１．Python 入門

1.1　なぜ Python なのか

図 3 に示すように、RaspberryPi など Linux ベースのボードコンピュータの場合、なぜ使われる言語は C 言語ではなく Python や Node.js が注目されるのでしょうか。いくつかの理由がありますが、ともに、バーチャルマシンで実行を行い、コンパイル作業、コンパイル環境が不要です。さらに、高機能であり、豊富なライブラリが用意されていています。また、図 4 のようにバーチャルマシンがハードウェアの違いを吸収してくれるので異なるプラットフォーム間でもほぼ共通に使えるという点です。

図 5 に示すように、Python には 2.x　と　3.x　のバージョンがあり、細かな点で仕様が異なります。本テキストでは 3.x を利用します。

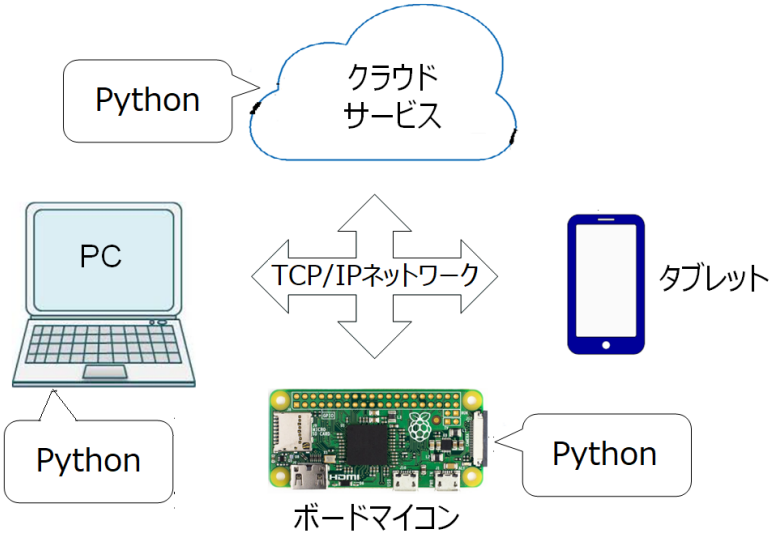


図 3　Python の守備範囲

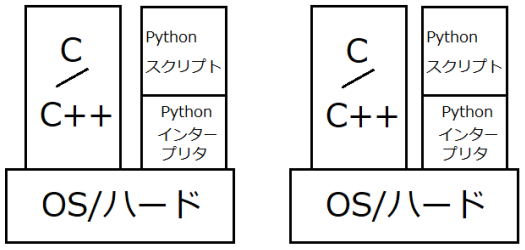


図 4　Python の実行環境

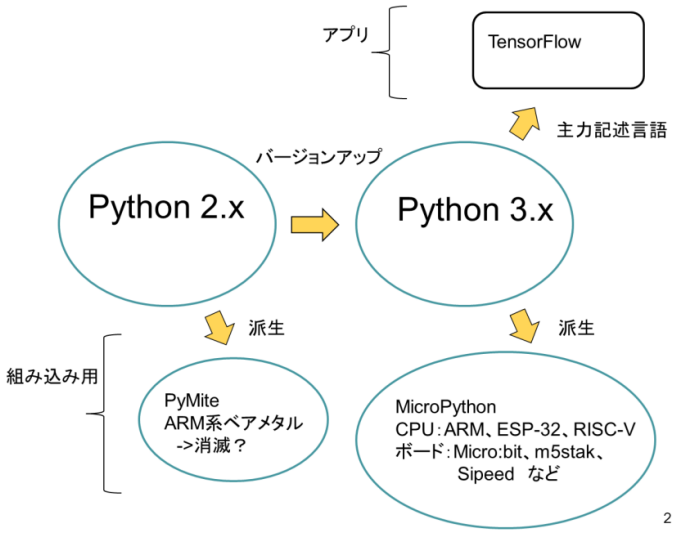


図 5　Python の進化

Rython インタープリタが OS/ハードの違いを吸収してくれる。

1.2　PC へのインストールと起動

異なるプラットフォーム間でもほぼ共通に使えるという点では、ボードコンピュータと PC とで同様に動かすことができ、どちらからでもある程度のデバッグや動作テストができます。PC へのインストールは図 6、図 7 に示すように、Python のページからインストーラをダウンロードして行います。パスの追加をチェックしておけばあとは何も設定しなくとも利用できるようになります。

ダウンロード先アドレス　`https://www.python.org/downloads/`

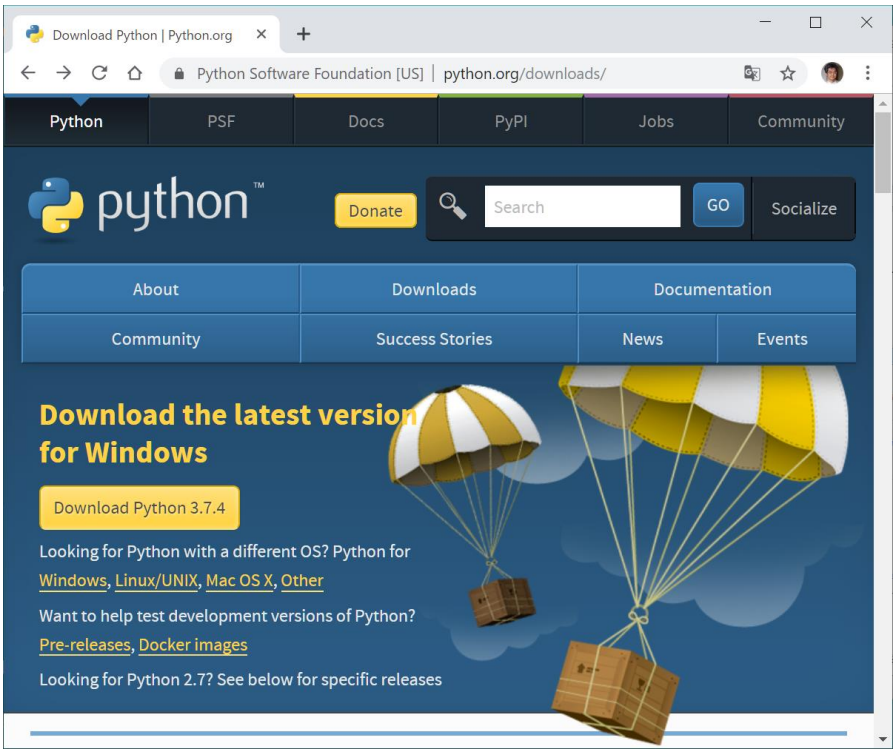


図 6　Python のダウンロードページ(2019.08 時点)

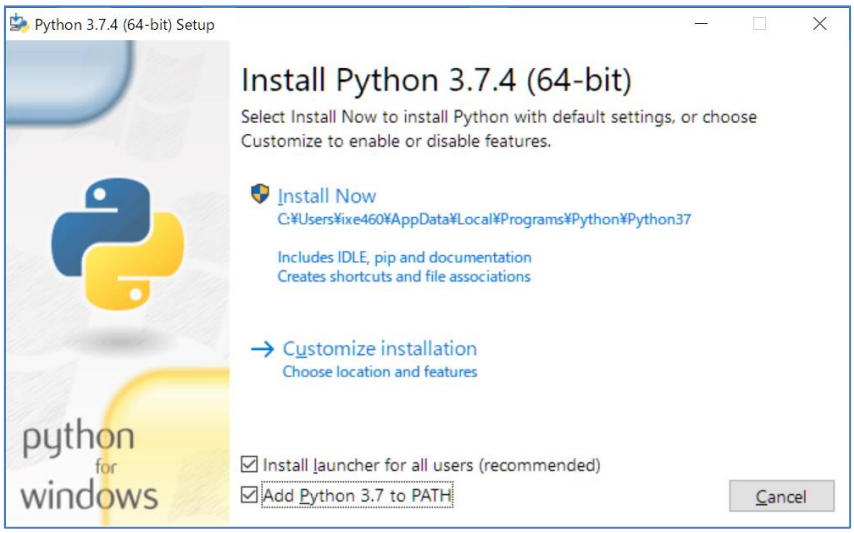


図 7　PC 用のインストーラ画面

〇Python の起動

PC でもボードコンピュータでも、基本的にはエディタでソースコードを書いて、例えばそのファイルが xxx.py とすると、コマンド窓から

```
python xxx.py
```

で起動します。RaspberryPi の場合、Python2.x と Python3.x 双方が既にインストールされています。この場合は図 8 に示すように Python2.x を起動する場合は python 、Python3.x を起動する場合は python3 で起動します。図 9 はコマンド窓のデフォルトのフォルダに pytest というフォルダを作り、そこに list00.py というプログラムファイルを用意して実行しています。

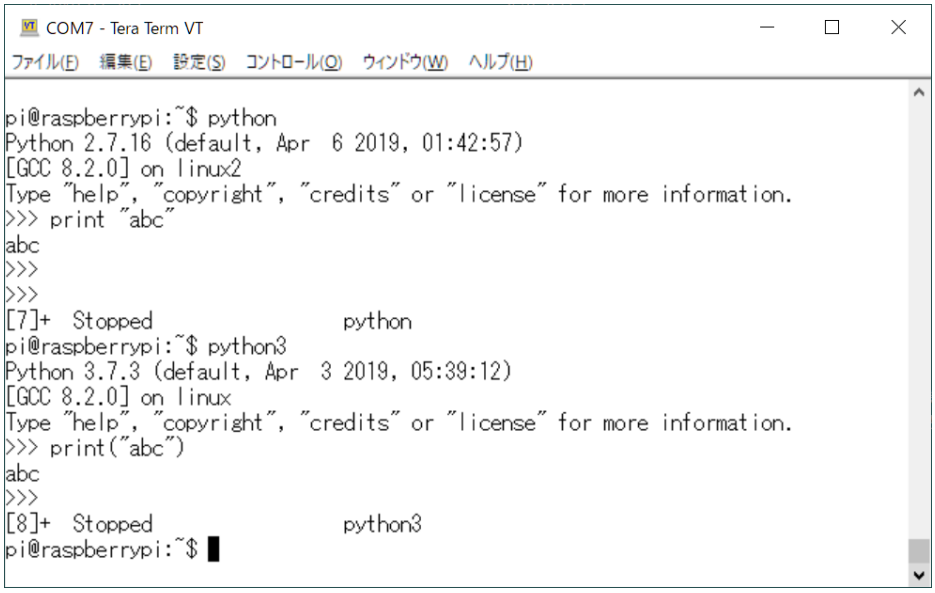


図　RaspberryPi での Python2.x と Python3.x の起動

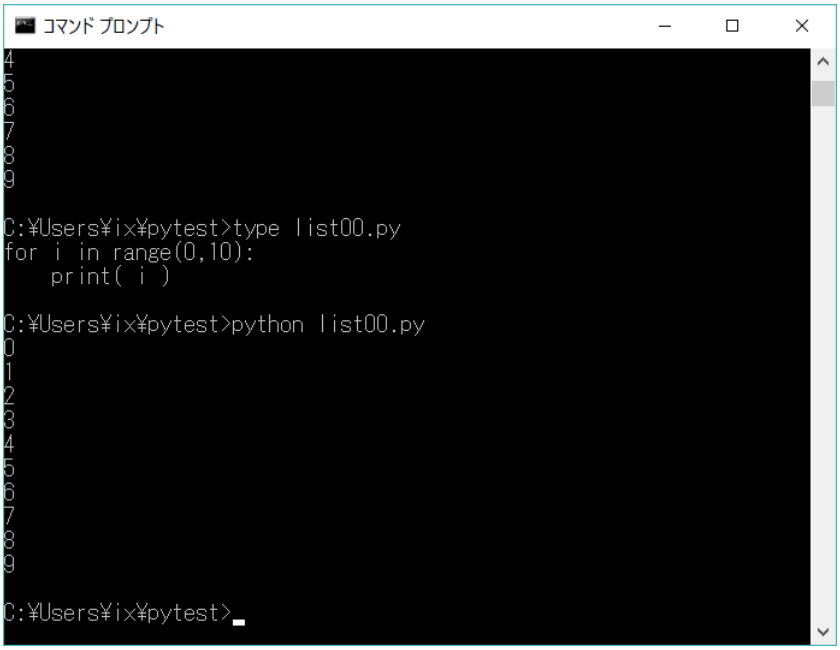


図 9　コマンド窓からの実行の様子

1.3　基本的な Python のプログラム作法

(1)基本的なループプログラム

Python も C 言語等と同様に変数の使用、繰り返しのための構文、があります。C 言語や Java と異なり括弧を使用しません。その分、インデントとコロン ： でブロックを区別します。インデントの深さに意味があるということです。また、変数宣言がいわゆる自動変数は宣言せずに使用できます。

はそれ以下がコメントとなります

リスト１　list00.py

```
# python sample program
for i in range(0,10):
    print( i )
```

〇ループについて

range() はリストを生成する関数なので、大きな回数のループを回すには向きません。

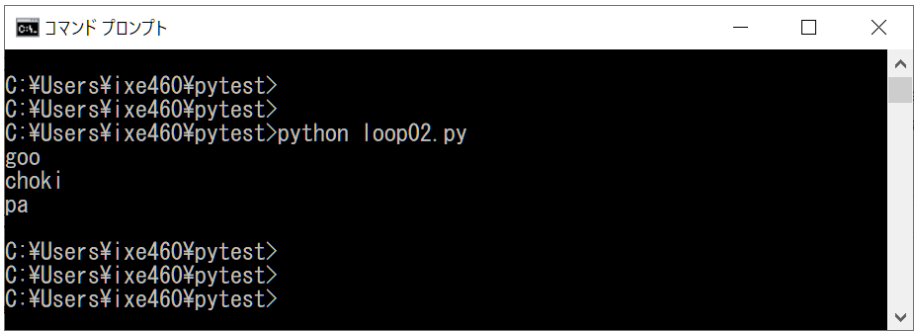
大きな回数のループを回すにはカウント変数と while ループを使いましょう。

```
cnt = 0
while cnt < 1000000 :
    処理
    cnt += 1
```

本来の for の使い方は以下のプログラムのようにリストを取り出すような場合です。

リスト　loop02.py

```
data = ['goo','choki','pa']
for w  in data:
    print( w )
```



図

コラム　python2 と python3 のプログラムの見分け方

web でサンプルプログラムを見つけたときなど、「このプログラムはどっち用？」となったとき、簡単に見分けるには print を見ます。

print の後に()がなければ python2 のプログラムです。()があれば pyhton3 のプログラムです。

100% ではありませんが、かなりの確率で見分けられます。

(2)関数を使う

関数を使用することもできます。def を使って関数の定義を開始します。C 言語のようなプロトタイプ宣言はありません。そのかわり、かならず関数を使用する前に定義しないといけません。関数定義の１行目の終わりにもコロンを使います。

また、引数の指定には順番に値が渡位置引数される「位置引数」の他に渡す先の変数を指定する「キーワード引数」や引数を省略できる「デフォルト引数値」などが利用できます。

リスト２　list02.py

```
def loop(x ,y ):
    for i in range(x,y):
        print( i )
    return i

def loop2(x = 1 ,y = 5):
    for i in range(x,y):
        print( i )
    return i

z = loop(1,5)
print()
print( z )
print()

loop(y = 5, x = 2 )
print()

loop2()
print()

loop2(y = 3)
```

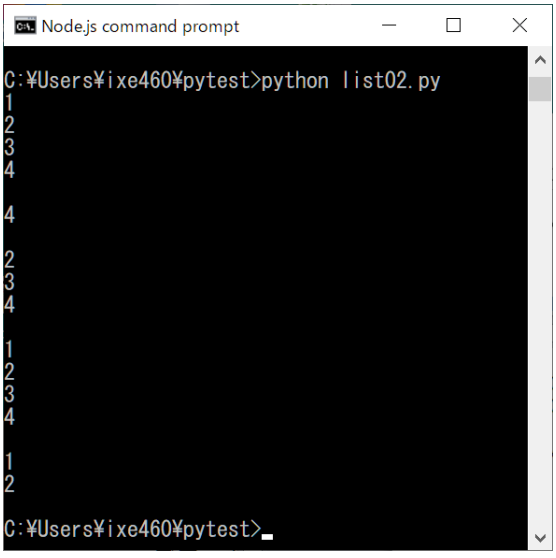


図　list02 の実行結果

⑶ クラスを使う

クラスも定義することができます。ただし、他の言語と同様クラスを使うには多くの項目の利用方法を知っておかないといけませんので、最初のうちはあえてクラスを使わなくてもよいと思いますが、既存のサンプルプログラムを理解するためにはクラスの理解は必要です。

リスト３は　loopclass　というクラスをインスタンスして操作しています。

リスト３　list03.py

```
class loopclass:
    x = 1

    def loop(self):
        for i in range(loopclass.x, self.y):
            print( i , end="")
        print("")

    def sety(self , m):
        self.y = m

z1 = loopclass()
z2 = loopclass()

z1.y = 10
print( z1.x )
print( z1.y )
z1.loop()

z2.sety(5)
print( z2.x )
print( z2.y )
z2.loop()
```

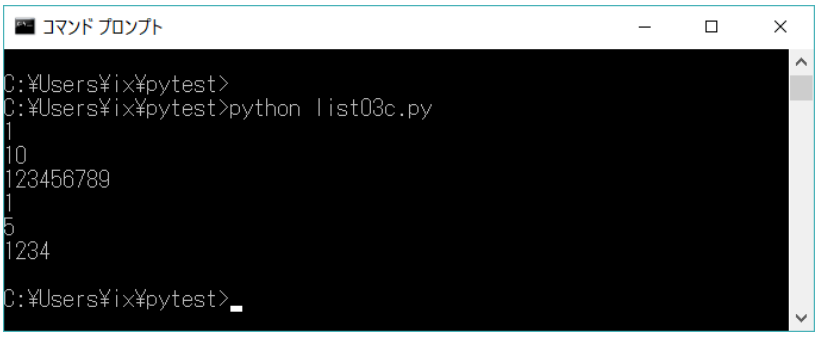


図 1.2.7　クラスを使ったプログラムの実行の様子

コラム　ブロックについて

→ 変則的ですが、１行だけならこんな書き方もできます。

```
for i in range(0,10): print( i )
```

⑷ マルチスレッドを使う

Python などスクリプト系の言語ではマルチスレッドを簡単に構築できます。

リスト４　list04.py

```
import threading
import time

def sub1():
    for n in range(5):
        time.sleep(1)
        print( "sub1" )

def sub2():
    for n in range(5):
        time.sleep(2)
        print( "sub2" )

th1 = threading.Thread( target= sub1 )
th2 = threading.Thread( target= sub2 )
th1.start()
th2.start()
```

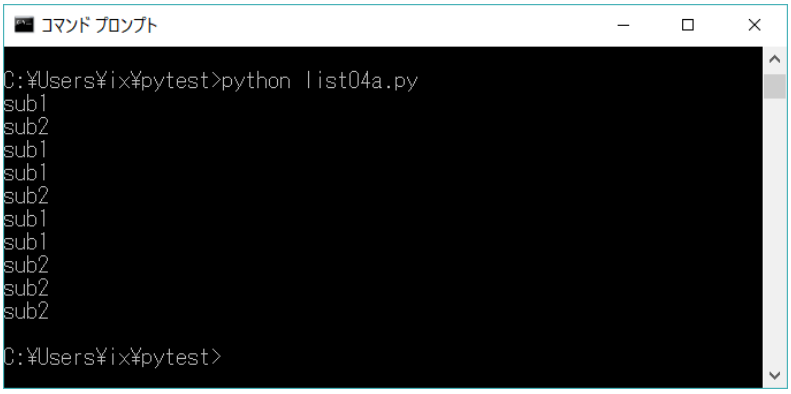


図 1.2.8　マルチスレッドの例

Python まとめ　コメントについて

Python にはブロックコメント(C 言語の /* から */ まで)が存在しません。

Python には処理中に現れた文字列は実行に影響しないという仕様を利用して、ブロック文字列※をコメントとして利用します。

この場合、前後の処理のインデントに合わせないとエラーになりますので、注意が必要です。

※ ブロック文字列：正式な呼び方かどうか不明。ダブルクォートかシングルクォートを三つ重ねたもので挟まれた文字列。

また。C 言語のプリプロセッサは存在しません。

通常の変数として記述したり、ブロック文字列としてコメントアウトしたりします。

#if 0 ～ #endif
↓
if false　あるいは　''' ～ ''' で囲む

Python まとめ　変数について

・変数名

1 文字目は英字かアンダーバー

2 文字目以降は英数字かアンダーバー

大文字と小文字は区別される

アンダーバーで始まるシンボルは特殊な扱いの場合があるので注意。

・変数について

変数自体に型情報は紐づいていません。

`a = 1.0` の後、 `a = "string"` とか `a = [1,2,3]` とかにしてもエラーにはなりません。

・変数のスコープについて

グローバル変数とローカル変数(関数内のみ有効)があります。ブロックスコープはありません。
関数内からグローバル変数を参照(読み取り)することはできますが変更(書き込み)することはできません。

→ グローバル変数を変更したい場合は、global 変数名 で宣言してからアクセスします。
global 宣言しないで書き込むと、新しいローカル変数を作成します。

このほかに import したモジュール内で有効なモジュールスコープ、
クラスで使用するクラス変数、インスタンス変数があります。

変数を使用する際、変数宣言する必要はありません。いきなり `x = 1` のように記述すれば
変数 x が定義されます。
なので、代入の際、間違って記述してもエラーになりません。

Python まとめ　関数について

・return 文がない場合は、None が返り値となる。

・戻り値、引数ともに型は指定されない。必要なら処理中に型チェック。

・関数定義時、デフォルト引数で呼び出し時に指定しなかったときの値を指定できる。

def func(a, b=1)：　→　func(10)　func(10,1)と同じ

func() とするとエラー(a にデフォルト値を設定してないので)

・関数呼び出し時、引数にキーワードを指定することで任意の順序で引数を指定できる

def func(a, b)：　→ func(b=2, a=1)　func(1, 2) と同じ

宣言された順序通りに設定する場合はキーワードは不要。

デフォルト引数と組み合わせて使うと便利。

def func(a, b=1, c=3)：　b はデフォルトのまま、c だけ変更したいとき、

func(1, c=10) のように指定する