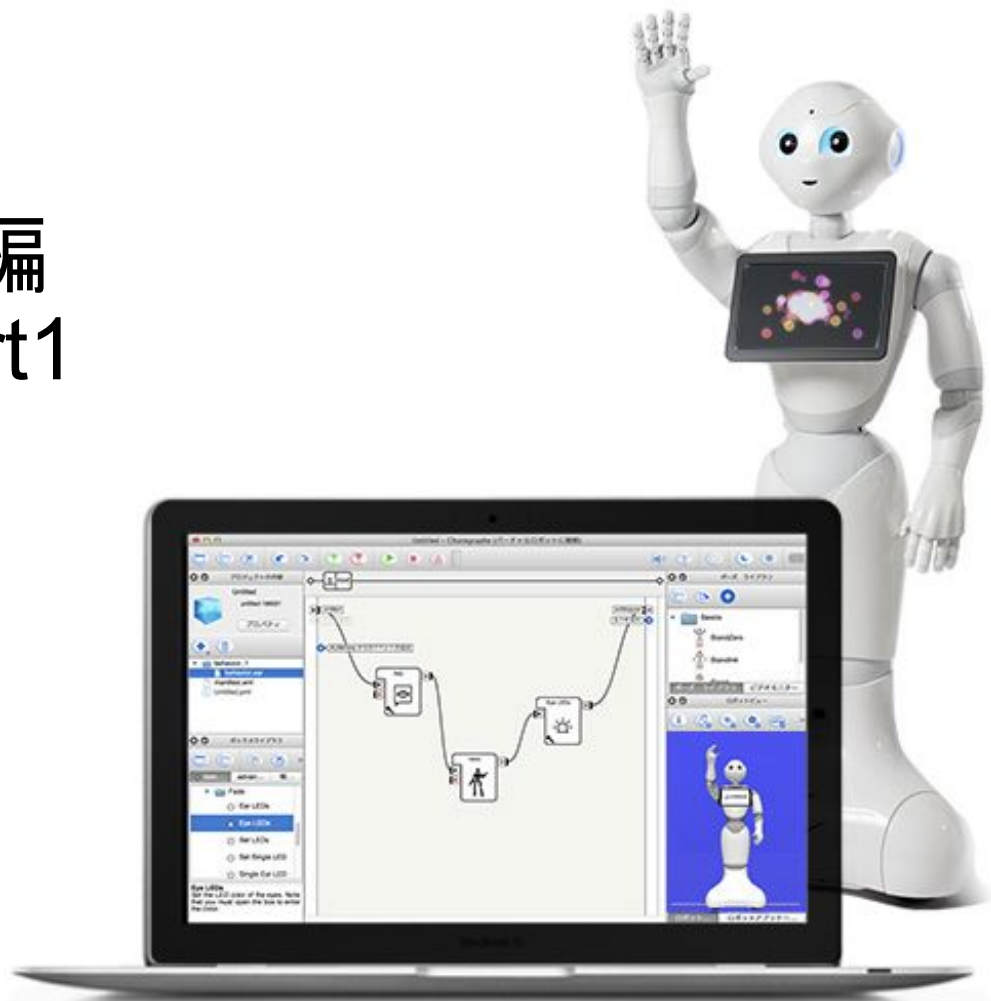


Atelier Akihabara

ワークショップ 番外編

Python入門 Part1



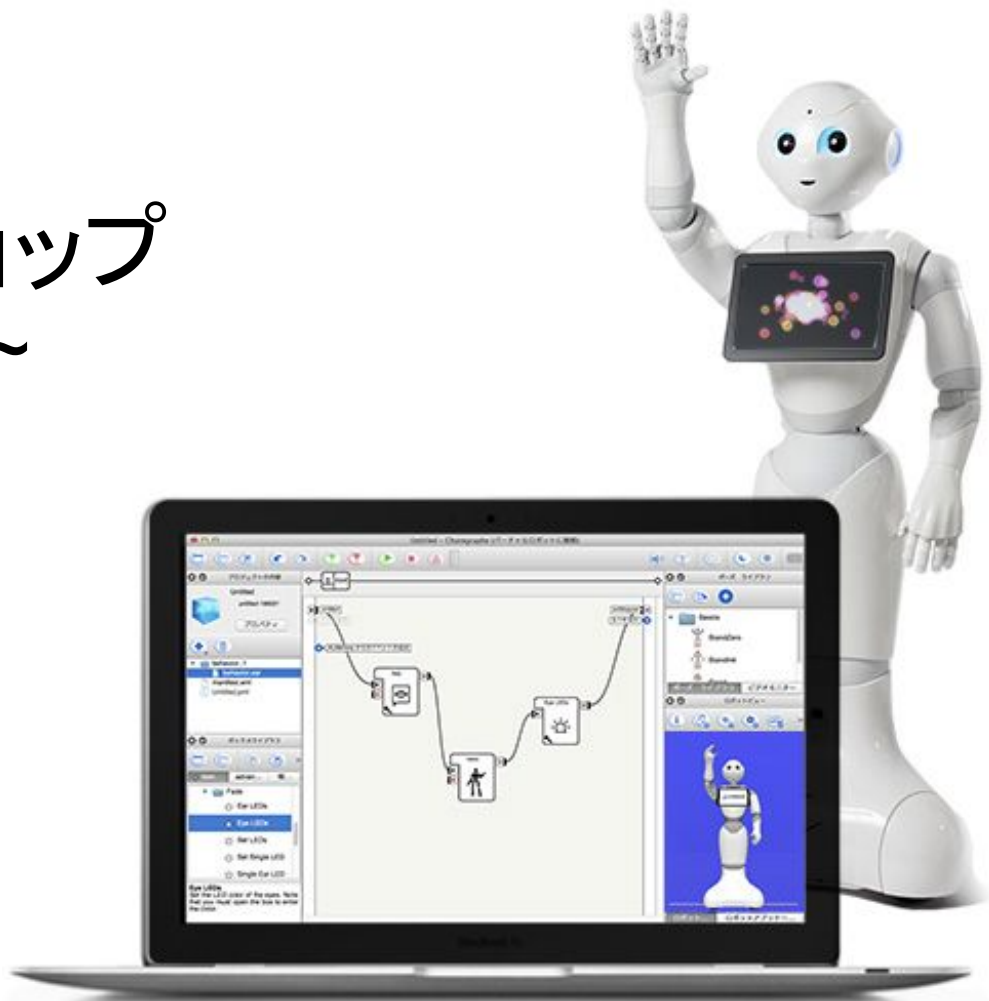
免責事項

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
ソフトバンクロボティクス(株)公式のものでは
ないことをご了承ください。

アトリエ秋葉原

Pepper ワークショップ

Python入門~Part1~



アトリエ秋葉原とは

実体験とコミュニティで開発を促進する

アトリエ

コミュニティ



Pepperのアプリ開発という
実体験

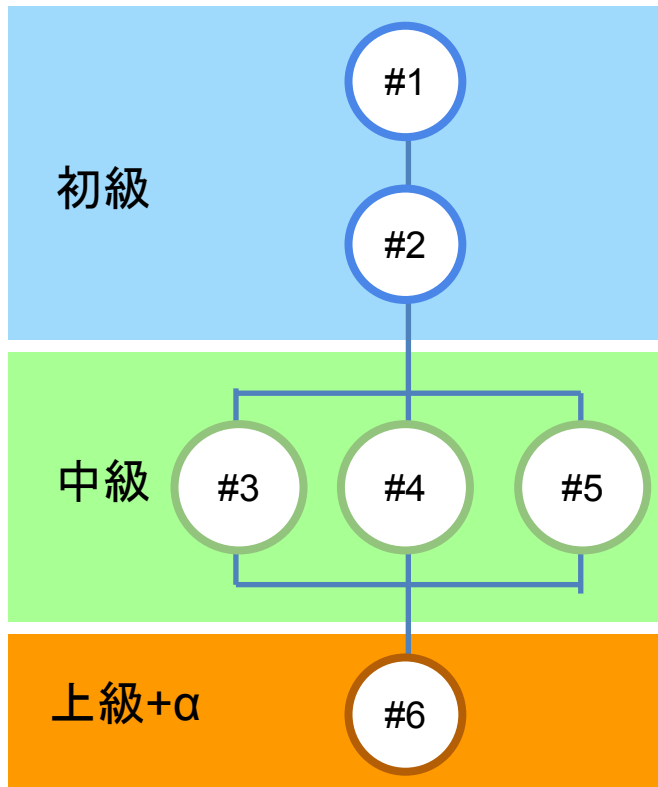
相互
促進



経験や知見を
コミュニティで共有

アトリエ秋葉原のサービス

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

アトリエサテライトについて

実体験とコミュニティで開発を促進する



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

軽く自己紹介をしましょう！

- お名前
- 所属
- プログラミング経験や本日の意気込み

今回ワークショップ講師を務める

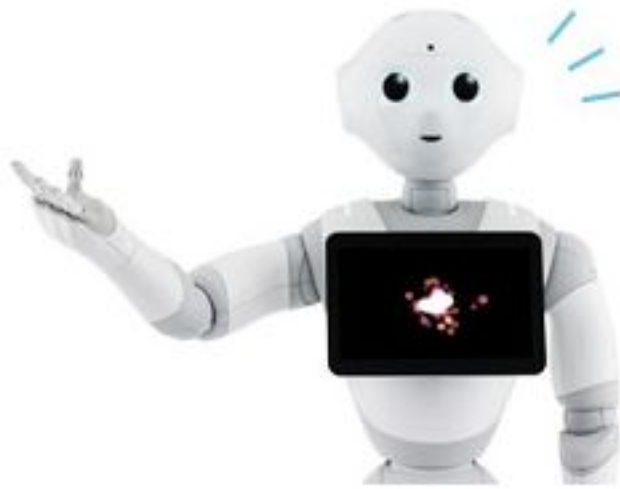
＊ ＊と申します。

よろしくお願いします

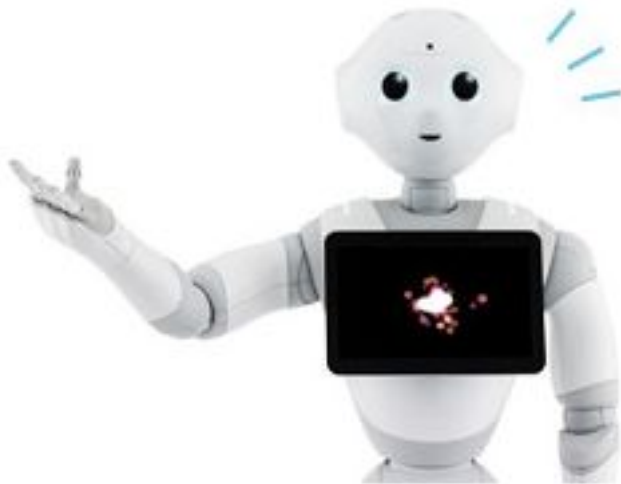
Pythonの基本的な記述の仕方を学ぶ

- 指定した文字列を出力する
- 変数
- リスト
- 辞書

このワークショップでは
基本的なpythonの文法を
学びます



Pythonとは？



Pythonとは

- 1991年にオランダ人のグイド・ヴァン・ロッサム氏によって開発された 汎用的なプログラミング言語
- There's only one way to do it
 - 読みやすく、効率もよいコードをなるべく簡単に書けるようにする

日本語では
ニシキヘビ



① シンプルな文法

- インデントを強制することで誰が書いても似たようなコードになる

```
int factorial(int x)
{
    if (x == 0) {
        return 1;
    } else {
        return x * factorial(x - 1);
    }
}
```

C言語

```
int factorial(int x) {
    if(x == 0) {return 1;} else
    {return x * factorial(x - 1); } }
```

Python

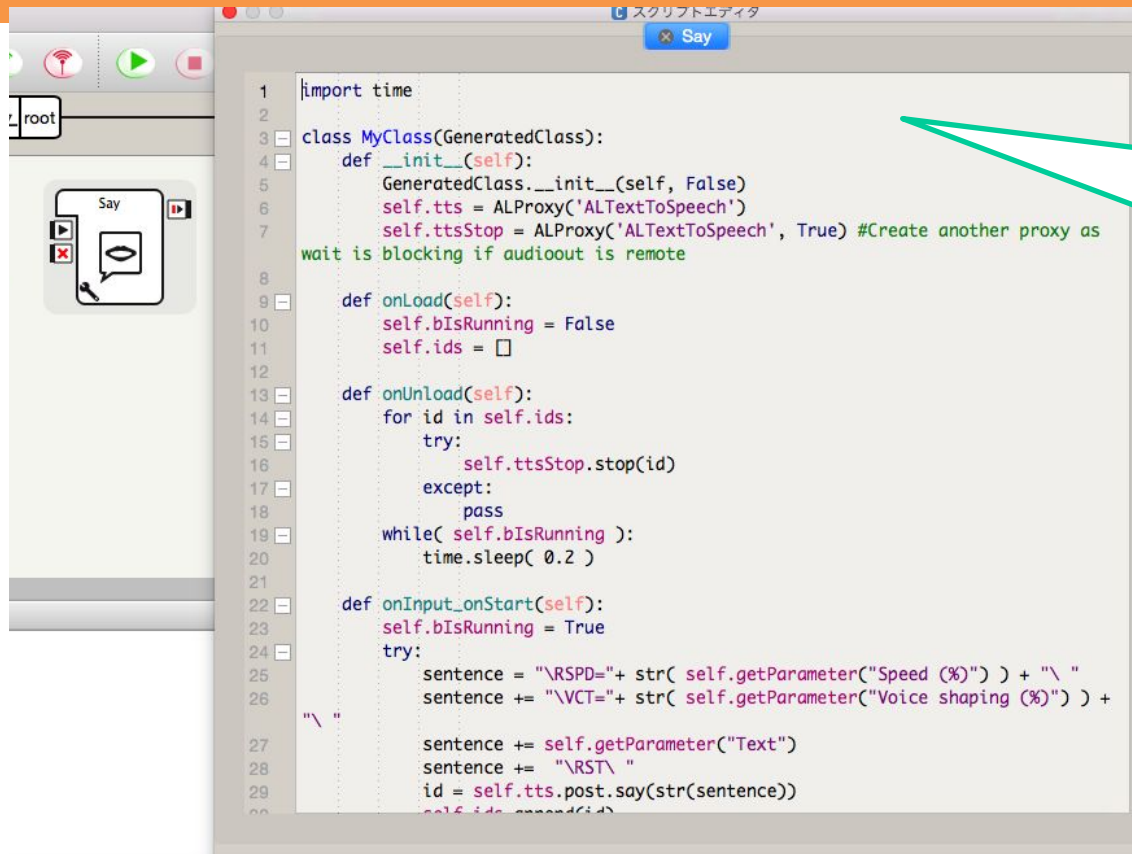
```
def factorial(x):
    if x == 0:
        return 1
    else:
        return x * factorial(x - 1)
```

② 豊富なライブラリ群

- 特に数学系のライブラリが充実

- 数学計算からデータベース、Web 開発、GUI アプリの作成等

Pythonの特徴



高度なアプリ開発にPythonは必須！

インストール方法

参考(Windows):

<http://qiita.com/taiponrock/items/f574dd2cddf8851fb02c>

参考(mac):

<http://qiita.com/ms-rock/items/6e4498a5963f3d9c4a67>

実行方法

-コマンドライン上で[python ファイル名]を入力

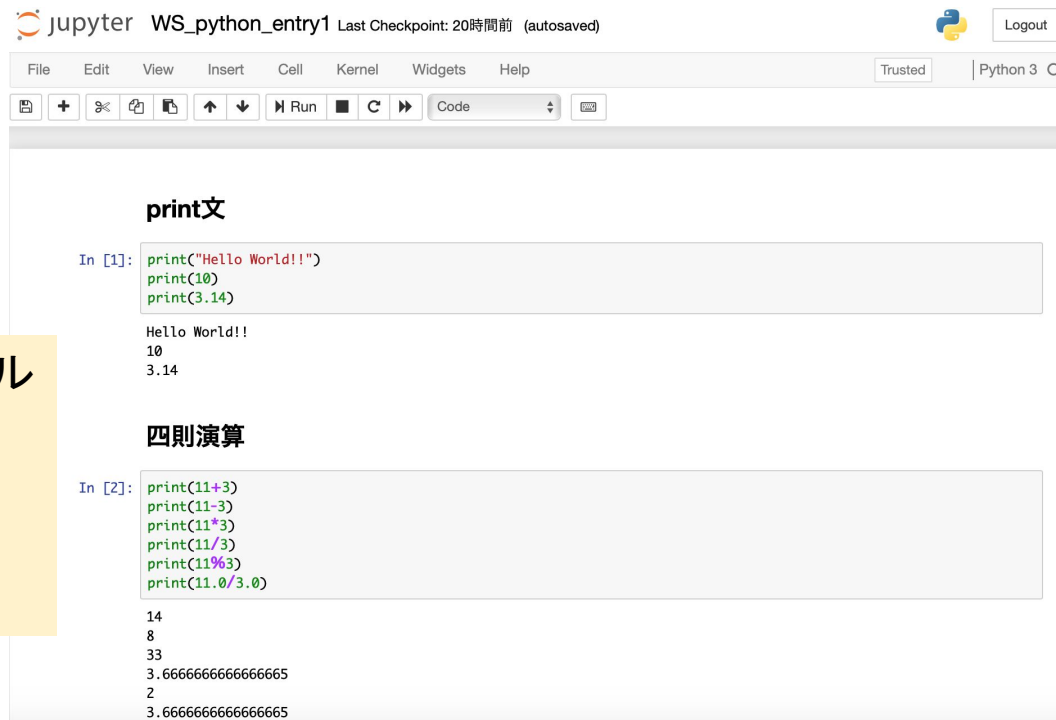
```
ooyamamariho-no-MacBook-Air:jibun mariho$  
ooyamamariho-no-MacBook-Air:jibun mariho$ python panda.py
```

本講座で使用する開発環境: Jupyter notebookとは？

Pythonコードをセルごとに対話的に実行できる開発環境

コードだけでなく、
HTMLのようなMarkdown形式で
書かれた文章も組み込むことが
できる

コードの実行を行うには、実行したいセル
に編集カーソルを合わせて、
Shift+Enter (Windows)
shift+return (mac)



The screenshot shows a Jupyter Notebook interface. At the top, it says "jupyter WS_python_entry1" and "Last Checkpoint: 20時間前 (autosaved)". There is a "Logout" button and a Python logo. Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3" buttons. Below the menu bar is a toolbar with icons for "New", "Open", "Save", "Copy", "Paste", "Undo", "Redo", "Run", "Clear", and "Code". The main area contains two code cells. The first cell is titled "print文" and contains the code: `print("Hello World!!")`, `print(10)`, and `print(3.14)`. The output of this cell is: `Hello World!!`, `10`, and `3.14`. The second cell is titled "四則演算" and contains the code: `print(11+3)`, `print(11-3)`, `print(11*3)`, `print(11/3)`, `print(11%3)`, and `print(11.0/3.0)`. The output of this cell is: `14`, `8`, `33`, `3.6666666666666665`, `2`, and `3.6666666666666665`.

```
jupyter WS_python_entry1 Last Checkpoint: 20時間前 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
New Open Save Copy Paste Undo Redo Run Clear Code

print文
In [1]: print("Hello World!!")
        print(10)
        print(3.14)

Hello World!!
10
3.14

四則演算
In [2]: print(11+3)
        print(11-3)
        print(11*3)
        print(11/3)
        print(11%3)
        print(11.0/3.0)

14
8
33
3.6666666666666665
2
3.6666666666666665
```



アトリエ秋葉原のサーバ上のJupyter notebookにアクセス

ブラウザ上でurlの欄に「192.168.100.X:Y」と書き込む。
(X,Yには数字が入ります。数字は口頭でお伝えします)

[Quit](#)[Logout](#)[Files](#)[Running](#)[Clusters](#)

Select items to perform actions on them.

[Upload](#)[New ▾](#)☐ 0

/

Name ▾

Last Modified

File size

☐

WS_python_entry1

1時間前

☐

WS_python_entry2

1時間前

☐

WS_python_entry3

2時間前

☐

WS_TF1

1日前



Python文法入門



—print文— 出力する

ws.py

```
print(出力させたい内容)
```

Choregraphe上

```
self.logger.info(出力させたい内容)
```

```
print("Hello World!!")  
print(10)  
print(3.14)
```

```
Hello World!!  
10  
3.14
```

文字列の場合は" "で囲む必要がある。(変数名と区別するため)
数字はそのままよい。

一四則演算一 和・差・積・商・剰余

演算	演算子
和	+
差	-
積	*
商	/
剰余	%

```
print(11+3)
print(11-3)
print(11*3)
print(11/3)
print(11%3)
print(11.0/3.0)
```

注意点:

Pepperで使用されるPython2系では `print(11/3)`→3(整数)となることに注意。

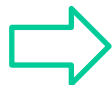
Python2系では商を小数で出力するには `print(11.0/3.0)`とすることがある。

—変数— 値を格納する

型	意味	例
int	整数値	1 , 150 , -30
float	浮動小数点数	1.414 , 0.5 , -1.2
str	文字列	“Apple” , “Hello!” , “50”

- Pythonでは型の指定をする必要がない(代入時)

```
x = 5  
y = 1.5  
z = “Apple”
```



自動でxはint型、yはfloat型、zはstr型と認識

```
int x = 5;  
float y = 1.5;  
str z = “Apple”;
```

フォーマット指定子

型	指定子
int	%d
float	%f
str	%s

使い方

文字列にint型の変数を代入すると宣言

“ x = %d ” % x

代入する変数

—format関数— 文字列に変数を埋め込む

- 文字列に変数を埋め込み出力することができる

“任意の文字列1{ }任意の文字列2”.format(変数)

⇒ 任意の文字列1変数任意の文字列2

“文字列1{0}文字列2{1}文字列3”.format (変数0, 変数1)

⇒ 文字列1変数1文字列2変数2文字列3

—print文— 文字・変数を出力する

```
x0 = 10  
x1 = 1.5  
x2 = "apple"  
print(x1+0.5)  
print(x2+"banana")  
print("x0 : {0},x1 : {1},x2 : {2}".format(x0,x1,x2))
```

```
2.0  
applebanana  
x0 : 10, x1 : 1.5, x2 : apple
```

ーリストー 複数の値を一つの値として扱う

- 値を複数格納することができる(配列という)
 - 値1つ1つのデータを要素という
 - データのいる場所(何番目にいるか)をindexという
 - リスト名[index(数字)]で要素を取り出すことができる

```
data = [5,19,-8,3,22,93]
```

index

0

1

2

3

4

5

data

5

19

-8

3

22

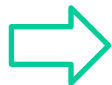
93



data[4]

ーリストー 文字列も配列

z = "apple"



z

a	p	p	l	e
---	---	---	---	---

```
z = "apple"  
list = [5,3.2,"melon",'b']  
print(z[3])  
print(list[2])  
print(list[0]+3)  
print(list[3]+"anana")
```

—辞書—

- 1組の値(keyとvalue)を複数格納することができる
 - 同じ辞書内に同じkeyは使えない
 - 順序は考慮されていない

```
pepper =  
{“height”:121,“weight”:29,“atelier”:“akihabara”}
```

pepper

key	value
height	121
weight	29
atelier	“akihabara”

```
pepper = {"height":121,"weight":29,"atelier":"akihabara"}  
print(pepper["height"])      #値の取得  
print(pepper)  
pepper["company"]="softbank" #要素の追加  
print(pepper)
```

	key	value	
pepper	height	121	
	weight	29	
	atelier	"akihabara"	
	company	"softbank"	←追加

—演習問題—

①花子さんの身長(cm)・体重(kg)が格納されている辞書がある。BMIを小数点第2位まで求めて出力せよ。
hanako = {"height":150,"weight":42}

はなこさんのBMIは18.67です。

変数がfloat型の場合：
(y: {:.2f}).format(y)と書くと
小数点以下の桁数指定ができる。

BMI:
$$\frac{\text{体重(kg)}}{\text{身長(m)} \times \text{身長(m)}}$$

ヒント

・辞書のvalueの取り出し方は？
→辞書名[key]

—演習問題—

②花子さんと太郎さんの身長・体重が格納されている辞書のリストがある。BMIを小数点第2位まで求めて出力せよ。

data

```
= [{"height":150,"weight":42}, {"height":170,"weight":60}]
```

はなこさんのBMIは18.67で太郎さんのBMIは20.76です。

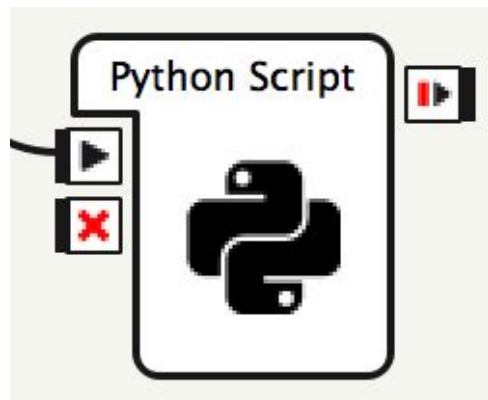
リストの要素が辞書になっている！

→リストから辞書を取り出して、valueを取り出す

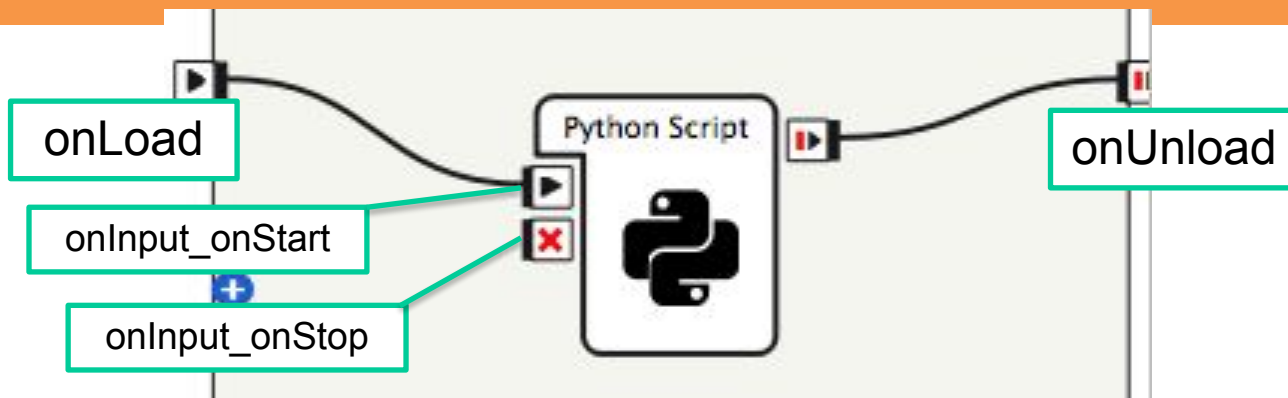
ChoregrapheでPythonを記述する場合について



使うボックスはこれ



Python Script boxの関数



<code>__init__</code>	behaviorが読み込まれた時
<code>onLoad</code>	親ボックスの <code>onStart</code> が実行された時
<code>onUnload</code>	親ボックスの <code>onStop</code> が実行された時
<code>onInput_onStart</code>	ボックスの <code>onStart</code> に信号が入力された時
<code>onInput_onStop</code>	ボックスの <code>onStopped</code> に信号が入力された時

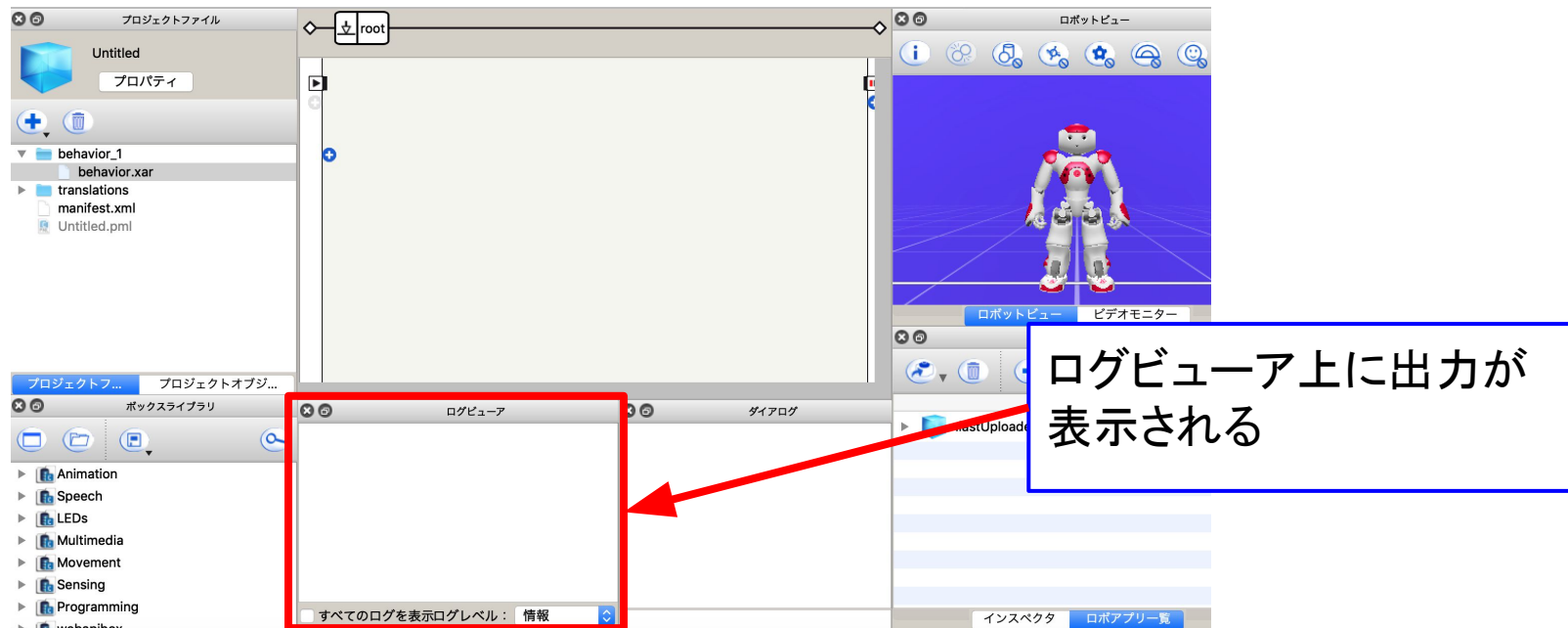
Python Script boxの関数

```
class MyClass(GeneratedClass):  
    def __init__(self):  
        GeneratedClass.__init__(self)  
  
    def onLoad(self):  
        #put initialization code here  
        pass  
  
    def onUnload(self):  
        #put clean-up code here  
        pass  
  
    def onInput_onStart(self):  
        #self.onStopped() #activate the output of the box  
        pass  
  
    def onInput_onStop(self):  
        self.onUnload() #it is recommended to reuse the clean-up as the box is stopped  
        self.onStopped() #activate the output of the box
```

ここに書く

Choregraphe上で出力を得るには

Pythonでは標準的な出力を得るためには`print()`関数を用いるのが一般的であるが、Choregraphe上で出力を得るためには**`self.logger.info()`**関数を用いる。



まとめ



Pythonの基本的な記述の仕方を学ぶ

出力(print文)

四則演算

変数の型

リスト

辞書

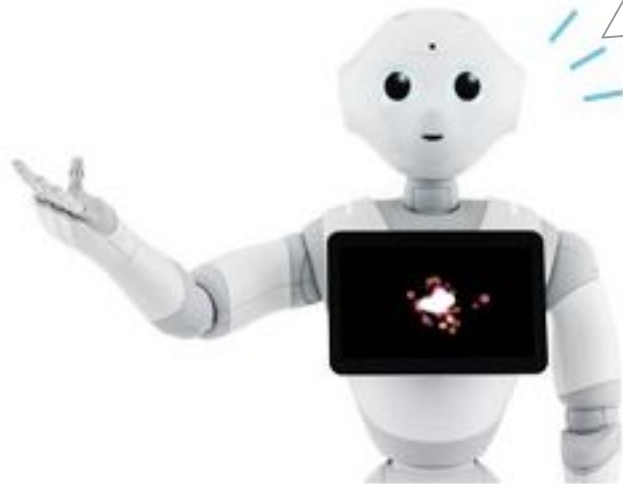
ここまで→ choregraphe特有のpythonの記述の仕方

条件分岐(if文)

繰り返し文(for文、while文)

関数とは

クラスとは



おまけ





ホーム アトリエ秋葉原とは 利用予約 アトリエ サテライト SDK FAQ リンク集

「Arduinoファンもくもく会#019 with アトリエ秋葉原(Pepper開発体験)」 イベントレポート



先日アトリエ秋葉原にてArduinoファンもくもく会#019 with アトリエ秋葉原(Pepper開発体験)を開催... [Read More »](#)

いいね! 0

Tweet

イベント

イベントレポート

AtelierStaff

Pepper アトリエ秋葉原 with SoftBank

「アトリエ秋葉原 ブログ」で検索

・ワークショップのスライドをダウンロードできます

・イベントの紹介とイベントのレポートが見ることができます

ダウンロード

ワークショップ教材
80点のモーションライブラリ

最近の投稿

「Arduinoファンもくもく会#019 with アトリエ秋葉原(Pepper開発体験)」 イベントレポート

今回は温度センサとPepperを連動させたラズパイハンズオン!



アトリエ秋葉原FBグループ

「アトリエ秋葉原 FB」で検索

・アトリエ秋葉原のFacebookグループです

・情報共有や質問ができます

Qiita  ホーム  コミュニティ 



Pepper

フォロー中

435

投稿

661

フォロワー

Pepperに関する情報が集まっています。現在435件の投稿があります。また661人のユーザーが Pepperタグをフォローしています。

最近いいねされた投稿



doki_k が2018/05/21に投稿

AWS IoT で Pepper と RaspberryPi 間を MQTT でやり取りしてみる

 6



Python



RaspberryPi



Pepper



awsIoT



hws-hitorobo が2015/07/23に投稿 

PEPPER 目のLEDを複雑に光らせる

 18




Pepper



Choregraphe



yuka_nm が2016/09/23に投稿 

Watson Speech to Text を使ってPepper同士は会話できるのか? ~シンプルな伝言からラップバトルへの挑戦まで

 26



やってみた



Pepper




Watson



SpeechToText



JohnTomato が2015/12/15に投稿 

PepperとWatson SpeechToTextAPIを連携させて継続的な音声認識サービスを作ってみた

 52



AdventCalendar



Bluemix



NAOqi



Pepper



Watson



kakkey が2016/08/23に投稿 

長押ししてイベント発火させるボックスで、長押しのタイミングを分かるように改善してみた

 1

Qiita

「Qiita pepper」で検索

・プログラミングに関する知識を
記録・共有するためのサービス

・Pepperタグに有志による
Pepperに関する様々な技術情報があります



アップロード済み すべて再生

≡ 並べ替え



第17回 Pepperと外部センサーを繋いでみよう! (Mesh編)

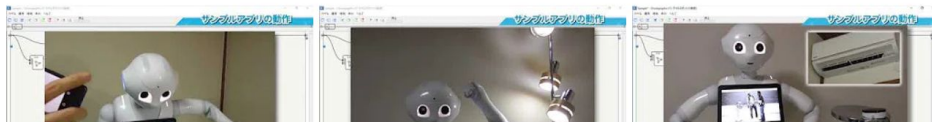
視聴回数 108 回・4 か月前

第16回 PepperにSlackから指示を送ってみよう!

視聴回数 149 回・6 か月前

第15回 Pepperでテレビ画面を制御してみよう!...

視聴回数 47 回・7 か月前



Pepper Developer Network

「Pepper Developer Network」で検索

- ・SBR公式Youtubeチャンネル

- ・ロボアプリ開発でよく使うパターンや知らないハマっちゃうポイントについて説明しています。

お疲れ様でした

おつかれさまでした！
これにてPepperで学ぶPython講座その1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

お帰りの際はアンケートの記入に
ご協力ください



<https://bitly.com/atelierakb>



一演習問題一 答え

①花子さんの身長・体重が格納されている辞書がある。BMIを小数点第2位まで求めて出力せよ。

```
hanako = {"height":150,"weight":42}  
bmi = hanako["weight"]*10000.0/(hanako["height"]*hanako["height"])  
print("はなこさんのBMIは{:.2f}です。".format(bmi))
```

一演習問題一 答え

②花子さんと太郎さんの身長・体重が格納されている辞書のリストがある。BMIを小数点第2位まで求めて出力せよ。

```
data = [{"height":150,"weight":42},{"height":170,"weight":60}]
bmi1 = data[0]["weight"]*10000.0/(data[0]["height"]*data[0]["height"])
bmi2 = data[1]["weight"]*10000.0/(data[1]["height"]*data[1]["height"])
print("はなこさんのBMIは{0:.2f}でたろうさんのBMIは{1:.2f}です".format(bmi1,bmi2))
```