'application/json',
                'Ocp-Apim-Subscription-Key': key,
            }

            body = {
                "personGroupId": gnameId,
                "faceIds": [
                    faceId
                ],
            "maxNumOfCandidatesReturned": 10,
            "confidenceThreshold": 0.5
            }
            try:
                rr = requests.post(url, headers = headers, data = json.dumps(body))
            except Exception as e:
                self.logger.info(e)  # 必要であれば失敗時の処理
            else:
                self.logger.info(rr)
                if str(rr)=="<Response [200]>":
                    self.logger.info(rr.json())
                    word = 'あなたは'+str(rr.json()[0]['candidates'][0]['confidence'])+'ぐらいの確率で'+str(rr.json()[0]['candidates'][0]
['personId'].encode('utf-8'))+'さんです'
                    self.word(word)
                else:
                    self.logger.info("response error")
        iden()
```

おつかれさまでした！
これにてWS番外編は終わりになります。
WSは続けてぜひ受講してみてください

タッチアンドトライで質問もしてみてください



番外編

AzureとIoTを使った
具体的な活用事例は
こちらへ

ヘッドウォータースさんのホームページ
https://www.headwaters.co.jp/

アンケートへのご協力お願いします

https://bit.ly/pepperatelier