

ようこそアトリエ秋葉原へ

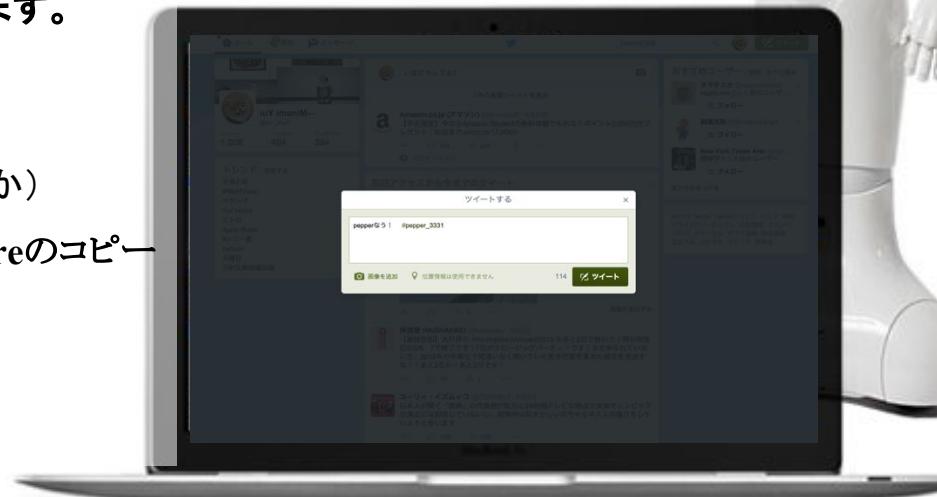
ここスペースは写真OKです。

ぜひハッシュタグ [#pepper_3331](#)をつけて呴いてください

事前にMicrosoft Azure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

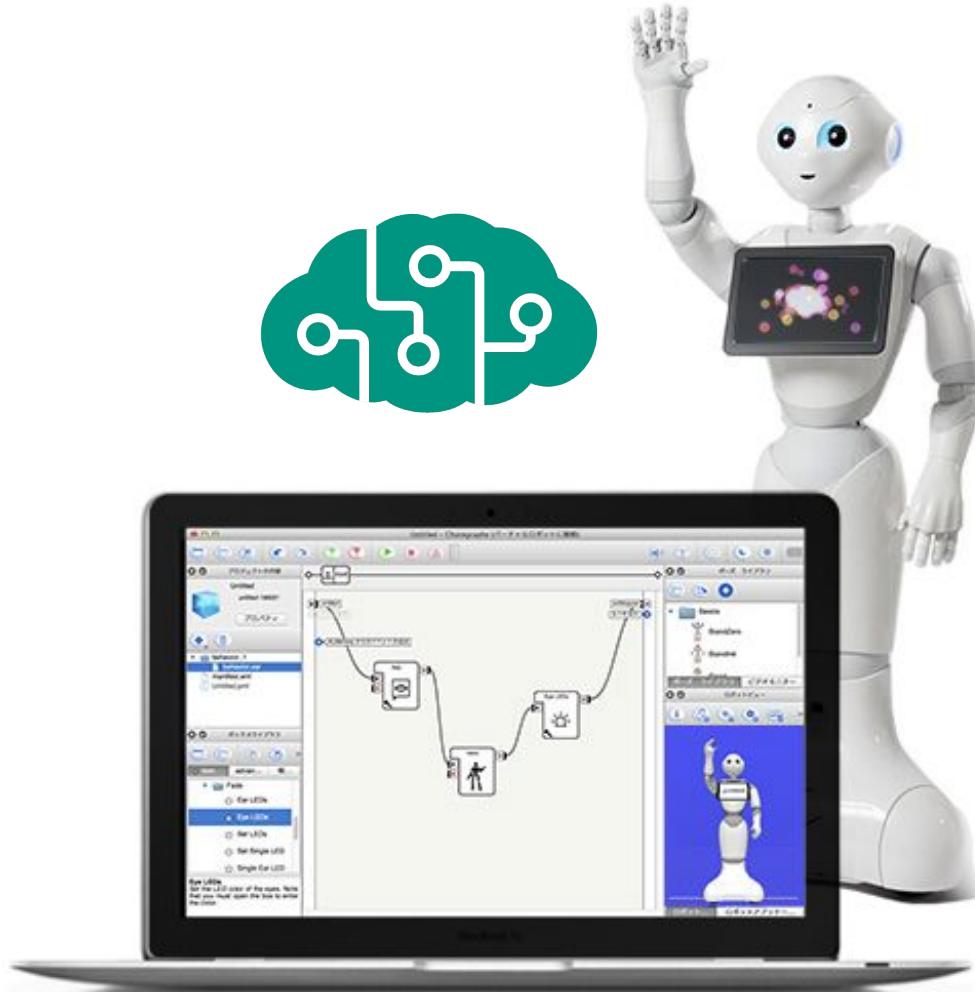
ワークショップ 番外編： Cognitive Service FaceAPI



Microsoft
Azure

2018/5/27

Softbank Robotics Corp. 2017 All rights reserved.



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。

また情報は2018年6月のものです。現在は変更
がある可能性があります。



実体験とコミュニティーで開発を促進する

アトリエ



相互
促進

コミュニティー

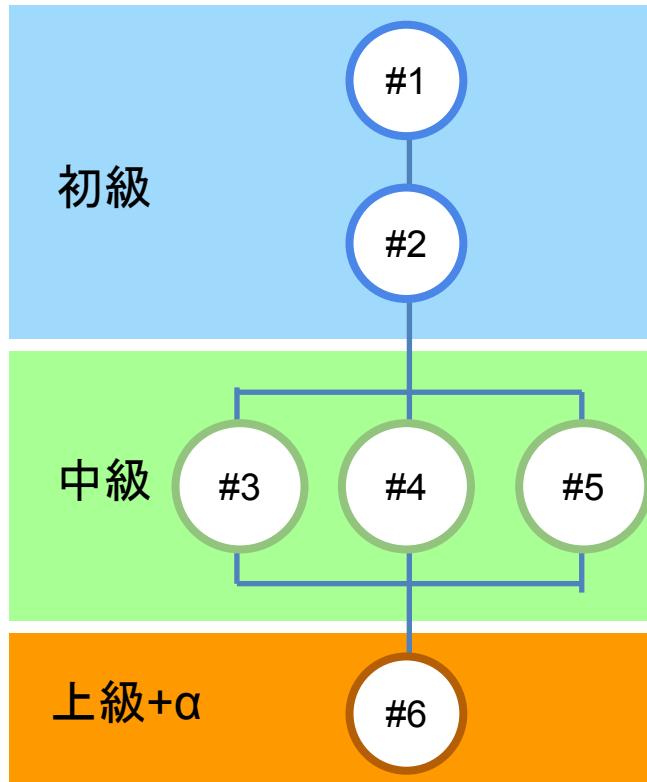


- Pepperのアプリ開発を実体験

- 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

ワークショップ番外編について

アトリエスタッフが製作したオリジナルワークショップ

- ・外部APIとの連携を試そう(天気とTwitter)
- ・Pepperのディレクトリ構造を知ろう
- ・ペッパーリモコンを作ろう
- ・NAOqi2.5.5とNAOqi2.4.3の違い
- ・Pepperで学ぶPython基礎講座その1(変数の扱い方)
- ・Pepperで学ぶPython基礎講座その2(制御文を知る)
- ・Pepperで学ぶPython基礎講座その3(関数を作る)
- ・Pepperで学ぶPython基礎講座その4(BOXを編集)
- ・既存のBOXをPythonで書きかえてみよう(メールとQRコード)
- ・Azure Face APIで顔認証 ハンズオン
- ・Pepperで学ぶ、はじめてのWatson(Visual Recognition編)
- ・Pepper x TensorFlow 入門

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、コミュニティースペース

秋葉原で回答できない質問は
各サテライトへ

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

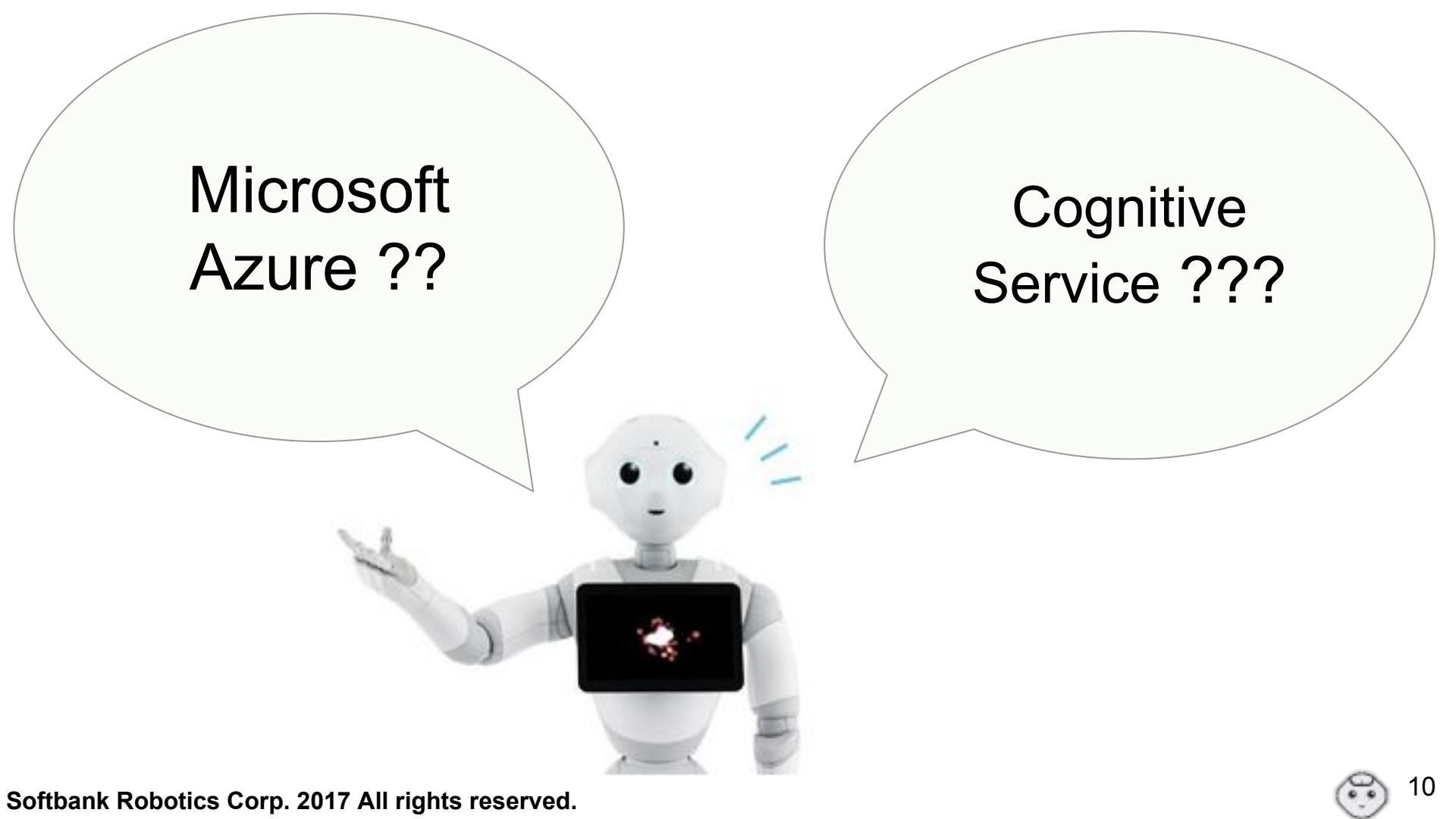
例：

本日の案内を勤めさせていただきます、
○○ ○○(○○○○○)と申します。



1. Microsoft Azure Cognitive Service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？

Microsoft Azureとは、マイクロソフトのクラウド サービスである

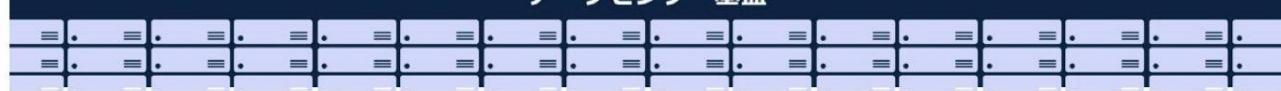
プラットフォームサービス (PaaS)



インフラストラクチャ サービス (IaaS)



データセンター基盤



© 2016 Microsoft Corporation. All rights reserved



Microsoft Azure Cognitive Serviceとは？

- Microsoft Azure内の製品の一つとして提供
- 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APIを呼び出すだけで利用可能
- REST形式のWeb APIの呼び出しが可能
 - C#、Python、AndroidなどはSDK／クライアントライブラリ／サンプル有
- 多くのサービスは一定の無料枠付きの従量課金スタイルになっている
⇒「[Cognitive Servicesの料金に関するページ](#)」



Microsoft Azure Cognitive Serviceとは？

Microsoft & Azure の各製品に対する学習コスト比較

Microsoft & Azure の各製品に対する学習コスト比較		
低		高
Cognitive Services	Azure Machine Learning	Cognitive Toolkit
<ul style="list-style-type: none">■ マイクロソフトで構築した学習モデルを使用■ 学習モデルの構築や学習データの準備は不要 <p>※ 自由度が低い</p>	<ul style="list-style-type: none">■ アルゴリズムをドラッグ & ドロップで作るGUI形式■ 学習モデルのカスタマイズが可能■ 学習データを用意して学習させ、独自の学習モデルの構築も可能	<ul style="list-style-type: none">■ 深層学習のフレームワーク■ Python／C#／C++などでアルゴリズムを構築可能■ 独自に用意した学習データから学習モデルを構築 <p>※ 自由度が高い</p>



Microsoft Azure Cognitive Serviceとは？

Cognitive Serviceの主な機能

視覚-Vision-

Computer Vision API | **Face API** | Content Moderator | Emotion API | Video API | Custom Vision Service | Video Indexer

音声-Speech-

Translator Speech API | Speaker Recognition API | Bing Speech API | Custom Speech Service

言語-Language-

Language Understanding Intelligent Service | Text Analytics API | Bing Spell Check API | Translator Text API | Web Language Model API | Linguistic Analysis API

知識-Knowledge-

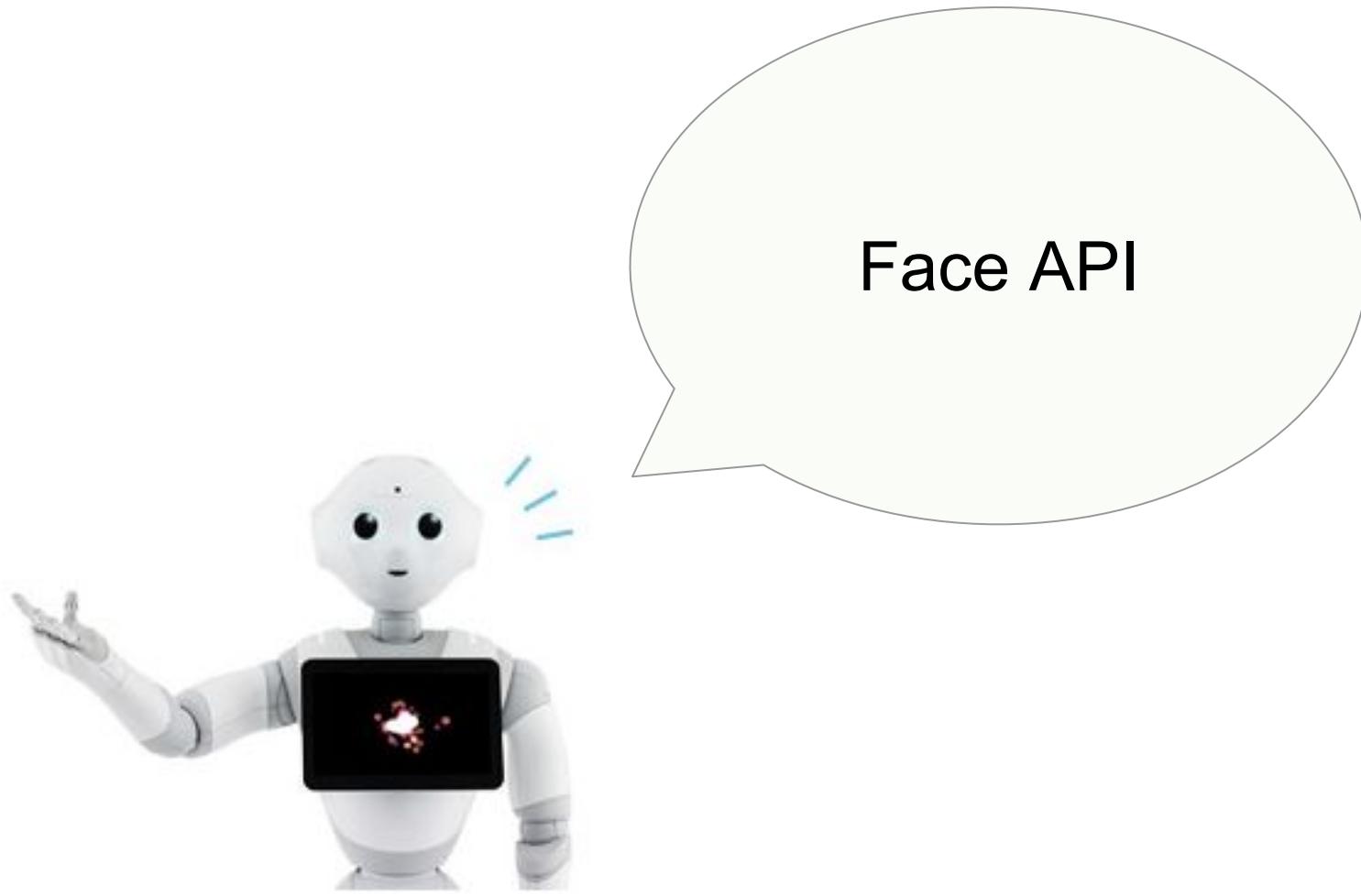
Academic Knowledge API | Knowledge Exploration Service | QnA Maker API | Entity Linking Intelligence Service API | Custom Decision Service | Recommendations API

検索-Search-

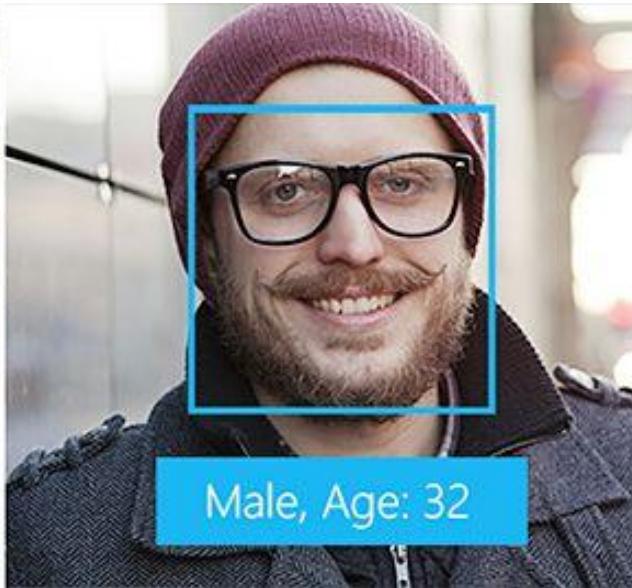
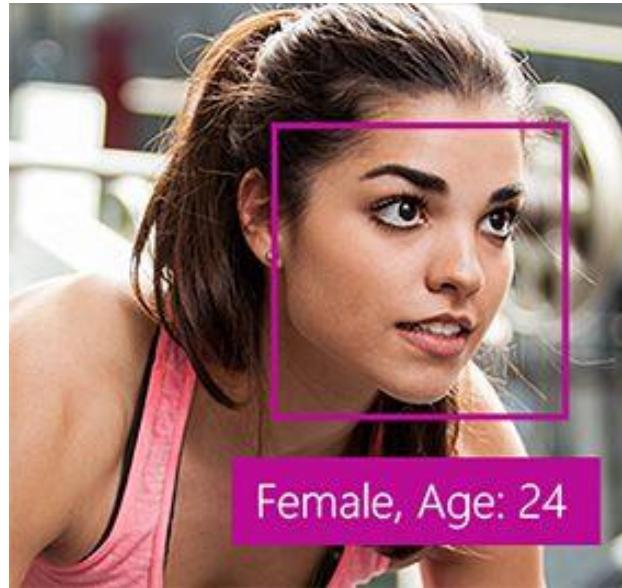
Bing Web Search API | Bing Image Search API | Bing News Search API | Bing Video Search API | Bing Autosuggest API | Bing Custom Search | Bing Entity Search API

詳しくは<https://www.buildinsider.net/small/cognitiveservices/01/>





Face API の主な機能①～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でファイルを指定する
- 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 の主な機能②～顔認識～

主に4つの機能：

[顔の確認] [似た顔の検索] [顔のグループ分け] [人物の識別]
をサポート

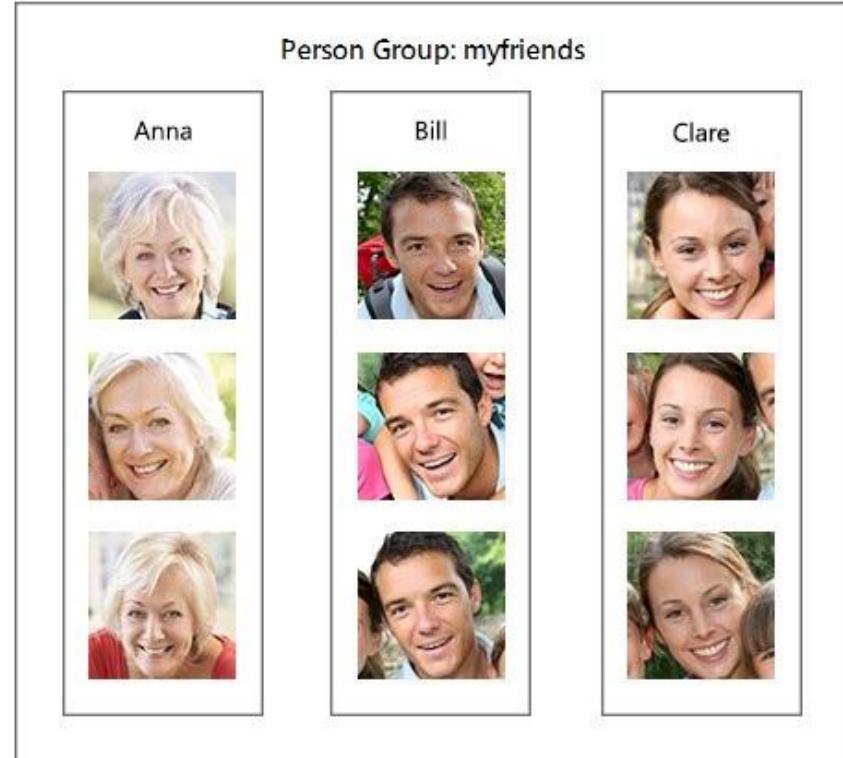
⇒ 今回は[人物の識別]機能を用いていく



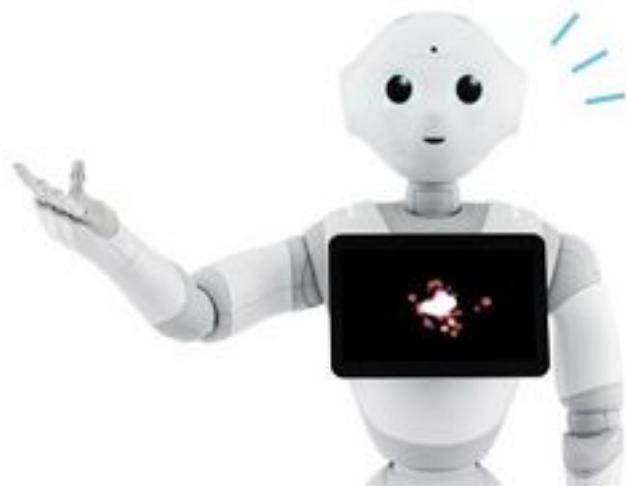
Face API (顔認識)～人物の識別～

Face APIは、検出された顔情報とデータベースの情報に基づいて人を識別するためあらかじめデータベースを作成しておく必要がある

- 右図は、“myfriends”という名前の LargePersonGroup / PersonGroupの例
- 各グループには、最大1,000,000人/10,000人の登録可能
- 各人物オブジェクトは、最大248の顔を登録可能

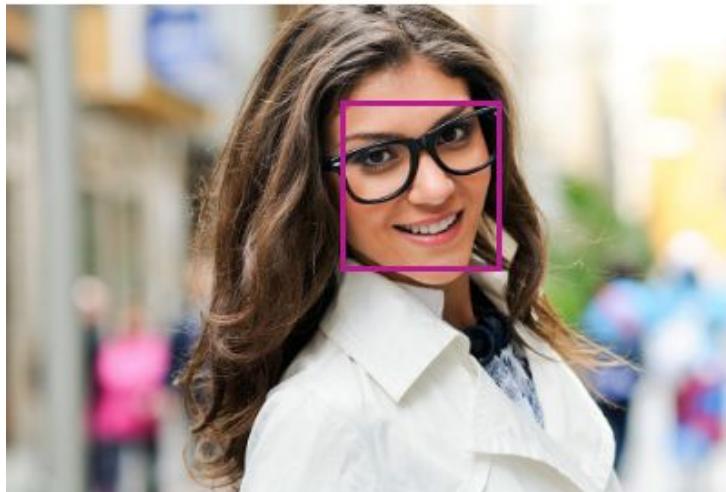


顔を検出
させてみよう



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 ポータルサイト(<https://portal.azure.com/>)

下準備の流れ

1. Azureポータルサイトにアクセス
2. 無料のリソースの作成
3. リソースのデプロイ
4. 作成したリソースのAPIkeyを取得

The screenshot shows the Microsoft Azure portal interface. On the left, there is a sidebar with various service icons and links: 'リソースの作成', 'すべてのサービス', 'お気に入り' (including 'ダッシュボード', 'すべてのリソース', 'リソース グループ', 'App Service', 'Function App', 'SQL データベース', 'Azure Cosmos DB', 'Virtual Machines', 'ロード バランサー', 'ストレージ アカウント', '仮想ネットワーク', 'Azure Active Direct...', 'モニター', 'Advisor', and 'Security Center'). The main content area is titled 'ダッシュボード' and shows a list of resources: 'test_face_detection' (Cognitive Services), 'm1studio' (Machine Learning ...), 'm1studioPlan' (Machine Learning ...), and 'm1studiorstorage' (ストレージ アカウント). Below the resource list, there are sections for 'Azure の導入がより簡単に', 'Windows Virtual Machines', 'Linux Virtual Machines', 'App Service', 'Functions', and 'SQL Database'. At the bottom, there are buttons for 'Service Health' and 'Marketplace'.



顔を検出しよう～下準備②～

The screenshot shows two views of the Microsoft Azure portal. The top view is the 'New' blade, where the 'Resource creation' button is highlighted. The search bar contains the text 'Marketplace を検索'. The bottom view is the 'Marketplace' search results page, where the search term 'Face api' has been entered. The 'Face API' result by Microsoft is highlighted with a yellow box. A large blue arrow points from the search bar in the top view to the search bar in the bottom view.

1. リソースの作成

2. Face APIの検索

3. Face APIの選択



顔を検出しよう～下準備③～

The screenshot shows the Microsoft Azure portal interface. On the left, the main navigation bar includes 'リソースの作成', 'すべてのサービス', 'お気に入り', 'ダッシュボード', 'すべてのリソース', 'リソース グループ', 'App Service', 'Function App', 'SQL データベース', 'Azure Cosmos DB', 'Virtual Machines', 'ロード バランサー', 'ストレージ アカウント', '仮想ネットワーク', 'Azure Active Direct...', and 'モニター'. Below this is a search bar and a 'ホーム > 新規 > Marketplace' breadcrumb trail. The central area displays the 'Face API' service details, including its description and a 'Create' button at the bottom.

The right side shows a detailed 'Create Face API' dialog box. It has fields for 'Name' (with placeholder 'Enter a name'), 'サブスクリプション' (set to '従量課金'), '場所' (set to 'オーストラリア東部'), and '価格レベル' (set to 'F0 (20 1 分あたりの呼び出し回数, 30K 1 ...)'). There are also options for 'Resource group' (radio buttons for '新規作成' and '既存のものを使用') and a '作成' (Create) button at the bottom.

1. リソースの作成
2. 任意の*Nameを付ける
3. *サブスクリプションは無料課金
4. *場所は **オーストラリア東部**
5. *価格レベル **F0** を選択
6. 任意の*Resource group を付ける
7. 作成を選択でデプロイ



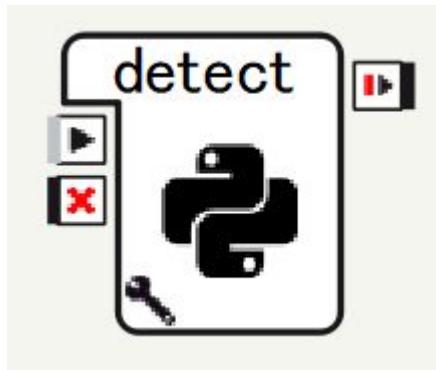
顔を検出しよう～下準備④～

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation bar includes 'リソースの作成', 'すべてのサービス', 'お気に入り', 'ダッシュボード', and a highlighted 'すべてのリソース' option. Below these are 'リソースグループ', 'App Service', 'Function App', and 'SQL データベース'. In the center, the 'test_face_detection - Keys' page for Cognitive Services is displayed. The page has tabs for 'Overview', 'アクティビティログ', 'アクセス制御 (IAM)', 'タグ', and '問題の診断と解決'. Under 'RESOURCE MANAGEMENT', there is a 'Keys' tab which is also highlighted with a yellow box. On the right, there are fields for 'NAME' (set to 'test_face_detection') and two 'KEY' fields, 'KEY 1' and 'KEY 2', each with a blue 'Copy' button. A large blue arrow points from the 'Keys' tab on the left towards the 'KEY 1' and 'KEY 2' fields on the right.

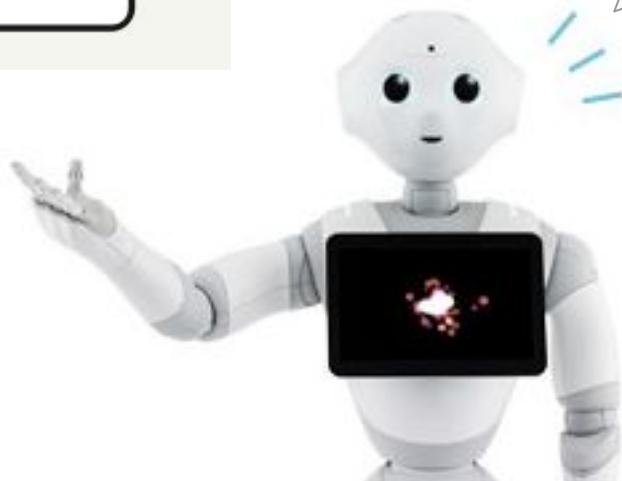
1. すべてのリソース>作成したリソース名>Keys を選択
2. KEY1、KEY2のどちらかをコピー
(青色のボタンでコピーできる)
3. 下準備完了



Choregrapheで実装しよう！

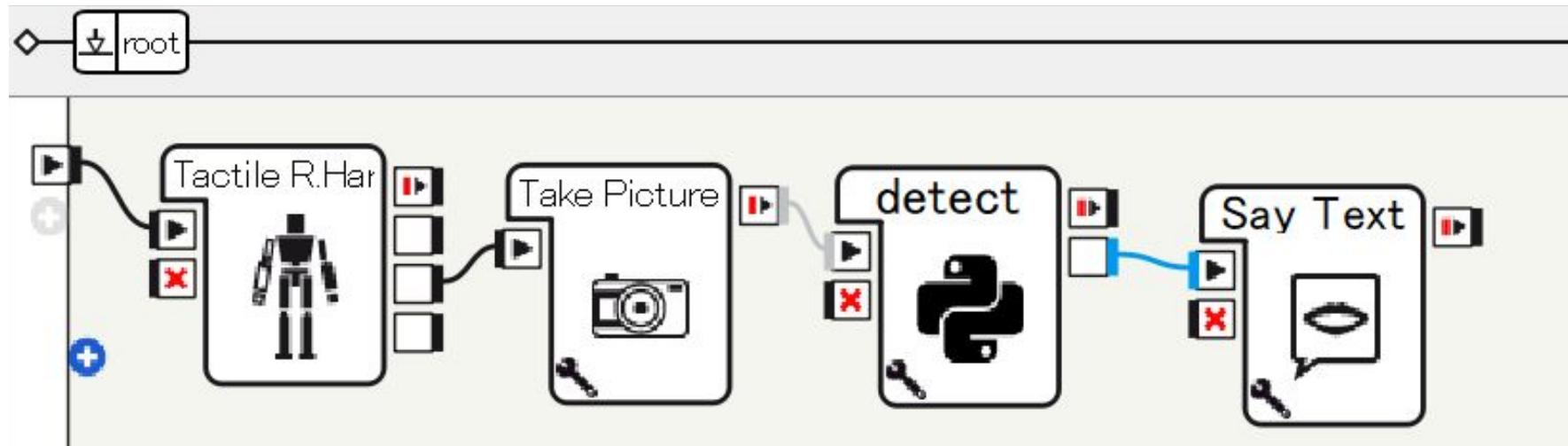


使うボックスはコレ！！



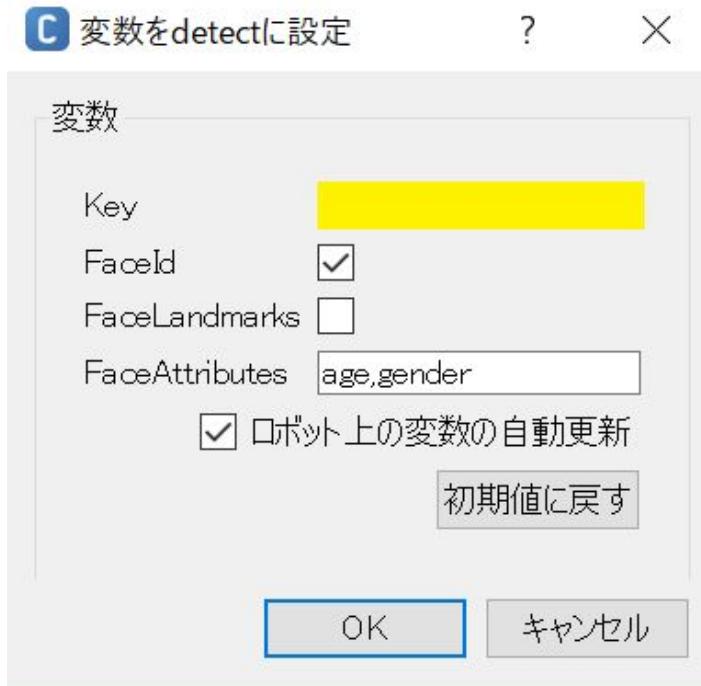
顔を検出しよう

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

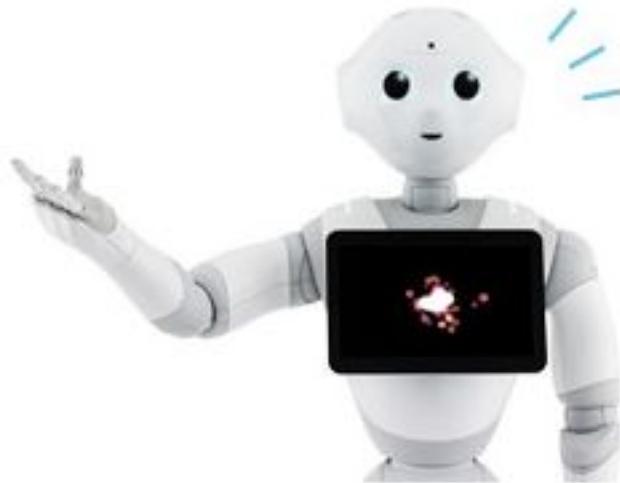
[FacelId]...detected face APIによって返されるID、呼び出されてから24時間後に自動的に消滅

[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

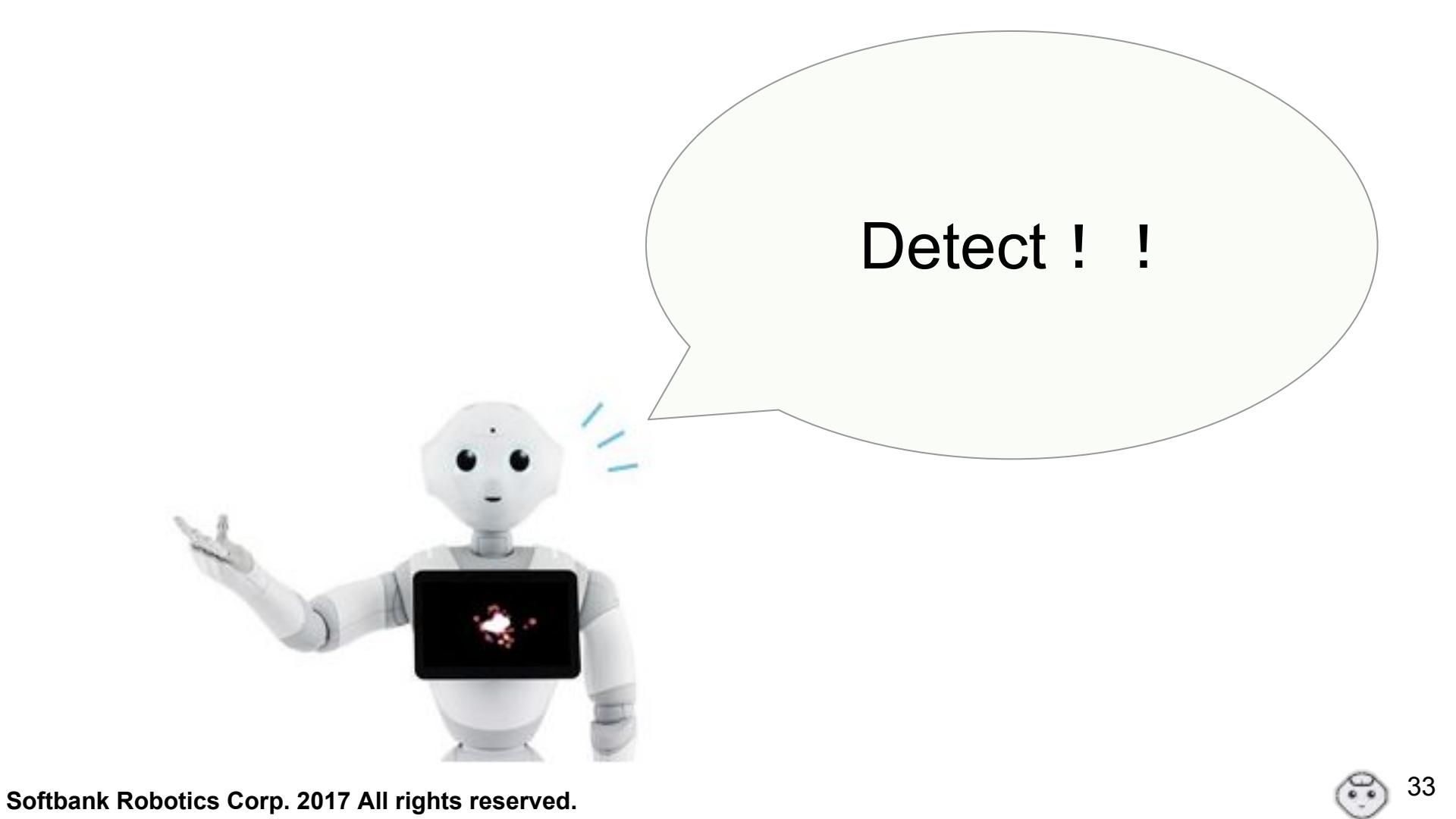
[.connect]...プロキシ動作のトンネル接続への変更

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

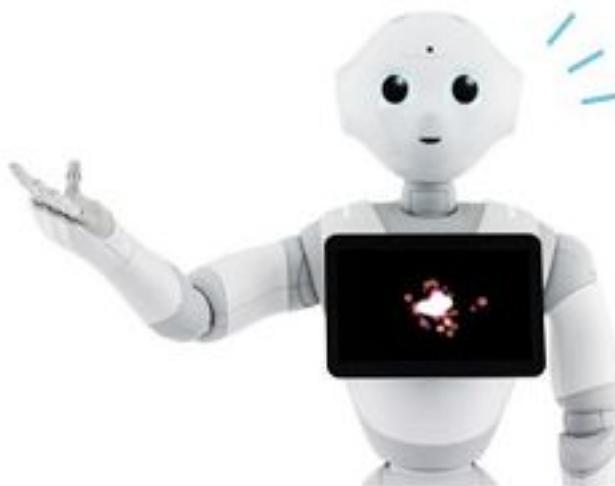


Requests のサンプルコード





Detect ! !



detect Boxを完成させよう！～API Reference①～

DetectのAPI Reference

Http Method
POST

Open API testing console

West US

West US 2

East US

East US 2

West Central US

South Central US

West Europe

North Europe

Southeast Asia

East Asia

Australia East

Brazil South

Canada Central

Central India

UK South

Japan East

Request URL

`https://[location].api.cognitive.microsoft.com/face/v1.0/detect[?returnFaceId][&returnFaceLandmarks][&returnFaceAttributes]`



detect Boxを完成させよう！～API Reference②～

Request parameters

returnFacelid (optional) boolean Return facelids of the detected faces or not. The default value is true.

returnFaceLandmarks (optional) boolean Return face landmarks of the detected faces or not. The default value is false.

returnFaceAttributes (optional) string Analyze and return the one or more specified face attributes in the comma-separated string like "returnFaceAttributes=age,gender". Supported face attributes include age, gender, headPose, smile, facialHair, glasses, emotion, hair, makeup, occlusion, accessories, blur, exposure and noise. Face attribute analysis has additional computational and time cost.

Request headers

Content-Type (optional) string Media type of the body sent to the API.

Ocp-Apim-Subscription-Key string Subscription key which provides access to this API. Found in your [Cognitive Services accounts](#).



detect Boxを完成させよう！～API Reference③～

Request body

To detect in a URL (or binary data) specified image.

JSON fields in the request body:

Fields	Type	Description
url	String	URL of input image.

application/json application/octet-stream

```
{ "url": "http://example.com/1.jpg" }
```



detect Boxを完成させよう！

detectボックスの37行目

```
19  
20曰  
21  
22  
23  
24曰  
25  
26  
27  
28  
29曰  
30  
31  
32  
glasses.  
33  
34  
35  
36曰  
37  
38曰  
39
```

```
def onInput.onStart(self, p): #「Take Picture」ボックスで撮った写真を引数pで受け取っている  
    #アクセスするurl  
    url = 'https://australiaeast.api.cognitive.microsoft.com/face/v1.0/detect'  
    #ヘッダー情報  
    headers = {  
        'Content-Type': 'application/octet-stream',  
        'Ocp-Apim-Subscription-Key': key,  
    }  
    #/パラメータ情報  
    params = {  
        'returnFaceId': self.getParameter("FaceId"), # The default value is true.  
        'returnFaceLandmarks': self.getParameter("FaceLandmarks"), # The default v  
        'returnFaceAttributes': self.getParameter("FaceAttributes"), # age, gender  
    }  
    #パス指定の場合  
    #img_url = '/home/nao/.local/share/PackageManager/apps/.lastUploadedChoreograph  
try:  
    #  
except:  
    self.logger.info(e) # 必要であれば失敗時の処理
```

ここに書く！！！



detect Boxを完成させよう！問題

```
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！！



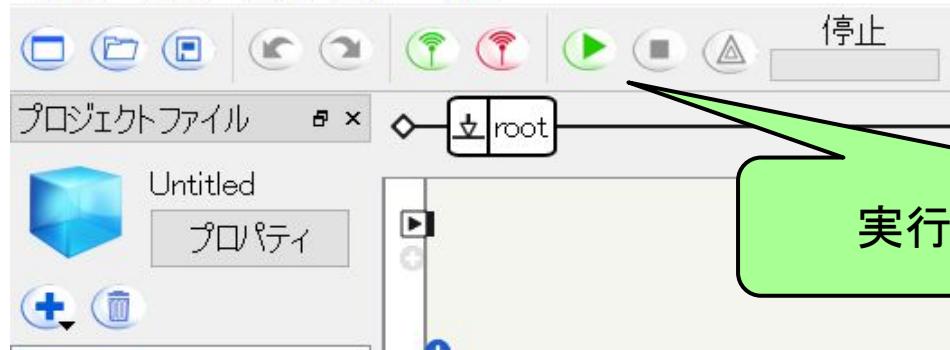
detect Boxを完成させよう！(答え)

```
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
```

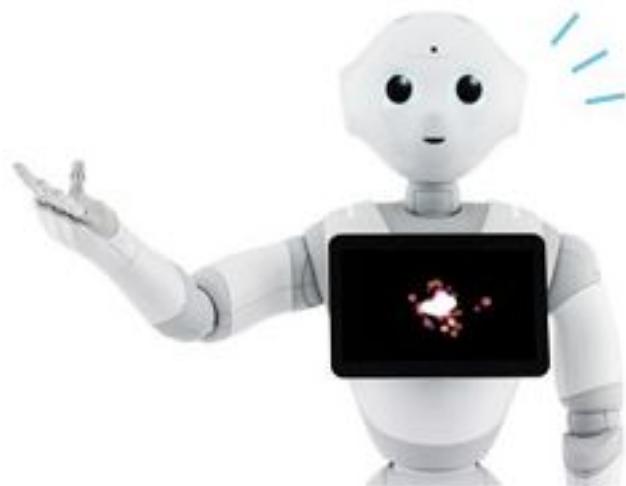
この後扱うボックスにもAPIにアクセスするためのrequests処理を行っている！！

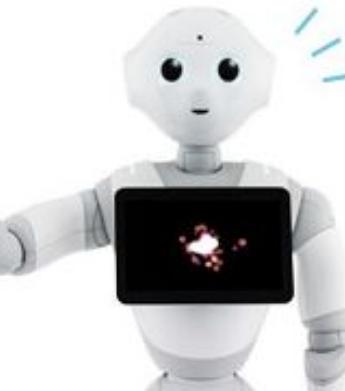
C Untitled - Choregraphe

ファイル 編集 接続 表示 ヘルプ

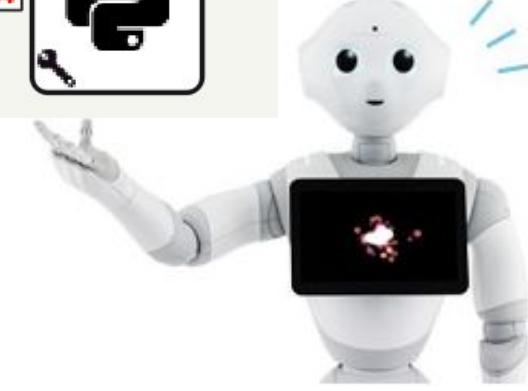
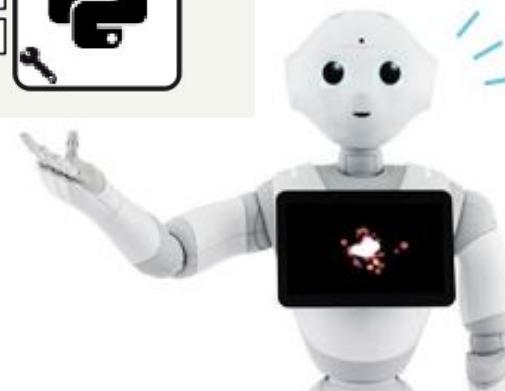
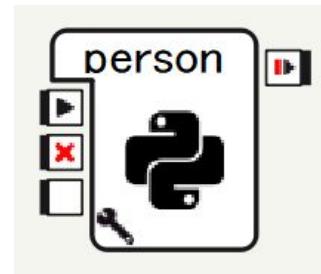
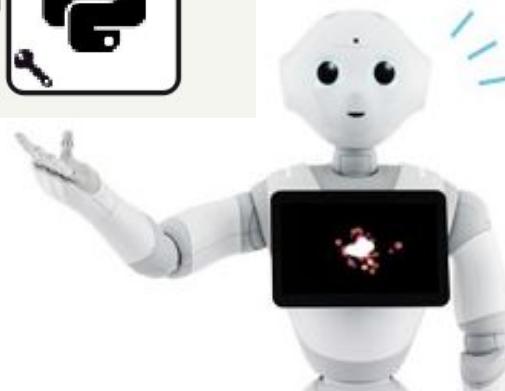
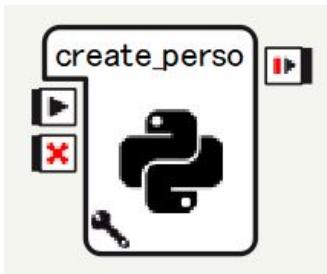


転移学習を行おう！
～下準備編～





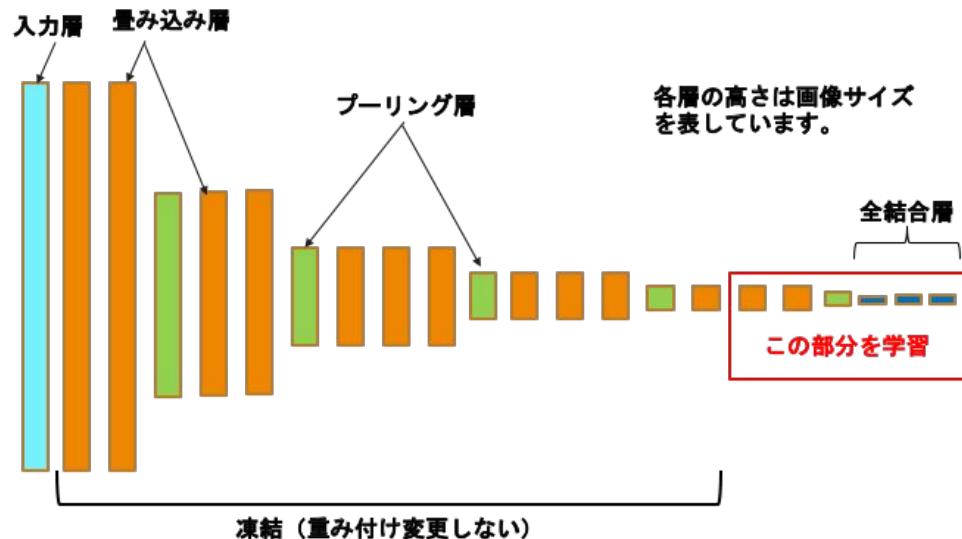
その前にちょっと下準備！
使うボックスはコレ！！



画像を登録して転移学習を行おう

転移学習とは

転移学習とは、学習済みのモデルの出力層以外の重みを転用し、出力層のみを学習して特徴量の抽出を行う手法



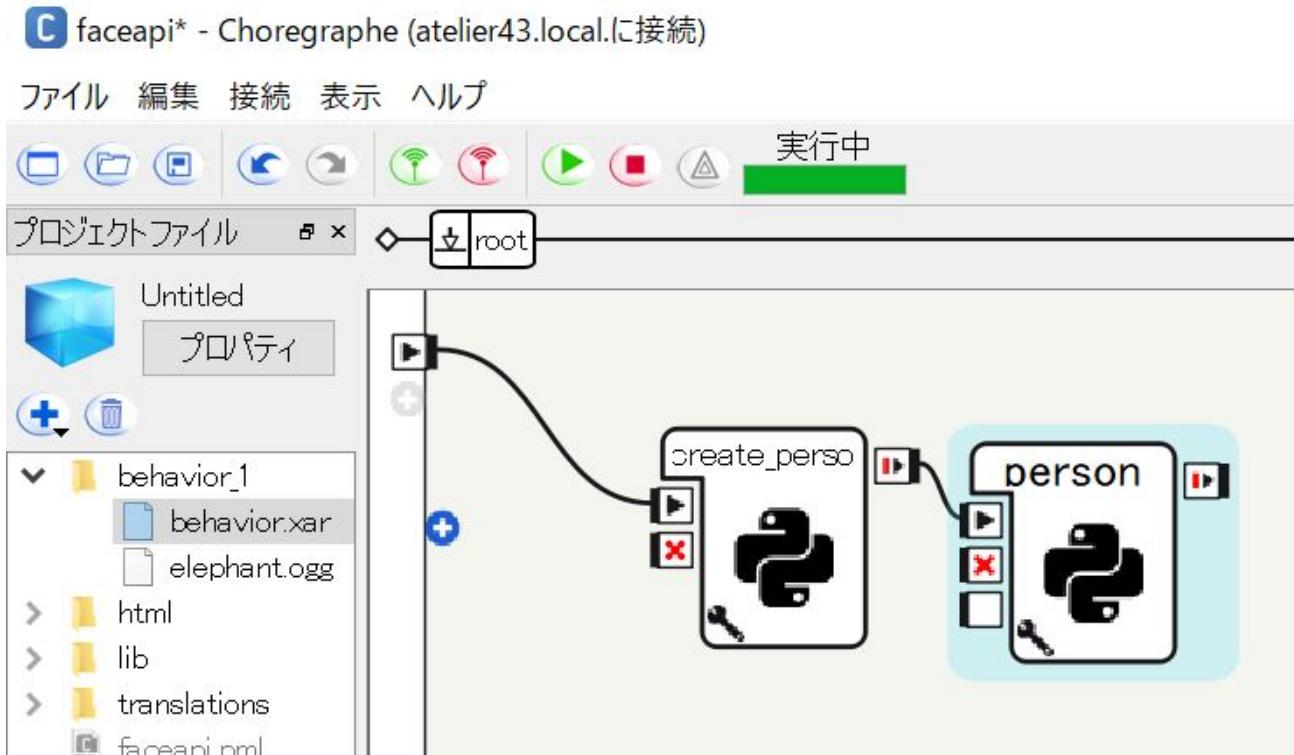
参照:

<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>

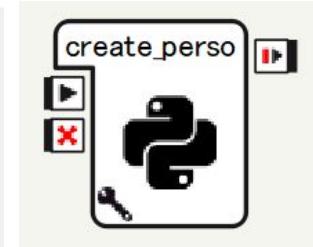
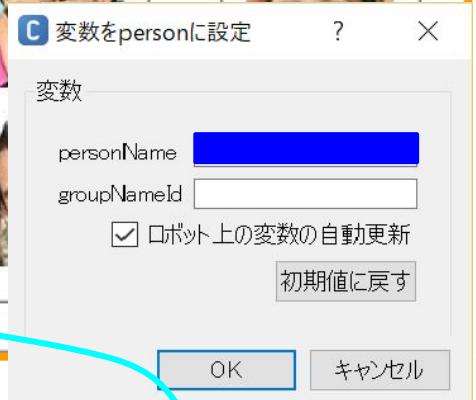
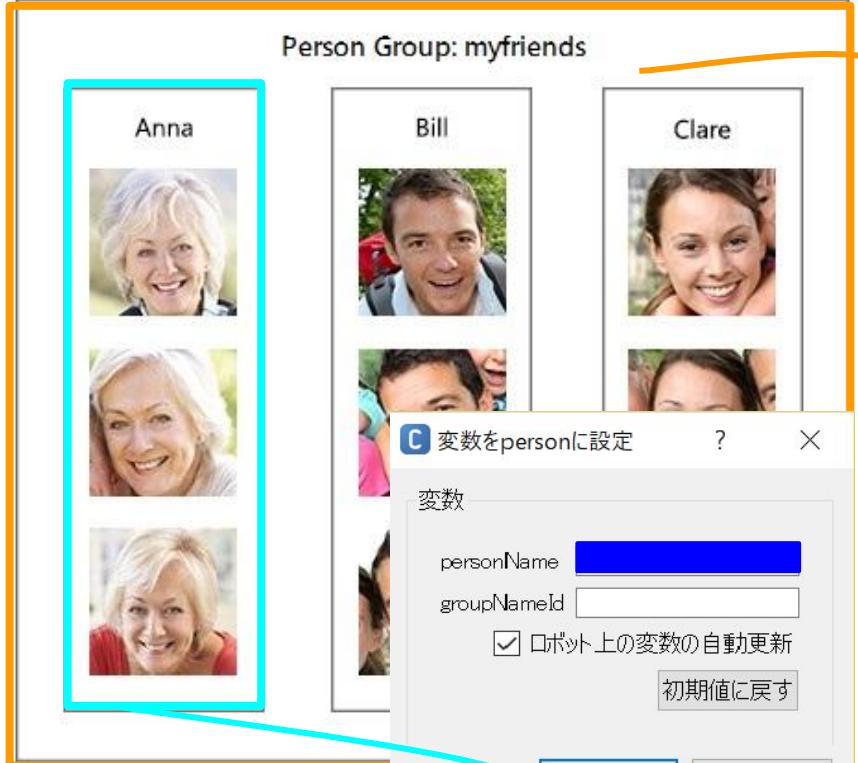


画像を登録して転移学習を行おう

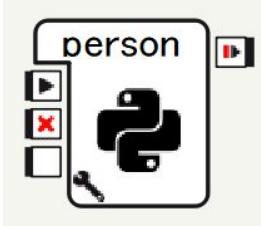
ボックスを以下のようにつないで下さい



画像を登録して転移学習を行おう



groupIdは同じものを指定

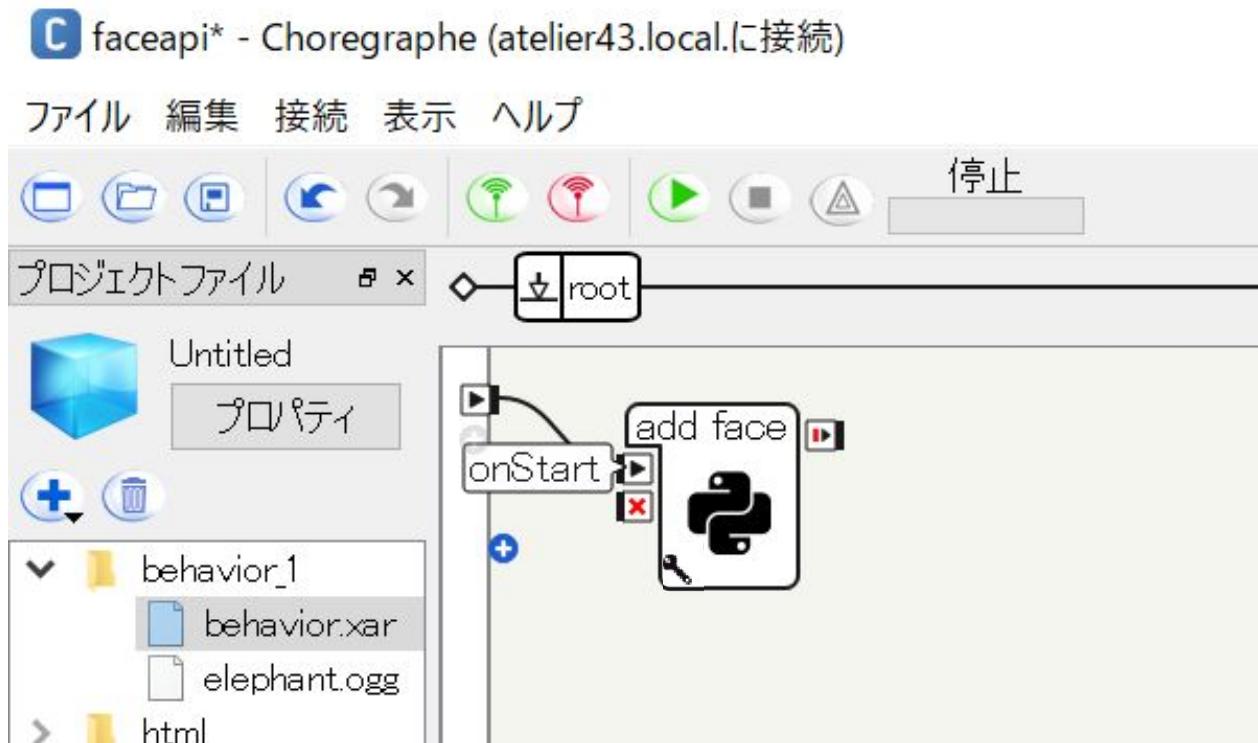


画像を登録して転移学習を行おう



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

The screenshot shows the Microsoft Bot Framework Composer interface. On the left, there's a list of contacts under 'Person Group: myfriends': Anna, Bill, and Clare. Anna's profile picture is highlighted with a pink border. A pink curved arrow points from this highlighted image to the 'add face' block on the right. The 'add face' block has a Python icon and two buttons: a play button and a delete button. To the right of the block is a configuration dialog window titled 'C 変数をadd_face'. It contains fields for 'fileName' and 'Id', both of which are partially obscured by a purple redaction box. There is also a checked checkbox for 'ロボット上の変数の自動更新' (Automatically update variables in the robot) and a '初期値に戻す' (Reset to initial value) button. At the bottom are 'OK' and 'キャンセル' (Cancel) buttons.

- Idには先程ログに返された値を指定
- 追加したい顔写真プロジェクトにインポート
- 拡張子付きでファイル名を指定

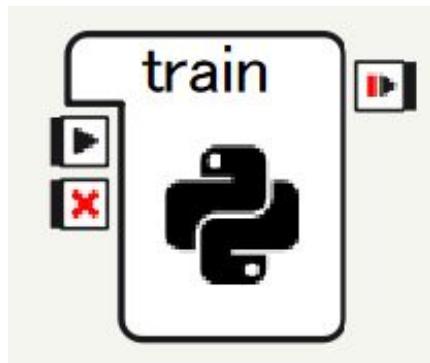


画像を登録して転移学習を行おう



転移学習を行おう！

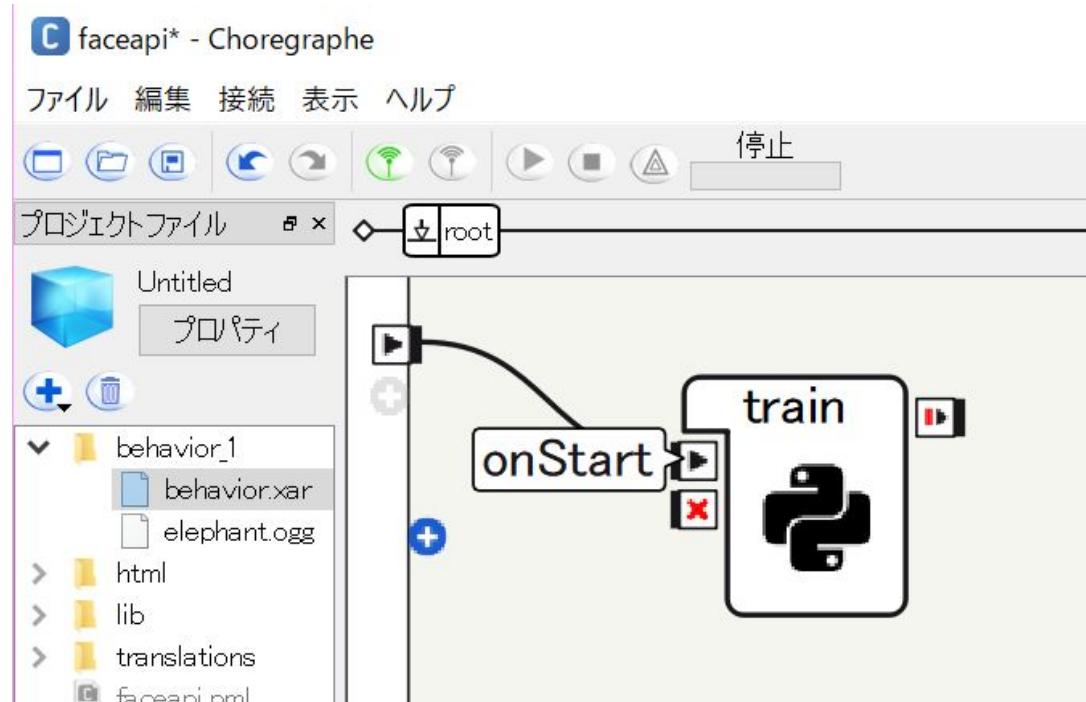




使うボックスはコレ！！

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実行して下さい



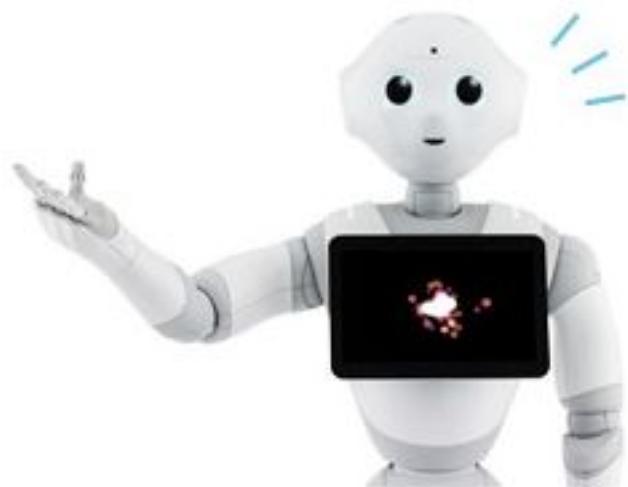
画像を登録して転移学習を行おう

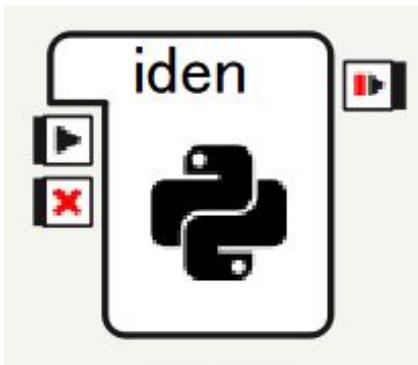
```
14曰  
15  
16曰  
17  
18曰  
19  
20  
21曰  
22  
23  
24曰  
25  
26曰  
27  
28曰  
29曰  
30  
31  
32曰  
33  
34  
35曰  
36  
37曰  
38  
39  
40曰  
41  
42  
43曰  
44  
45曰  
46  
47曰  
48曰  
49  
50曰  
51  
52  
  
def onInput_onStart(self):  
    def train():  
        url = self.url + gnameId + '/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()
```

転移学習を用いているので、
APIへのアクセスのみで学習可能！

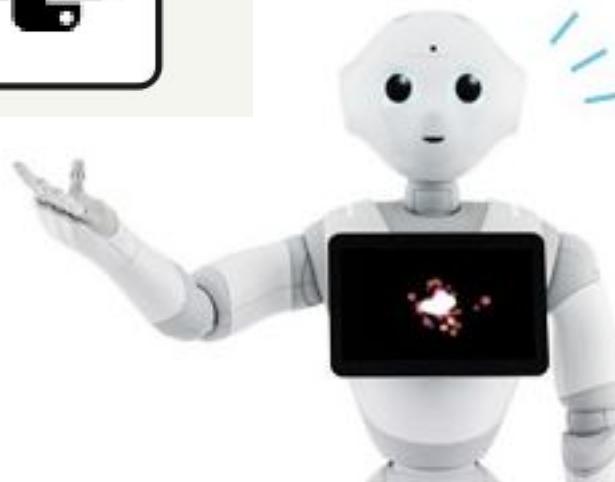


Pepperから
顔を判別しよう



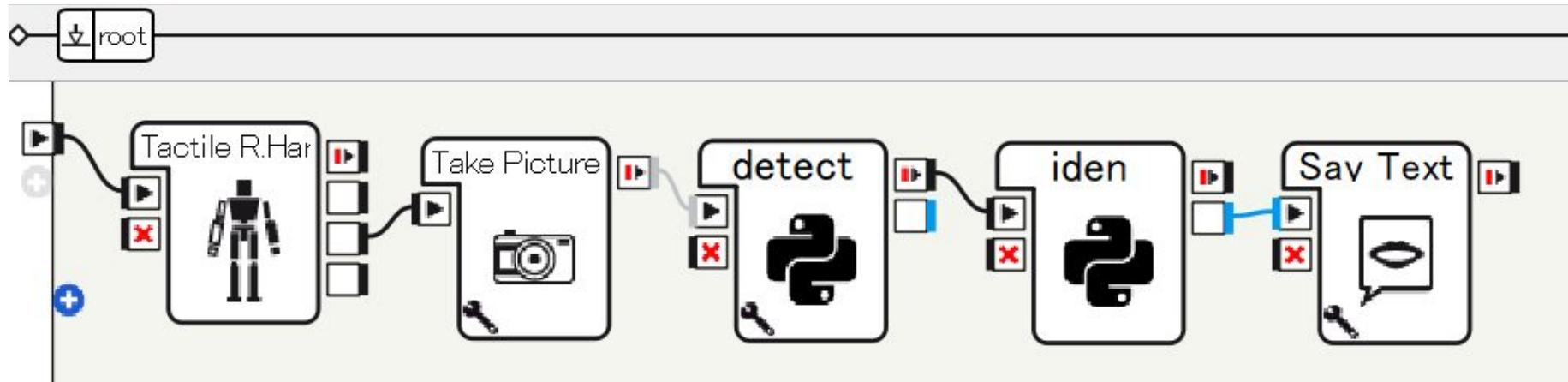


使うボックスはコレ！！



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実行して下さい



iden Boxについて

iden Box 30~31行目

```
27曰
28
29
30
31
32
33曰
34
35曰
    "faceIds": [
        faceId
    ],
    "maxNumOfCandidatesReturned": 10,
    "confidenceThreshold": 0.5
}
try:
    rr = requests.post(url, headers = headers, dat
except Exception as e:
```

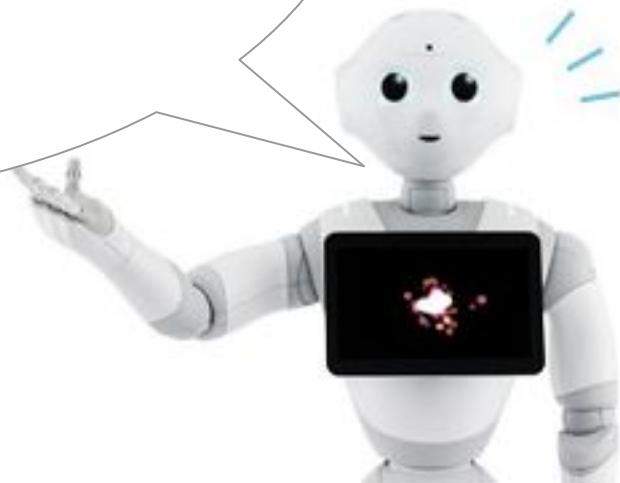
max~: 登録したPersonGroupから最大何人返すか

conf~: 閾値(現在は50%)以上を返す

⇒2つの条件で返す結果に制限をかけられる



**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**



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>

