

実習科目研修 クラウドコンピューティングA

AWS AIサービスAPIを使用したAIシステムの開発

第1回

Translate（翻訳）サービス

Polly（音声合成）サービス

機械学習マネージドサービス

これらのマネージドサービスに ML の経験は不要。

コンピューター
タビジョン

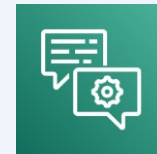


Amazon Rekognition



Amazon Textract

チャットボット

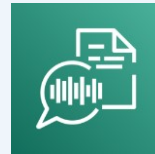


Amazon Lex

音声



Amazon Polly



Amazon Transcribe

予測



Amazon Forecast

言語



Amazon Comprehend



Amazon Translate

レコメンデーション



Amazon Personalize

Amazon Translate



Amazon Translate

- Amazon Translate は高度な機械学習テクノロジーを使用して、**高品質の翻訳をオンデマンドで提供するフルマネージドのテキスト翻訳サービス**です。
- アプリケーションで使う多言語ユーザーエクスペリエンスを開発する
- ドキュメントを複数の言語に翻訳する
- 複数の言語で入力されたテキストを分析する

Amazon Translate のユースケース



国際的なウェブサイト



ソフトウェアのローカライズ



多言語チャットボット



国際的なメディア

テキストを翻訳する (P42～P45)

テキストを日本語から英語に翻訳するプログラム
(trans_text_ja.py) P42

```
# (1) boto3 をインポート
import boto3
# (2) Translate サービスクライアントを作成
translate = boto3.client('translate')
# (3) 日本語の文章
text = 'あとでメールを送ります。'
# (4) 日本語から英語に翻訳
result = translate.translate_text(
    Text=text, SourceLanguageCode='ja',
    TargetLanguageCode='en')
# (5) 翻訳後の文章を表示
print(result['TranslatedText'])
# 翻訳元の言語コードを表示
print('翻訳元の言語コード = {}'.format(result['SourceLanguageCode']))
# 翻訳先の言語コードを表示
print('翻訳先の言語コード = {}'.format(result['TargetLanguageCode']))
```

Boto3 boto3.client関数

機能: 指定した名前のサービスクライアントを作成する
使い方: boto3.client(クライアント名)
戻り値: サービスクライアントのオブジェクト

Translate translate_textメソッド

機能: 指定した文字列を翻訳する
使い方: translate_text(
 Text=翻訳する文字列,
 SourceLanguageCode=翻訳元の言語,
 TargetLanguageCode=翻訳先の言語)
戻り値: 翻訳先のオブジェクト (Pythonの辞書)

translate_textメソッドの対応言語（P48～P52）

言語名	言語コード
アフリカーンス語	af
アルバニア語	sq
アムハラ語	am
アラビア語	ar
アルメニア語	hy
アゼルバイジャン語	az
ベンガル語	bn
ボスニア語	bs
ブルガリア語	bg
カタルーニャ語	ca
中国語（簡体字）	zh
中国語（繁体字）	zh-TW
クロアチア語	hr
チェコ語	cs
デンマーク語	da
ダリ語	fa-AF
オランダ語	nl
英語	en

エストニア語	et
ペルシャ語(Persian)	fa
フィリピン語、タガログ語	tl
フィンランド語	fi
フランス語	fr
フランス語（カナダ）	fr-CA
グルジア語	ka
ドイツ語	de
ギリシャ語	el
グジャラート語	gu
ハイチ・クレオール語	ht
ハウサ語	ha
ヘブライ語	he
ヒンディー語	hi
ハンガリー語	hu
アイスランド語	is
インドネシア語	id
アイルランド語	ga
イタリア語	it

日本語	ja
カナダ語	kn
カザフ語	kk
韓国語	ko
ラトビア語	lv
リトアニア語	lt
マケドニア語	mk
マレー語	ms
マラヤーラム語	ml
マルタ語	mt
マラーティー語	mr
モンゴル語	mn
ノルウェー語	no
パシュトゥー語	ps
ポーランド語	pl
ポルトガル語	pt
ポルトガル語（Portugal）	pt-PT
パンジャブ語	pa
ルーマニア語	ro

ロシア語	ru
セルビア語	sr
シンハラ語	si
スロバキア語	sk
スロベニア語	sl
ソマリア語	so
スペイン語	es
スペイン語（メキシコ）	es-MX
スワヒリ語	sw
スウェーデン語	sv
タミール語	ta
テルグ語	te
タイ語	th
トルコ語	tr
ウクライナ語	uk
ウルドゥー語	ur
ウズベク語	uz
ベトナム語	vi
ウェールズ語	cy

translate_textメソッドの詳細（P48～P52）

translate_textメソッドの引数

引数名	型	必須	機能
Text	文字列	○	翻訳する文字列。最大5000バイト。使用する文字の種類に応じて、何文字に相当するかは変化する
SourceLanguageCode	文字列	○	翻訳元の言語コード。'auto'（自動）を指定した場合、AIサービスのComprehendを呼び出して、翻訳元の言語を自動判定する。
TargetLanguageCode	文字列	○	翻訳先の言語コード
TerminologyNames	文字列のリスト	×	用語の登録名

translate_textメソッドの戻り値

キー	型	値の内容
TranslatedText	文字列	翻訳後の文章
SourceLanguageCode	文字列	翻訳元の言語コード
TargetLanguageCode	文字列	翻訳先の言語コード
AppliedTerminologies	辞書	翻訳に適用された用語
ResponseMetadata	辞書	応答に関するメタデータ

```
{'ResponseMetadata': {'HTTPHeaders': {'cache-control': 'no-cache',
                                         'content-length': '102',
                                         'content-type': 'application/x-amz-json-1.1',
                                         'date': 'Thu, 07 Apr 2022 03:10:14 GMT',
                                         'x-amzn-requestid':
                                           '2061c768-eb84-400f-be30-b34155b63182'},
                      'HTTPStatusCode': 200,
                      'RequestId': '2061c768-eb84-400f-be30-b34155b63182',
                      'RetryAttempts': 0},
 'SourceLanguageCode': 'ja',
 'TargetLanguageCode': 'en',
 'TranslatedText': "I'll send you an email later."}
```

用語を登録して翻訳に使う (P53～P60)

用語を登録しないで翻訳するプログラム
(trans_term_disabled.py) P54

```
# boto3 をインポート
import boto3
# Translate サービスクライアントを作成
translate = boto3.client('translate')
# 日本語の文章
text = 'ひぐぺん工房の新しい本が秀和システムから出ます。'
# 日本語から英語に翻訳
result = translate.translate_text(
    Text=text, SourceLanguageCode='ja',
    TargetLanguageCode='en')
# 翻訳されたテキストを表示
print(result['TranslatedText'])
```

```
$> python trans_term_disabled.py
A new book by Higupen Kobo will be released from the Shuwa System.
$>
```


用語のCSVファイル (P54)

1 行目 翻訳元の言語コード, 翻訳先の言語コード

2 行目以降 翻訳元の使用語, 翻訳先の使用語

用語のCSVファイル (term_ja.csv) P55

ja, en

ぴぐぺん工房, HigPen Works

秀和システム, "SUWA SYSTEM CO.,LTD"

用語を登録する (P55)

用語を登録するプログラム (trans_term_import.py) P55

```
# boto3 をインポート
import boto3
# Translate サービスクライアントを作成
translate = boto3.client('translate')
# CSVファイルを開く
with open('term_ja.csv', 'rb') as file:
    # 用語を登録
    translate.import_terminology(
        Name='term_ja',
        MergeStrategy='OVERWRITE',
        TerminologyData={'File': file.read(), 'Format': 'CSV'})
```

Translate import_terminologyメソッド

機能: 用語を登録する

使い方: import_terminology(

 Name=登録名,

 MergeStrategy='OVERWRITE',

 TerminologyData={'FILE': ファイルの内容, 'Format': 'CSV'})

戻り値: 登録結果を含む辞書

用語を翻訳に使う (P59)

用語を使って翻訳するプログラム (trans_term_enabled.py) P59

```
# boto3 をインポート
import boto3
# Translate サービスクライアントを作成
translate = boto3.client('translate')
# 日本語の文章
text = 'ひぐぺん工房の新しい本が秀和システムから出ます。'
# 日本語から英語に翻訳 (登録した用語を使用)
result = translate.translate_text(
    Text=text, SourceLanguageCode='ja',
    TargetLanguageCode='en',
    TerminologyNames=['term_ja'])
# 翻訳された文章を表示
print(result['TranslatedText'])
```

```
$> python trans_term_enabled.py
A new book by HigPen Works will be released from the SHUWA SYSTEM CO., LTD.

$>
```


TerminologyPropertiesListに含まれる情報 (P62)

キー	型	値の内容
Arn	文字列	ARN (Amazon Resource Name) 。AWSがリソースを識別するための名前
CreatedAt	日時	用語が最初に登録された日時
LastUpdatedAt	日時	用語が最後に更新された日時
Name	文字列	用語の登録名
SizeBytes	整数	バイト数
SourceLanguageCode	文字列	翻訳元の言語コード
TargetLanguageCodes	文字列のリスト	翻訳先の言語コード、複数の場合もある
TermCount	整数	含まれる言葉の個数

Translate

list_terminologiesメソッド

機能: 登録名の一覧を取得する

使い方: list_terminologies()

戻り値: 登録名の一覧を含む辞書

用語データを取得する (P63)

用語データを取得するプログラム (trans_term_get.py) P62

```
# boto3 をインポート
import boto3

# pprint をインポート
import pprint

# Translate サービスクライアントを作成
translate = boto3.client('translate')

# 用語データを取得
result = translate.get_terminology(
    Name='term_ja', TerminologyDataFormat='CSV')

# 結果を整形して表示
pprint.pprint(result)
```

Translate get terminologiesメソッド

機能：用語データを取得する

使い方: `get_terminologies(Name=登録名, TerminologyDataFormat=形式)`

戻り値: 用語データの場所を含む辞書

```
$> python trans_term_get.py  
[ResponseMetadata: {'HTTPHeaders': {'cache-control': 'no-cache',  
    'content-length': '1928',  
    'content-type': 'application/x-amz-json-1.1',  
    'date': 'Fri, 08 Apr 2022 04:21:54 GMT',  
    'x-amzn-requestid': 'c112c9f6-75dc-4f54-b00d-1288ff694fc3'},  
    'HTTPStatusCode': 200,  
    'RequestId': 'c112c9f6-75dc-4f54-b00d-1288ff694fc3',  
    'RetryAttempts': 0},  
    'TerminologyDataLocation': {'Location': 'https://aws-translate-terminology-prod-us-east-1.s3.us-east-1.amazonaws.com/0854864937/term_ja/LATEST/71c7e08f-1dfc-48a9-916d-f717838b7af5/CSV/?X-Amz-Security-Token=IQoJb3JpZ2luZXVzLWVhc3QtMSJHMEUCIFVx%2Fdkdd5v5mUEYqabjInuiGZC2WdWm2PFBPB1UADWoAiEATPF2Va9AZ%2FIshIxa41jbJy2FfsVsSKdu6PFXTwzlglQlzpP%2F%2F%2F%2F%2F%2F%2F%2F%2FARADGGwyODkOMjI3NjAwnZUiDGigmh%2BvTYNrqsMu1yrXAQQHHw54k4yoac5dl3wgNprO6GVpgWvr25l7daM5jzLKDA4lu7rSsf%2FHwbzb7BBKki6%2BCzcUhggVuCCOUTD0llwiaTA%2BFUNGS7EOAcVLldId80Q%2Fk7xSkZcfzVZOppoivijIX8x7NxrrVVv%2BNk%2BgXT5t%2BKehYaTB8TbIBKztaki6%2BM48gQyQNokNV4L%2FGIUjqoraaPwhBYsuNIIMbyfSiVSWS239YoIxPfciAYQTJK4W3fhgfjRX8YPnRpSt6lgYGzuYSFo3OY2l08imQPmmueDpydON5raIfNf8LeTM8LUWHJ8QCnb8i%2FOna18omSigYlc2N%2F15bkzOPCIbj7L9YOncUpGIlgE302BM8sj2R%2B%2BZ2RJURzcouhwgjiQGufEyX3Z5heSqMUud%2F40UmemeETSVtc6D4kStrzb09RJLUZMMACNWTPFP3Cs5umKx3R3tniHHgvRM7JqdKaAljZVsiUIiwML7ChXIITNBPGPKzkWWARqv2FWsc8OT63kt35UZ3%2FM1NIC9kd5d9r3YYPr3ECD8zhlat%2FqnghCh1cmgy6v5fh8KLt%2Bpb2BMhgHOg%2FzuIQtoqrF%2FjnqpWVwwDDHyrf6SBjlAXKhQGDPmqpz4Xmt639ZWp8miVCBIUnnsUXRRORnlQLWIImPU%2BPCEujZDNF3MoHDIO2nWhaYQXY89t9bn0ys3RrXM4xy6vh16VOUrut6FeLh9KO1NrcHYAN6xoQ6TeW%2BBXi2nbLnPVpuGPSPrhPYH7%2BHrkNGkvGRpmjo2MF83JOPTcgajgpFlE9zTQzTx5khAgUpeJ5kLNYH08Q%3D%3DX-Amz-Algorith=AWS4-HMAC-SHA256&X-Amz-Date=20220408T042154Z&X-Amz-Signedhost&X-Amz-Expires=1800&X-Amz-Credential=ASIAUGYHQ3CFZ3TP4JV6%2F20220408%2Fus-east-1%2Fs3%2FAws4_request&X-Amz-Sig=913945d93943fa2fb8b63fecce1e45ebd1fa4240d0dff649019025e46e9aac2',  
    'RepositoryType': 'S3'},  
    'TerminologyProperties': {'Arn': 'arn:aws:translate:us-east-1:440854864937:terminology/term_ja/LATEST',  
    'CreatedAt': datetime.datetime(2022, 4, 7, 12, 57, 47, 766000, tzinfo=tzlocal()),  
    'Format': 'CSV',  
    'LastUpdatedAt': datetime.datetime(2022, 4, 7, 12, 57, 47, 913000, tzinfo=tzlocal()),  
    'Name': 'term_ja',  
    'SizeBytes': 81,  
    'SourceLanguageCode': 'ja',  
    'TargetLanguageCodes': ['en'],  
    'TermCount': 2}]
```

用語データをダウンロードする (P65)

用語データをダウンロードするプログラム (trans_term_download.py) P65

```
# boto3 をインポート
import boto3
# urllib をインポート
import urllib
# Translate サービスクライアントを作成
translate = boto3.client('translate')
# 用語データを取得
result = translate.get_terminology(
    Name='term_ja', TerminologyDataFormat='CSV')
# 用語データのURLを取得
url = result['TerminologyDataLocation']['Location']
# 用語データのURLを表示
print(url)
# 書き込み用のファイルを開く
with open('term_ja_download.csv', 'wb') as file_out:
    # 用語データのURLを開く
    with urllib.request.urlopen(url) as file_in:
        # 用語データをダウンロードしてファイルに書き込む
        file_out.write(file_in.read())
```

```
"ja","en"
"秀和システム","SHUWA SYSTEM CO.,LTD"
"ピグペン工房","HigPen Works"
```

用語を削除する (P66)

用語を削除するプログラム (trans_term_delete.py) P68

```
# boto3 をインポート
import boto3
# Translate サービスクライアントを作成
translate = boto3.client('translate')
# 用語を削除
translate.delete_terminology(Name='term_ja')
```

Translate delete_terminologyメソッド

機能: 用語データを削除する

使い方: delete_terminology(Name=登録名)

戻り値: なし

テキストファイルを翻訳する (P69～P73)

テキストファイルを翻訳プログラム (trans_file.py) P71

```
# boto3 をインポート
import boto3
# Translate サービスクライアントを作成
translate = boto3.client('translate')
# 入力ファイルを開く
with open('trans_file_in.txt', 'r', encoding='utf-8') as file_in:
    # 出力ファイルを開く
    with open('trans_file_out.txt', 'w', encoding='utf-8') as file_out:
        # 入力ファイルを1行ずつ読み込む
        for text in file_in:
            # 空行でなければ翻訳する
            if text != '\n':
                result = translate.translate_text(
                    Text=text,
                    SourceLanguageCode='ja',
                    TargetLanguageCode='en',
                    TerminologyNames=['term_ja'])
            # 翻訳されたテキストをファイルに書き込む
            file_out.write(result['TranslatedText'])
        # 改行をファイルに書き込む
        file_out.write('\n')
```

翻訳するテキストファイル (trans_file_in.txt)

私どものソフトウェアをお使い頂き、誠にありがとうございます。
頂いたご質問の件、おそらくPythonのバージョンが異なることが原因と思われます。

お使いの環境はWindowsでしょうか、あるいはmacOSやLinuxでしょうか。

Windowsの場合は「python」コマンドを、macOSやLinuxの場合は「python3」コマンドをお使いください。
macOSやLinuxで「python」コマンドを使った場合、Python 3ではなくPython 2が起動するため、ソフトウェアが想定通りに動作しない場合があります。

上記の対策で、問題が解決することを願っています。
今後とも私どものソフトウェアをご活用頂けますよう、どうぞ宜しくお願い申し上げます。

ひぐぺん工房

翻訳されたテキスト (trans_file_out.txt)

Thank you very much for using our software.

The question you received is probably due to a different version of Python.

Is your environment Windows, macOS or Linux?

For Windows, use the “python” command, and for macOS and Linux, use the “python3” command.

If you use the “python” command on macOS or Linux, the software may not work as expected because Python 2 is started instead of Python 3.

I hope the above measures will resolve the issue.

Thank you so that you can continue to use our software in the future.

HigPen Works

CSVファイルを翻訳する (P74~P77)

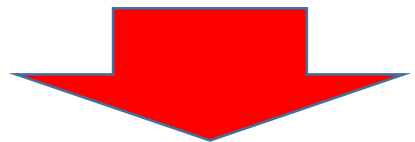
CSVファイルを翻訳するプログラム (trans_csv.py) P75

```
# boto3 インポート
import boto3
# csv をインポート
import csv
# リージョンを指定して Translate サービスクライアントを作成
translate = boto3.client('translate', 'us-east-2')
# 入力ファイルを開く
with open('trans_csv_in.csv', 'r', encoding='utf-8') as file_in:
    # 出力ファイルを開く
    with open('trans_csv_out.csv', 'w', encoding='utf-8',
              newline='') as file_out:
        # csv を出力する準備
        writer = csv.writer(file_out)
        # 入力するファイルを1行ずつ読み込む
        for row in csv.reader(file_in):
            print(row)
            # レビューの本文を翻訳する
            result = translate.translate_text(
                Text=row[2],
                SourceLanguageCode='auto',
                TargetLanguageCode='ja')
            # レビューの本文を翻訳結果に書き換える
            row[2] = result['TranslatedText']
            # 出力ファイルに書き込む
            writer.writerow(row)
```

翻訳するCSVファイルと翻訳されたCSVファイル

翻訳するCSVファイル (trans_csv_in.csv)

```
20200105,"Bear","It was good that shipment was quick."  
20200216,"Bär","Ich möchte, dass die Versandkosten günstiger sind."  
20200327,"Ours","Je veux plus de variations de couleurs."  
20200408,"クマ","また注文したいです。"
```



translate.translate_text() メソッド

翻訳したCSVファイル (trans_csv_out.txt)

```
20200105,Bear,出荷が早かったのは良かった。  
20200216,Bär,送料は安くしてほしい  
20200327,Ours,カラーバリエーションはもっと欲しい  
20200408,クマ,また注文したいです。
```

JSONファイルを翻訳する (P78～P81)

JSONファイルを翻訳するプログラム (trans_json.py) P80

```
# boto3 をインポート
import boto3
# json をインポート
import json
import pprint
# リージョンを指定して Translate サービスクライアントを作成
translate = boto3.client('translate', 'us-east-2')
# 入力ファイルを開く
with open('trans_json_in.json', 'r', encoding='utf-8') as file_in:
    # JSON ファイルを読み込む
    reviews = json.load(file_in)
    pprint.pprint(reviews)
# レビューを1件ずつ処理する
for review in reviews:
    # レビューの本文を翻訳する
    result = translate.translate_text(
        Text=review['Text'],
        SourceLanguageCode='auto', TargetLanguageCode='ja')
    # レビューの本文を翻訳結果に置き換える
    review['Text'] = result['TranslatedText']
# 出力ファイルを開く
with open('trans_json_out.json', 'w', encoding='utf-8') as file_out:
    # JSON ファイルを書き込む
    json.dump(reviews, file_out, indent=4, ensure_ascii=False)
```

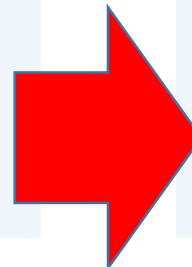
翻訳するJSONファイルと翻訳されたJSONファイル

翻訳するJSONファイル (trans_json_in.json)

```
[
  {
    "Date": "20200105",
    "Name": "Bear",
    "Text": "It was good that shipment was quick."
  },
  {
    "Date": "20200216",
    "Name": "Bär",
    "Text": "Ich möchte, dass die Versandkosten günstiger sind."
  },
  {
    "Date": "20200327",
    "Name": "Ours",
    "Text": "Je veux plus de variations de couleurs."
  },
  {
    "Date": "20200408",
    "Name": "クマ",
    "Text": "また注文したいです。"
  }
]
```

翻訳したJSONファイル (trans_json_out.json)

```
[
  {
    "Date": "20200105",
    "Name": "Bear",
    "Text": "出荷が早かったのは良かった。"
  },
  {
    "Date": "20200216",
    "Name": "Bär",
    "Text": "送料は安くしてほしい"
  },
  {
    "Date": "20200327",
    "Name": "Ours",
    "Text": "カラーバリエーションはもっと欲しい"
  },
  {
    "Date": "20200408",
    "Name": "クマ",
    "Text": "また注文したいです。"
  }
]
```



translate.translate_text() メソッド

Amazon Polly



Amazon Polly

- Amazon Polly はテキストを自然な音声に変換するマネージドサービスです。Amazon Polly は複数の言語をサポートし、さまざまな音声の種類があります。
 - プレーンテキストまたは音声合成マークアップ言語 (SSML) フォーマットから音声を生成する
 - 複数の音声フォーマットで出力を作成する
 - 従量課金制で、AWS インフラストラクチャを使ってコストを抑える

Amazon Polly のユースケース



ニュースサービス制作



語学研修



ナビシステム



アニメーション制作

テキストから音声ファイルを作成する (P88～P92)

テキストから音声ファイルを作成するプログラム (polly_synth.py) P89

```
# boto3 をインポート
import boto3
# contextlib をインポート
import contextlib
# os をインポート
import os
# Polly サービスクライアントを作成
polly = boto3.client('polly')
# 音声合成するテキスト
text = 'こんにちは！音声合成を使ったプログラムを一緒に作りましょう。'
# テキストから音声を合成
result = polly.synthesize_speech(
    Text=text, OutputFormat='mp3', VoiceId='Mizuki')
# 出力ファイルのパス
path = 'polly_synth.mp3'
# 音声のストリームを開く
with contextlib.closing(result['AudioStream']) as stream:
    # 出力ファイルを開く
    with open(path, 'wb') as file:
        file.write(stream.read())
# 出力ファイルを再生する
if os.name == 'nt':
    os.startfile(path)
```

Polly synthesize_speechメソッド

機能: テキストから音声を合成する

使い方: `synthesize_speech(Text=テキスト, OutputFormat=出力形式, VoiceId=音声ID)`

戻り値: 音声合成結果を含む辞書

音声IDを指定する (P93～P98)

Amazon Polly の音声

音声IDの一覧を取得するプログラム (polly_voices.py) P97

```
# boto3 をインポート
import boto3
# pprint をインポート
import pprint
# Polly サービスクライアントを作成
polly = boto3.client('polly')
# 音声IDの一覧を取得
result = polly.describe_voices()
# 取得した結果を整形して表示
pprint.pprint(result)
```

キー	型	値の内容
Gender	文字列	性別
Id	文字列	ID
LanguageCode	文字列	言語コード
LanguageName	文字列	言語名
Name	文字列	名前
SupportedEngines	文字列のリスト	対応するエンジン

Polly describe_voicesメソッド

機能: 音声IDの一覧を取得する

使い方: describe_voices()

戻り値: 音声IDの一覧を含む辞書

```
$> python polly_voices.py
{'ResponseMetadata': {'HTTPHeaders': {'content-length': '10967',
                                         'content-type': 'application/json',
                                         'date': 'Mon, 11 Apr 2022 08:38:00 GMT',
                                         'x-amzn-requestid': '491f642f-3b43-4749-9985-20a568ce612b'},
                      'HTTPStatusCode': 200,
                      'RequestId': '491f642f-3b43-4749-9985-20a568ce612b',
                      'RetryAttempts': 0},
 'Voices': [{'Gender': 'Female',
              'Id': 'Lotte',
              'LanguageCode': 'nl-NL',
              'LanguageName': 'Dutch',
              'Name': 'Lotte',
              'SupportedEngines': ['standard']},
            {'Gender': 'Male',
              'Id': 'Maxim',
              'LanguageCode': 'ru-RU',
              'LanguageName': 'Russian',
              'Name': 'Maxim',
              'SupportedEngines': ['standard']},
            {'Gender': 'Female',
              'Id': 'Ayanda',
              'LanguageCode': 'en-ZA',
              'LanguageName': 'South African English',
              'Name': 'Ayanda',
              'SupportedEngines': ['neural']},
            {'Gender': 'Female',
              'Id': 'Salli',
              'LanguageCode': 'en-US',
              'LanguageName': 'US English',
              'Name': 'Salli',
              'SupportedEngines': ['neural', 'standard']}]}
```

synthesize_speechメソッドの詳細（P99～P101）

synthesize_speechメソッドの引数（P99）

引数名	型	必須	機能
Text	文字列	○	音声を合成するテキスト。プレーンテキストまたはSSMLを記述する
OutputFormat	文字列	○	出力形式。MP3('mp3')、Ogg Vorbis('ogg_vorbis')、PCM('pcm')、JSON('json')のいずれかを指定する。
Voiceld	文字列	○	音声ID
Engine	文字列	×	音声合成のエンジン。標準音声('standard')、またはニューラル音声('neural')が選べる。デフォルトは標準音声。
LanguageCode	文字列	×	言語コード。バイリンガル音声において、どの言語を使用するのかを指定する。
LexiconNames	文字列のリスト	×	音声合成に使うレキシコン（語彙集）の名前
SampleRate	文字列	×	出力サンプルの周波数。Hz（ヘルツ）単位で指定する。
SpeechMarkTypes	文字列のリスト	×	出力するスピーチマークの種類
TextType	文字列	×	テキストの形式。デフォルトはプレーンテキスト。SSMLの場合は'ssml'を指定する

synthesize_speechメソッドの戻り値（P101）

キー	型	値の内容
AutoStream	StreamingBody	合成された音声ストリーム
ContentType	文字列	ストリームの形式
RequestCharacters	整数	合成された文字数

synthesize_speechメソッドの戻り値（P101）

OutputFormat	ContentType
'mp3'	'audio/mpeg'
'ogg_vorbis'	'audio/ogg'
'pcm'	'audio/pcm'
'json'	'audio/json'

標準音声とニューラル音声の違いを比較する (P102～P107)

標準音声を使うプログラム (polly_standard.py) P103

```
# boto3 をインポート
import boto3
# contextlib をインポート
import contextlib
# os をインポート
import os

# Polly サービスクライアントを作成
polly = boto3.client('polly')
# 音声合成するテキスト
text = 'Hello everyone! Welcome to the Programming World
Report.'
# 音声合成する
result = polly.synthesize_speech(
    Text=text, # 音声合成するテキスト
    OutputFormat='mp3', # 音声のフォーマット
    VoiceId='Joanna' # 音声ID
)
# 出力ファイルのパス
path = 'polly_standard.mp3'
# 音声のストリームを開く
with contextlib.closing(result['AudioStream']) as stream:
    # 出力ファイルを開く
    with open(path, 'wb') as file:
        # 音声を読み込んで出力ファイルに書き込む
        file.write(stream.read())
# 出力ファイルを再生する
if os.name == 'nt':
    os.startfile(path)
```

ニューラル音声 (P104～P105)

ニューラル音声を使うプログラム (polly_neural.py) P104

```
# boto3 をインポート
import boto3
# contextlib をインポート
import contextlib
# os をインポート
import os


# リージョンを指定して、Polly サービスクライアントを作成
polly = boto3.client('polly', 'us-east-1')
# 音声合成するテキスト
text = 'Hello everyone! Welcome to the Programming World
Report.'
# 音声合成する
result = polly.synthesize_speech(
    Text=text, # 音声合成するテキスト
    OutputFormat='mp3', # 音声のフォーマット
    VoiceId='Joanna', # 音声ID
    Engine='neural' # ニューラル音声を指定
)
# 出力ファイルのパス
path = 'polly_neural.mp3'
# 音声のストリームを開く
with contextlib.closing(result['AudioStream']) as stream:
    # 出力ファイルを開く
    with open(path, 'wb') as file:
        # 音声を読み込んで出力ファイルに書き込む
        file.write(stream.read())
# 出力ファイルを再生する
if os.name == 'nt':
    os.startfile(path)
```

ニュースキャスタースタイル (P105～P107)

ニュースキャスタースタイルを使うプログラム (polly_news.py) P106

```
# boto3 をインポート
import boto3
# contextlib をインポート
import contextlib
# os をインポート
import os
# リージョンを指定して、Polly サービスクライアントを作成
polly = boto3.client('polly', 'us-east-1')
# 音声合成する SSML 形式のテキスト
text = '''
< speak>
< amazon:domain name="news">
Hello everyone! Welcome to the Programming World Report.
</amazon:domain>
</speak>
'''

# エンジンとテキスト形式を指定してテキストから音声を合成
result = polly.synthesize_speech(
    Text=text, # 音声合成するテキスト
    OutputFormat='mp3', # 音声のフォーマット
    VoiceId='Joanna', # 音声ID
    Engine='neural', # エンジン = ニューラル音声
    TextType='ssml' # テキストのタイプ
)
```



```
# 出力ファイルのパス
path = 'polly_news.mp3'
# 音声のストリームを開く
with contextlib.closing(result['AudioStream']) as stream:
    # 出力ファイルを開く
    with open(path, 'wb') as file:
        # 音声を読み込んで出力ファイルに書き込む
        file.write(stream.read())
# 出力ファイルを再生する
if os.name == 'nt':
    os.startfile(path)
```

レキシコンのファイル (P109)

レキシコンのファイル (my_lexicon.pls) P109

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
  xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
    http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
  alphabet="ipa" xml:lang="ja">
  <lexeme>
    <grapheme>P2P</grapheme>
    <alias>ピアツーピア</alias>
  </lexeme>
  <lexeme>
    <grapheme>WWW</grapheme>
    <alias>ワールドワイドウェブ</alias>
  </lexeme>
</lexicon>
```

レキシコンを登録する (P110～P114)

レキシコンを登録するプログラム (polly_lexicon_put.py) P111

```
# boto3 をインポート
import boto3
# Polly サービスクライアントを作成
polly = boto3.client('polly')
# レキシコンファイルを開く
with open('my_lexicon.pls', 'r', encoding='utf-8') as file:
    # レキシコンを登録
    polly.put_lexicon(Name='MyLexicon', Content=file.read())
```

Polly put_lexiconメソッド

機能: レキシコンを登録する

使い方: put_lexicon(Name=登録名, Content=レキシコンの内容)

戻り値: 辞書 (内容は空)

レキシコンを使って音声を合成する (P114~P115)

レキシコンを使うプログラム (polly_lexicon_enabled.py) P114

```
# boto3 をインポート
import boto3
# contextlib をインポート
import contextlib
# os をインポート
import os
# Polly サービスクライアントを作成
polly = boto3.client('polly')
# 音声合成するテキスト
text = 'P2Pは複数の端末が対等な立場で通信する方式です。'
# 音声合成する
result = polly.synthesize_speech(
    Text=text, # 音声合成するテキスト
    OutputFormat='mp3', # 音声のフォーマット
    VoiceId='Mizuki', # 音声ID
    LexiconNames=['MyLexicon'] # 使用するレキシコン
)
# 出力ファイルのパス
path = 'polly_lexicon_enabled.mp3'
# 音声のストリームを開く
with contextlib.closing(result['AudioStream']) as stream:
    # 出力ファイルを開く
    with open(path, 'wb') as file:
        # 音声を読み込んで出力ファイルに書き込む
        file.write(stream.read())
# 出力ファイルを再生する
if os.name == 'nt':
    os.startfile(path)
```

レキシコンを削除する (P115～P116)

レキシコンを削除するプログラム (polly_lexicon_delete.py) P116

```
# boto3 をインポート
import boto3
# Polly サービスクライアントを作成
polly = boto3.client('polly')
# レキシコンを削除
polly.delete_lexicon(Name='MyLexicon')
```

Polly delete_lexiconメソッド

機能: レキシコンを削除する
使い方: delete_lexicon(Name=登録名)
戻り値: 辞書 (内容は空)

レキシコンに関する他の機能（P117）

Polly list_lexiconメソッド

機能: レキシコンの一覧を取得する

使い方: list_lexicon()

戻り値: レキシコンの一覧を含む辞書

Polly get_lexiconメソッド

機能: 指定したレキシコンデータを取得する

使い方: get_lexicon(Name=登録名)

戻り値: レキシコンデータの場所を含む辞書

スピーチマークを出力する (P118~P122)

出力されたスピーチマーク (polly_mark.txt) P118

```
{
  "time":0,"type":"sentence","start":0,"end":48,"value":"スピーチマークを出力してみます。"}
  "time":6,"type":"word","start":0,"end":12,"value":"スピーチ"}
  "time":6,"type":"viseme","value":"s"}
  "time":80,"type":"viseme","value":"i"}
  "time":160,"type":"viseme","value":"p"}
  "time":223,"type":"viseme","value":"i"}
  "time":391,"type":"viseme","value":"j"}
  "time":502,"type":"viseme","value":"i"}
  "time":560,"type":"word","start":12,"end":21,"value":"マーク"}
  "time":560,"type":"viseme","value":"p"}
  "time":647,"type":"viseme","value":"a"}
  "time":835,"type":"viseme","value":"k"}
  "time":921,"type":"viseme","value":"i"}
}
```



```
{
  "time":1925,"type":"viseme","value":"@"}
  "time":1997,"type":"word","start":36,"end":39,"value":"み"}
  "time":1997,"type":"viseme","value":"p"}
  "time":2064,"type":"viseme","value":"i"}
  "time":2115,"type":"word","start":39,"end":45,"value":"ます"}
  "time":2115,"type":"viseme","value":"p"}
  "time":2196,"type":"viseme","value":"a"}
  "time":2292,"type":"viseme","value":"s"}
  "time":2463,"type":"viseme","value":"i"}
  "time":2544,"type":"viseme","value":"sil"}
}
```

項目	内容
time	音声ストリームの先頭からの経過時間 (単位はミリ秒)
type	スピーチマークの種類
start	文や単語の開始位置 (テキストの先頭からのバイト数)
end	文や単語の終了位置 (テキストの先頭からのバイト数)
value	値。内容はスピーチマークの種類によって異なる

スピーチマークを出力するプログラム (P120)

スピーチマークを出力するプログラム (polly_mark.py) P120

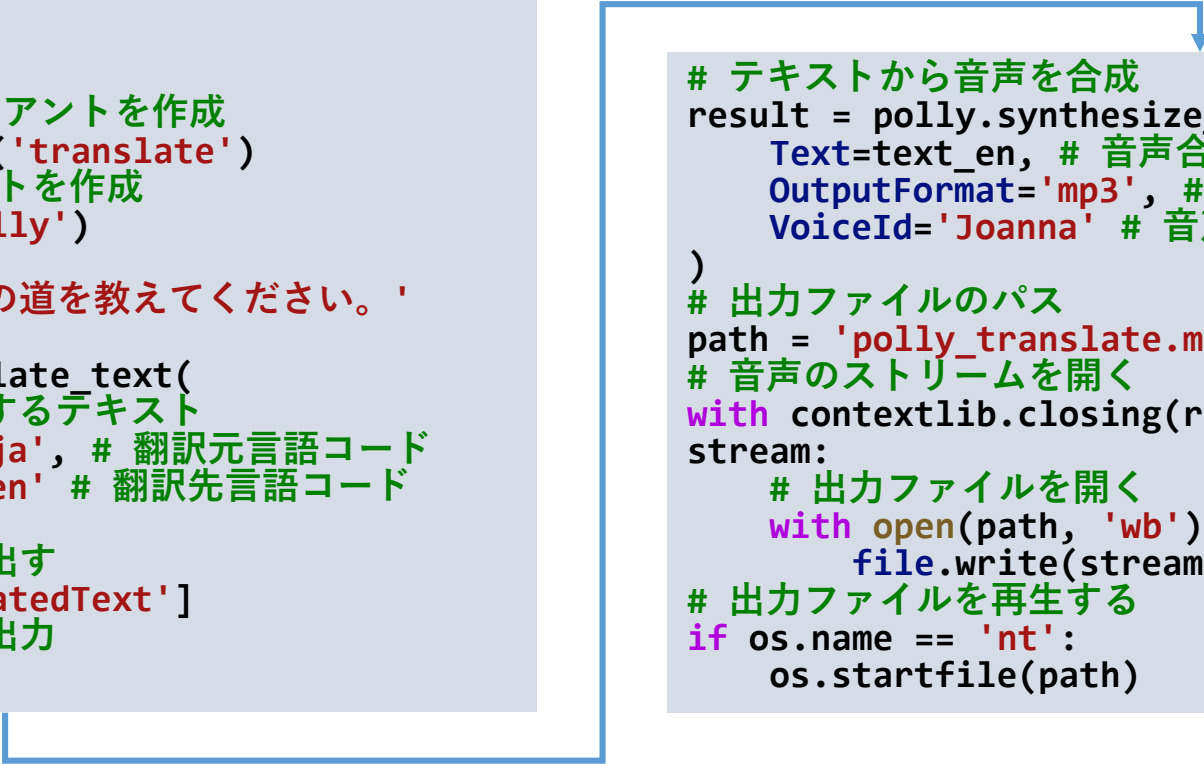
```
# boto3 をインポート
import boto3
# cotextlib をインポート
import contextlib

# Polly サービスクライアントを作成
polly = boto3.client('polly')
# 音声合成するテキスト
text = 'スピーチマークを出力してみます。'
# テキストからスピーチマークを作成
result = polly.synthesize_speech(
    Text=text, OutputFormat='json', VoiceId='Mizuki',
    SpeechMarkTypes=['sentence', 'word', 'viseme'])
# 出力ファイルのパス
path = 'polly_mark.txt'
# 音声のストリームを開く
with contextlib.closing(result['AudioStream']) as stream:
    # 出力ファイルを開く
    with open(path, 'wb') as file:
        # 音声（スピーチマーク）を読み込んで出力ファイルに書き込む
        file.write(stream.read())
```

テキストを翻訳して音声合成する (P123～P126)

テキストを翻訳して音声合成するプログラム (polly_translate.py) P124

```
# boto3 をインポート
import boto3
# contextlib をインポート
import contextlib
# os をインポート
import os
# Translate サービスクライアントを作成
translate = boto3.client('translate')
# Polly サービスクライアントを作成
polly = boto3.client('polly')
# 翻訳するテキスト
text_ja = '一番近い駅までの道を教えてください。'
# 翻訳
result = translate.translate_text(
    Text=text_ja, # 翻訳するテキスト
    SourceLanguageCode='ja', # 翻訳元言語コード
    TargetLanguageCode='en' # 翻訳先言語コード
)
# 翻訳されたテキストを取り出す
text_en = result['TranslatedText']
# 翻訳されたテキストを標準出力
print(text_en)
```



```
# テキストから音声を合成
result = polly.synthesize_speech(
    Text=text_en, # 音声合成するテキスト
    OutputFormat='mp3', # 音声データのフォーマット
    VoiceId='Joanna' # 音声ID
)
# 出力ファイルのパス
path = 'polly_translate.mp3'
# 音声のストリームを開く
with contextlib.closing(result['AudioStream']) as stream:
    # 出力ファイルを開く
    with open(path, 'wb') as file:
        file.write(stream.read())
# 出力ファイルを再生する
if os.name == 'nt':
    os.startfile(path)
```