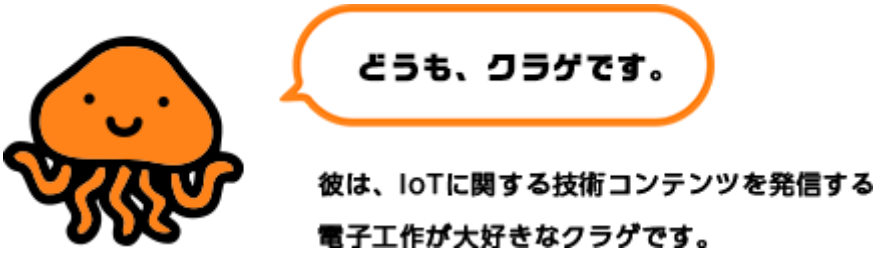
JELLYWARE

MENU

2019/07/08

Python基礎を学ぶ



どうも、ディープなクラゲです。

「OpenVINO™ でゼロから学ぶディープラーニング推論」シリーズの４回目記事です。

このシリーズは、ディープラーニング概要、OpenVINO™ツールキット、Neural Compute Stick、RaspberryPiの使い方、Pythonプログラミングをゼロから徹底的に学び、成果としてディープラーニング推論アプリケーションが理解して作れるようになることを目指します。

今回は、Python基礎を学びます。

基礎から幅広く学ぶことも大事ですが、すぐに使わない項目を学んでも忘れてしまいます。

ここでは実践的に「OpenVINO™ でゼロから学ぶディープラーニング推論」で使うPythonに絞ってじっくりと説明してゆきます！ここで完全に

理解しなくても、ディープラーニング推論を進める上で構文が出てきたときに「そういえばあったな」と思い出し、ここを見返せるようになれば良いかと思います。

【 目次 】

Pythonとは

Pythonはプログラミング言語の１つですが、特徴をあげると以下の３つです。

- スクリプト言語

- 多彩な用途
- インデント

「スクリプト言語」とは、C言語やJAVAのようにコンパイルやビルドが不要で、プログラムを書いたら即実行が可能という意味です。
Pythonは、今回のようにディープラーニングや機械学習などに多く使われていますが、組込みやWebアプリケーションにも使われています。
構文のまとまりにカッコを使わず、インデント（字下げ）で表現するのが特徴的です

このシリーズで扱うPythonは主に以下の３つに分類できます

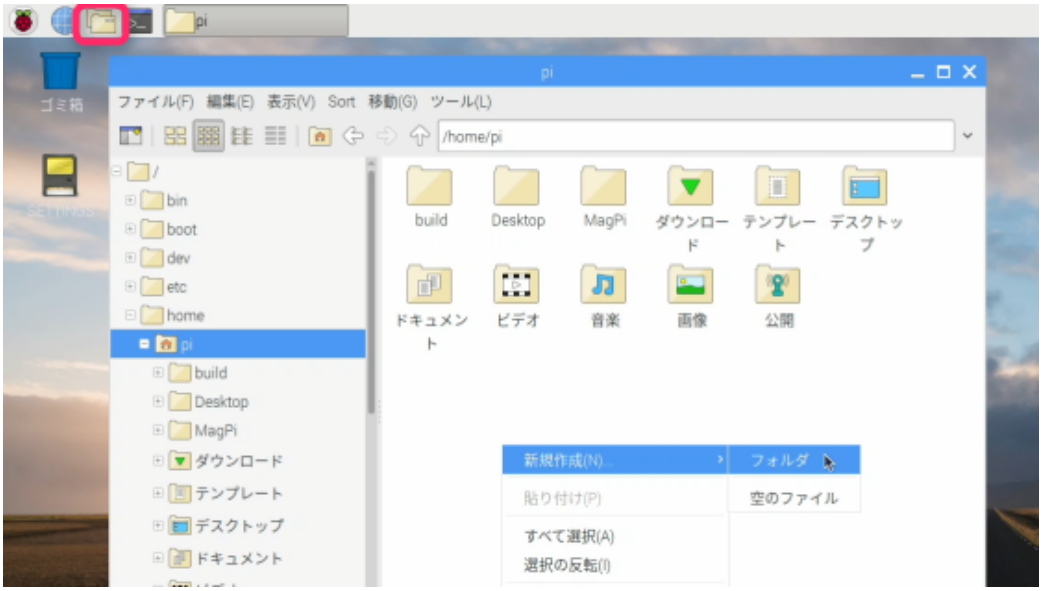
- Python基礎
- OpenCV
- NumPy

今回の記事では「Python基礎」を取り扱います。その他につきましては別の回で取り扱います。
なお、読み進めるにあたり、実際にコードを書いて実行しながら学習することを推奨します。

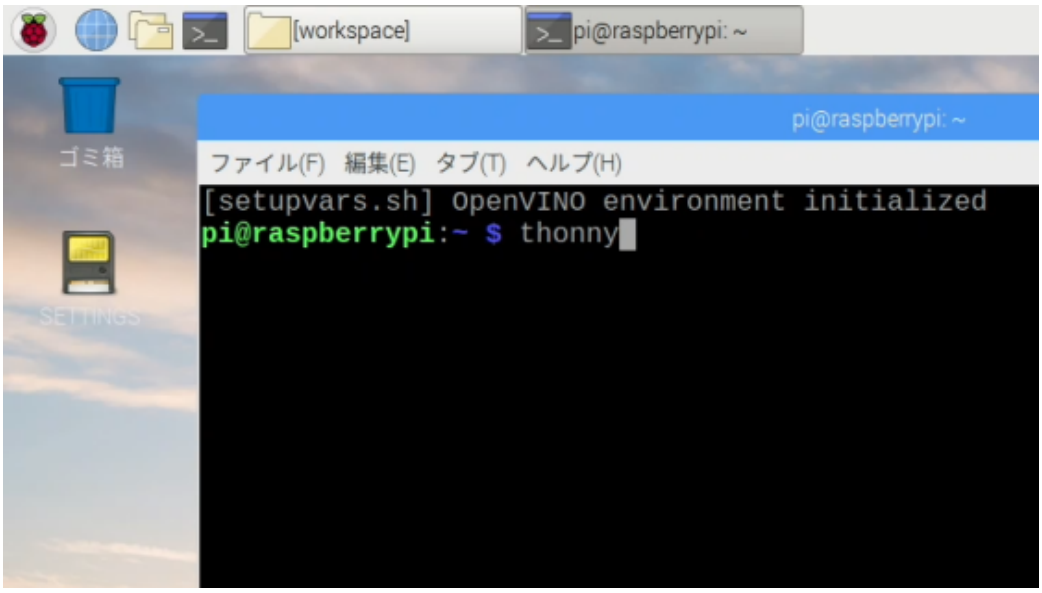
Thonny

プログラミングのコードを記述するためのエディタは様々なものがありますが、ここでは最新のRaspberryPiに標準でインストールされている"Thonny"というGUIエディタを使います。

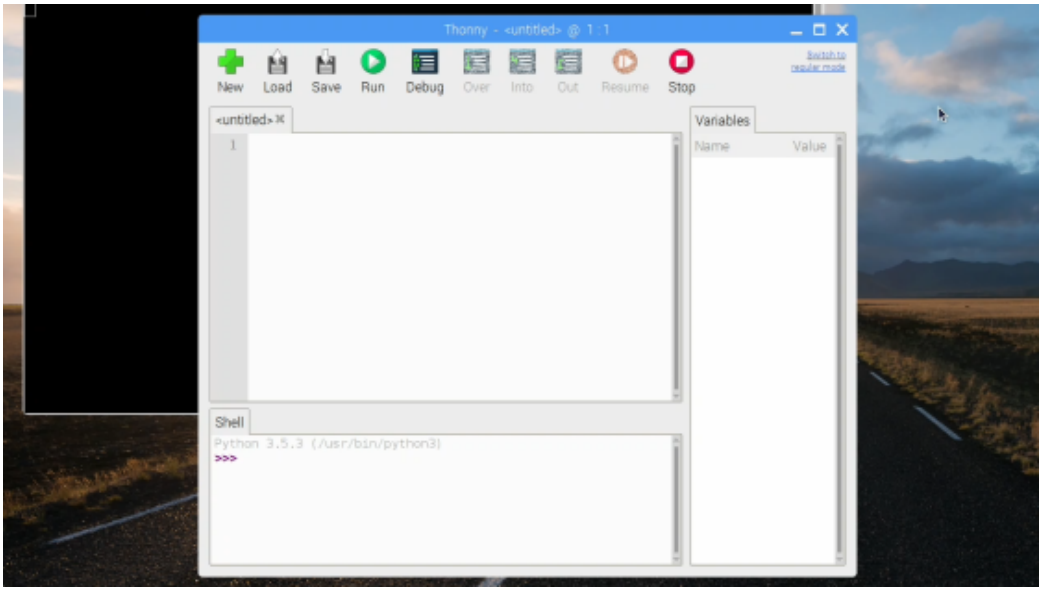
始めにコードを保存するための作業場所を作成しましょう。ユーザーのホームディレクトリの直下に`workspace`というフォルダを作成します。
ファイルマネージャー（左上のフォルダアイコン）を開いて、右クリックで`新規作成`＞`フォルダ`で簡単に作成できます。



次にThonnyを起動しますが、最初の起動方法にポイントがあります。必ずターミナルから`thonny`とコマンドを打って立ち上げて下さい。今回のPython基礎では特に問題は起きませんが、他の方法で立ち上げた場合は、OpenCVやInference Engineを使う際にエラーになります。



次のような画面が立ち上がります。



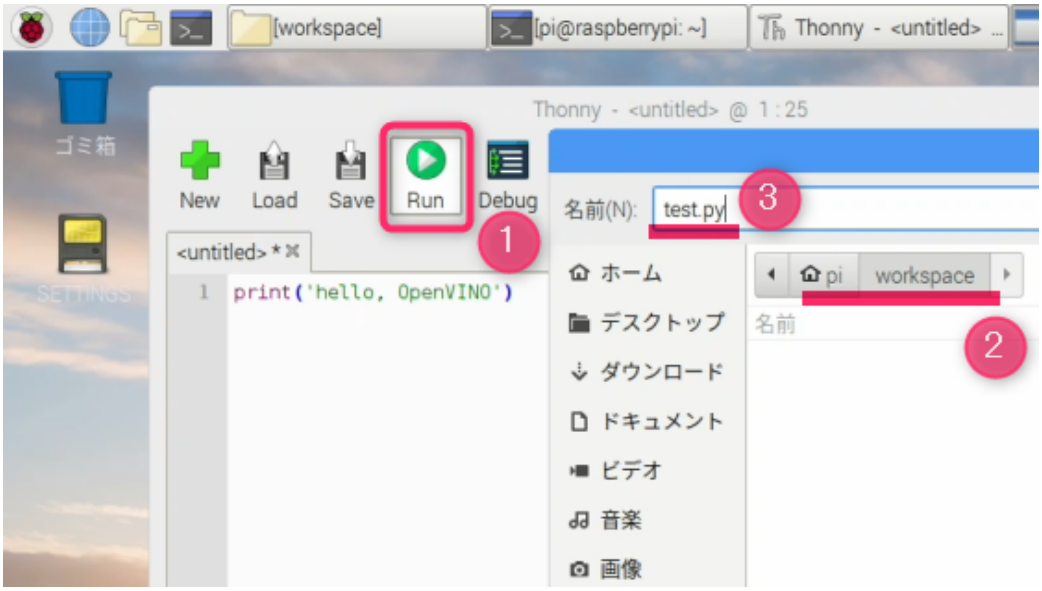
左上がコードを記述する領域、左下が結果を表示する領域、右側は変数の値を表示する領域です。右側の領域は今回は特に使わないので、狭くしたりクローズしたりしても構いません。
クラゲが使うメニューは５つだけです。

- New：新規作成
- Load：既存のコードを開く
- Save：上書き保存
- Run：プログラム実行
- Stop：プログラム強制停止

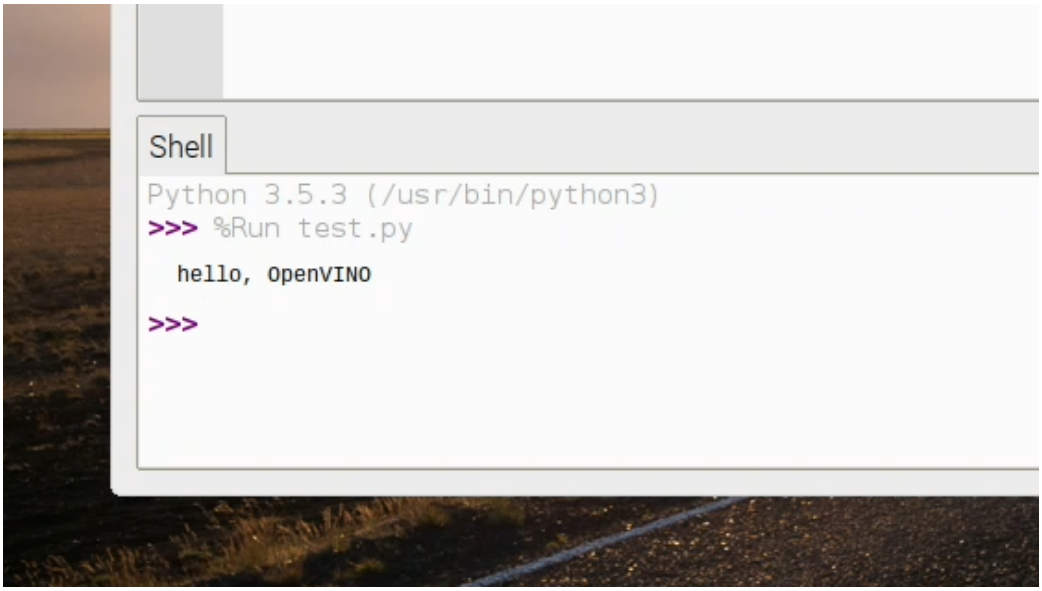
では、試しにコードを書いて実行してみましょう

```
print('hello, OpenVINO')
```

書いたら`Run`ボタンを押します。そうすると保存先を求められるので`workspace`を選んで、テキストボックスに`test.py`と記入して右下の`OK`ボタンを押すかEnterキーを押すと実行されます。



以下のような黒太文字の結果表示がでれば成功です。



これ以降の進め方は`New`で新規作成しても良いですし、test.pyに追記や上書きでも良いです。

コメント

`#`記号を書くことで、それ以降から行の終わりまでがコメント行になります。

コメント行とは、プログラム実行時に無視されるということです。主にプログラムに関する注釈を記述するために使用します。

```
#こんな感じで書きます
```

なお、複数行をコメントしたい場合は様々な方法がありますが、thonnyでは複数行を選択してから`Ctrl + #`キーを押すことで一気にコメント化とコメント解除が可能です。

文字列

文字列はシングルクォート`'`もしくはダブルクォート`"`で囲みます。クラゲはシングルクォートで統一しようと思います。文字列を表示するには以下のような形で`print`を使います。

```
print('hello jellyfish')
```

変数・代入・演算

変数の型宣言は不要です。整数も小数も文字列も同じように扱えます。代入は他のプログラミング言語同様`=`を用いて、左側に右側の値を代入します。

```
a = 2019
b = 3.14
c = 'jellyfish'

print(a)
print(b)
print(c)
```

記号`+`, `-`, `*`, `/`を用いれば四則演算可能です。

```
a = 100 + 200
b = 24 - 8
c = 10 * 3.14

print(a)
print(b)
print(c)
```

以下の２行目のような書き方も良く使われます。自分自身に100を加え、結果的に600になります

```
a = 500
a = a + 100

print(a)
```

「ある変数に数値を足して、結果を同じ変数に代入する」という書き方は良く使われますので覚えておいてください。 `a = a + 1` など `1` を足す場合は "インクリメント" と呼ばれます。

型変換

型を変更させる必要がある場合は、型変換を行う必要があります。
例えば、文字列から数値の小数に変更させる例です。 `float` を使います。

```
a = '3.14'
b = 10 * float(a)

print(b)
```

こちらは小数から整数に変更させる例です。 `int` を使います。

```
a = 3.14
b = 10 * int(a)

print(b)
```

条件分岐

どのプログラミング言語にも必ずある `if` 文を使って条件分岐を行います。Pythonでの注意点はコロンとインデントです。コロンは `if` の最後に書きます。インデントとは字下げのことで、以下のコードの場合はprint文の行です。TABキーを使ってスペースを空けます。
以下のコードは、変数 `time_of_sleeping` の値を見て、8より大きかった場合はprint文で表示を行い、それ以外の場合は何もせずプログラムを終了します。

```
time_of_sleeping = 10

if time_of_sleeping > 8:
    print('よく寝ました')
```

ifと半角スペースの後に書かれている`time_of_sleeping > 8`は条件式と呼ばれる部分です。条件式が合っていれば、その下のインデントされたブロックが実行されます。
条件式には`>`などの比較演算子が使われます。比較演算子は他に、`<` `<=` `>=` `==` `!=` などがあります。

以下の例で`time_of_sleeping`の値を変更してみて、条件式が合っているとき・合っていないときで、どのブロックが実行されているのか確認してください。

```
time_of_sleeping = 6

if time_of_sleeping < 8:
    print('しっかり睡眠を取りましょう！')
    print('クラゲからのアドバイスでした。')
print('以上')
```

ブール型と呼ばれる変数があり`if`とよく絡めて使われるので覚えておきましょう。ブール型は`True`と`False`のどちらかしか持てない変数のことです。
以下のコードでは、ブール変数`sleepy`の値が`True`のときにprint文を表示しています。

```
sleepy = False

if sleepy == True:
    print('仮眠しよう！')
```

リスト

リストは他のプログラミングでいうところの配列にあたります
簡単に言うと、一つの値ではなく、一連の値を格納できる変数です。
代表的な使い方を見てゆきます

リストの初期化

このように各要素をカンマで区切り、カッコ`[ ]`内に記述します。

```
a = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

リストの参照

`リスト名[n]`でリストの要素にアクセスできます
Pythonでは要素番号は 0 から始まります。例えば n が 5 の場合は 6番目の要素をアクセスすることになります。

```
a = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
b = a[5]

print(b)
```

今回は、リストの中身を整数値にしましたが、小数や文字列など他の変数も扱うことも可能です。

辞書

Pythonの辞書は一言で言うと「キーワード付きのリスト」です。WEBプログラミングをやったことのある人であれば、JSON形式という用語を聞いたことがあるかもしれませんが、ほぼそれに近いです。
では、実際にコードを書いて実行してみてましょう。以下は果物の値段を変数に格納して参照する例です。リストで取り扱うことも可能ですが、辞書を使うことにより分かりやすくなります。

```
# 辞書の初期化
a = {'apple':148, 'banana':99, 'orange':358}

# 辞書の参照
b = a['orange']

# 表示
print(b)
```

特徴的なのは、初期化のときに`{ }`を使うことと、`キーワード:値`というセットで格納することです。

リスト同様、`キーワード`と`値`は整数値以外も取り扱うことが可能です。ただし、`キーワード`に関しては文字列を使うことが一般的です。

タプル

タプルとは要素を変更できないリストです。初期化時のカッコの形が`(`)`であり、リストや辞書と異なります。

```
a = (10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
b = a[5]

print(b)
```

本シリーズでは主に、座標（X・Y）、画像の大きさ（幅・高さ）、色情報（青・緑・赤）などに使われます。

関数

関数とは、入力値を受け取り、処理を行った後、出力値を結果として返すものです。
入力値を「引数（ひきすう）」、出力値を「戻り値」と呼びます。
具体例として、半径を引数とし、円周を求める処理を行い、円周値を戻り値として返す関数`circle`を作ってみましょう。

関数は`def`で定義します。
`def 関数名(引数):`を記述した後にインデントを行い、処理をブロックに書きます。
円周の公式は 2 × π × r なので、変数`l`に対して`l = 2 * 3.14 * r`で代入を行っています。
`return 戻り値`で結果を返すことができます。

```
def circle(r):
    l = 2 * 3.14 * r
    return l
```

関数の呼び出し側を加えます。以下は引数に `10` を入れて呼び出す例です。

```
def circle(r):
    l = 2 * 3.14 * r
    return l

val = circle(10)
print(val)
```

関数の引数や戻り値は１つではなく複数でもOKです。
以下は、半径と円周率を引数として円周と円の面積を返す関数の例です。

```
def circle(r, pi):
    l = 2 * pi * r
    s = pi * r * r
    return l, s

val1, val2 = circle(10, 3.14)
print(val1)
print(val2)
```

これまでの関数の引数は、順番を一致させることにより受け渡しをしていました。（例えば上のコードの例では、`r` に `10` を渡し、`pi` に `3.14` を渡す）
関数呼び出し時に名前（キーワード）を付けることにより、順番に関係なく引数渡しを行うことができます。

```
def circle(r, pi):
    l = 2 * pi * r
    s = pi * r * r
    return l, s

val1, val2 = circle(pi=3.14, r=10)
print(val1)
print(val2)
```

順番は逆ですが、先ほどと同じ結果になったと思います。
このような引数の書き方を「キーワード引数」と呼びます。

import

PythonではOpenCVやNumPyなど様々な"モジュール"を読み込みます。
目的はモジュールの中にある複数の"関数"を活用するためです。

モジュールの読み込み方法は主に３通りです。
`time`モジュールの`sleep`という関数を使用する場合を例に説明します。`sleep`は引数に指定した秒だけプログラムを一時停止する関数です

import モジュール名

一番分かりやすい読み込み方法です
関数を使う場合は`モジュール名.関数名`で使います

```
import time

print('start')
time.sleep(3)
print('end')
```

import モジュール名 as 別名

`as`を加えれば、モジュール名を別名にして取り扱うことができます
以下の例ではモジュール名を`t`として取り扱っています

```
import time as t

print('start')
t.sleep(3)
print('end')
```

from モジュール名 import 関数名

特定の関数しか使わない場合、この形式で書くことにより、関数使用時にモジュール名が不要になります

```
from time import sleep

print('start')
sleep(3)
print('end')
```

どの読み込み方法でも同じ結果です。

繰り返し(while)

while True:

`while True:`を使うことで、ブロックに対し、無限ループで実行させることができます。ifと同じようにブロックはインデントする必要があります。

以下は無限ループで変数`num`の値を表示する例です。`sleep`関数で`1`を指定しているので1秒毎の表示です。ブロックの最後で`num`の値はインクリメントされているため、どんどん数値が増えてゆきます。

無限ループなのでプログラムは終了しません。Thonnyでは`Stop`ボタンを押すことでプログラムを強制終了します。

```
import time

num = 0
while True:
    print(num)
    time.sleep(1)
    num = num + 1
```

break

無限ループから抜け出すための`break`も良く使われます

以下は`if`を使って、`num`の値が`5`以上になったらループを抜けてプログラム終了する例です。

```
import time

num = 0
while True:
    print(num)
    time.sleep(1)
    if num >= 5:
        break
    num = num + 1
```

continue

`continue`はそれ以降のブロックをスキップしてブロック先頭に戻る処理を行います。

以下の先ほどのプログラムの`break`を`continue`に変更した例です。`num`が`5`以上になるとインクリメント部がスキップされて処理を繰り返します。

```
import time

num = 0
while True:
    print(num)
    time.sleep(1)
    if num >= 5:
        continue
    num = num + 1
```

繰り返し（for）

forもwhile同様に繰り返し処理ですが、主に無限ループ以外の用途で使います。

for 変数名 in range(n)

0 ～ n-1 までの値を 変数 に代入してブロックを繰り返します。
例えば 5 回繰り返し処理を行いたいというときは、以下のようなコードになります。

```
for num in range(5):
  print(num)
```

for 変数名 in リスト名

リストから順に値を取り出して変数に代入し、ブロックを繰り返します。

```
fruits = ['apple', 'orange', 'banana']

for x in fruits:
  print(x)
```

クラス

クラスと聞くと初心者の方は身構えてしまうと思いますが、ご安心ください。
クラスは、主に「クラスを自分で作成する場合」と「誰かが作ったクラスを使う場合」の2通りに分けることができます。前者はちょっと複雑で、「コンストラクタ」やら「継承」やら「self」やらややこしい用語やお作法も理解する必要がありますが、後者の場合は簡単です。関数に毛が生えたようなものというイメージでOKです。
ここでは具体的なコードは実行せずイメージで軽く説明します。

まず関数についての復習ですが、誰かが作った関数を使う場合は1ステップでいきなり使うことが出来ました。
以下はイメージですが、「クラゲに"kurage-san"という名前を付けて指定座標(100, 200)方向に泳がせる」という関数があったとします。

- 実行

```
# 関数
jellyfish_swim('kurage-san', 100, 200)
```

一方のクラスは以下の2ステップを踏みます

- 登録
- 実行

```
# クラス
a = Jellyfish('kurage-san')
a.swim(100, 200)
```

クラゲが1匹だけで、泳ぐというアクションしかない場合は、クラスを使うメリットは全くありませんが、複数のクラゲを登場させ、それぞれが様々なアクション（眠る、食べる、光るなど）を複数行う場合は、クラスで扱ったほうが管理しやすいというのが何となく理解できるでしょうか？

この「登録」と「実行」を小難しい言葉で言い換えると以下のようになります

- インスタンス生成

- メソッド呼び出し

「インスタンス」とは何か？ 良く例えられるのが「たい焼き」の例です。「クラス」がたい焼きを作る「たい焼き器」で、「インスタンス」は「たい焼き」そのもの。「たい焼き器」は１つしかありませんが、「たい焼き」はいくらでも作ることが可能です。今、これを聞いて理解できなくても全く問題ありません。聞き流してください。プログラムとしては、`インスタンス名 = クラス名(引数)`という形で使うということだけ覚えておいてください。

次に「メソッド呼び出し」ですが、これは簡単です。
クラスは複数の関数を持っていて、その中の１つの関数を指定して呼び出しているということです。プログラムとしては、`インスタンス名.メソッド名`　という形で使います

なお、Pythonではクラス名の最初を大文字にするのが規則ですので、関数なのかクラスなのかはそれで判断できます。

以上、「Python基礎を学ぶ」でした。

Newsletter

ご登録いただくと、新商品やイベント情報をお知らせします。

メールアドレスを入力してください

登録

個人情報について

VISION ABOUT WORKS BLOG

JELLYWARE

〒160-0004 東京都新宿区四谷2-3-6 パルム四谷702号室

03-6273-0758

info@jellyware.jp

個人情報について

CONTACT

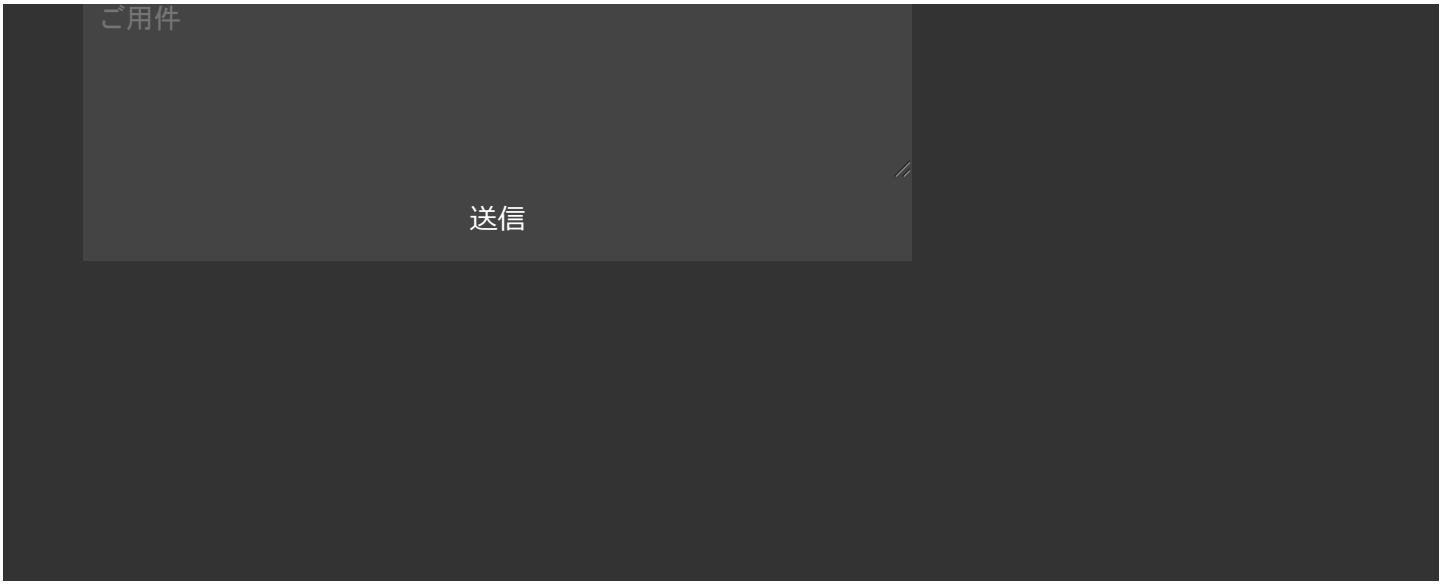
お名前

メールアドレス

メールアドレス確認

お電話番号（任意）

ご件名


ご用件
送信