■Pythonの基礎構文

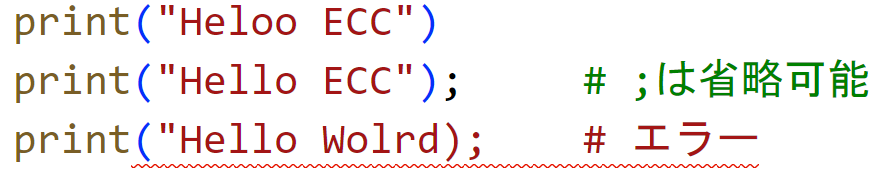
ターミナルに出力

◆Point!

文字列は’(シングルコーテーション)で囲う必要がある！

Pythonにおいて;(セミコロン)は不要！

print(値)



◆Point!

**SyntaxError：構文エラー文法ミス、実行しない**

**Exception:実行時に発生したエラー、実行は中止**

**(エラー)SyntaxError:** EOL while scanning string literal

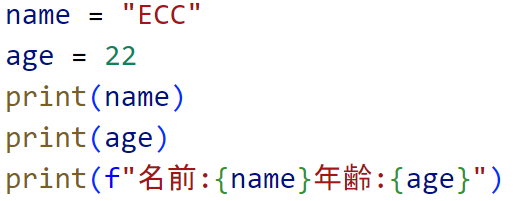
◆1.変数

Pythonではvariableという仕組みが準備されています。

Javaでは変数宣言にデータ型の指定が必要でしたが、

Pythonでは不要となります。

(※便利な反面、どのデータ型が格納されているのかわかりにくくなります。)



◆Point!

**変数は複数を同時に代入することが可能です**

**例)**

**x,y,z = 1,2,3**

◆Point!

**変数は原則使いまわさない！**

×例：num = 1

num = ‘1’

print(num \* 2) ←エラー

文字列の変数を使ったprint文

フォーマットを活用し、変数と文字列を便利に活用

print (f”文字列 { 変数 }”)

文字列の中に{}中括弧を記述し、その中に変数を入れると連結などせずに使用できます。

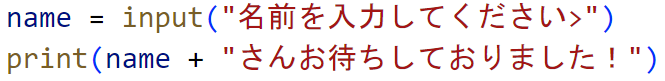
◆キーボード値の代入

input(表示文字列)

◆落とし穴Point!

inputで取得した値は**文字列**として扱われます！

数値として扱いたい場合などは**変換して使用**する必要があります



格納されているデータ型の確認をする場合はtype(値)で判定できます。

◆データ型の変換

|  |  |
| --- | --- |
| int型への変換: int(値) | float型への変換: float(値) |
|  |  |
| str型への変換: str(値) | bool型への変換: bool(値) |
|  |  |

◆Point!

**bool型の変換時**

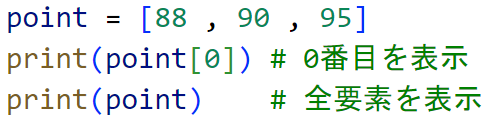
**0はfalse , それ以外はtrue**

**’’(空文字)はfalse , それ以外はtrue**

■2.コレクション

変数の塊(リスト)も宣言無しで作成が可能です。

変数 = [ 要素1 , 要素2 , 要素3 ….]



◇便利な関数

|  |  |
| --- | --- |
| sum(リスト) : リストの合計値を取得 | len(リスト) : リストの要素数を取得 |

◆リストの要素の追加削除

|  |
| --- |
| リストの追加  リスト.append(**追加**したい値) # リストの末尾に値を追加 |
|  |
| リストの削除  リスト.remove(**削除**したい値) # リスト内の指定した値を削除 |
| ◆Point!  **削除する場合は、**  **見つけた最初の値を削除しています** |

◆Point!

**Pythonではifやwhileなどで{　}を記述しません！**

**代わりに「インデントしている部分」が「ブロック」として扱われます**

**始まりは「:(コロン)」で認識し**

**終わりは「インデントされていない部分」になります**

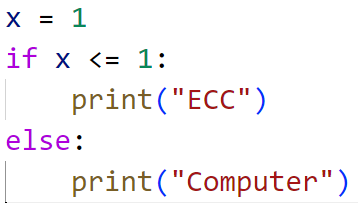
■3.条件分岐if

if 条件式:

処理1

else:

処理2



◆Point!

**Pythonでの引数は引数名を指定することも出来ます、**

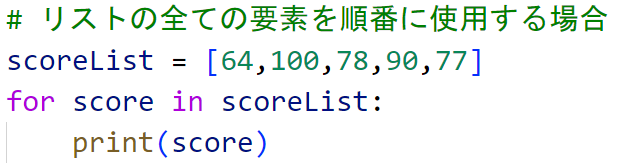
**これにより引数の順番を無視することも可能です**

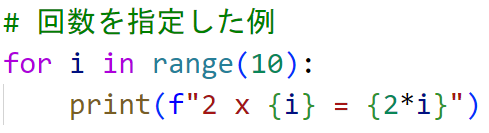
**見つかった先頭の値を削除しています**

■4.繰返し処理for

for カウント変数 in range(回数): for 要素を入れる変数 in リスト:

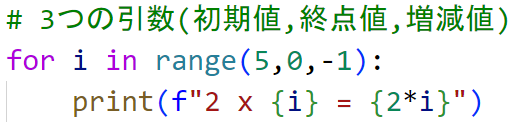
繰り返す処理 繰り返す処理





for カウント変数 in range(初期値,終点値,増減値):

繰り返す処理

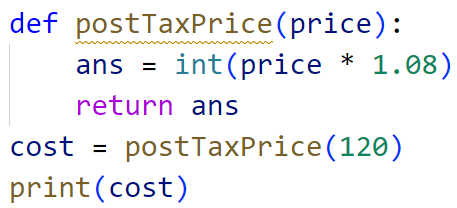


■5.関数

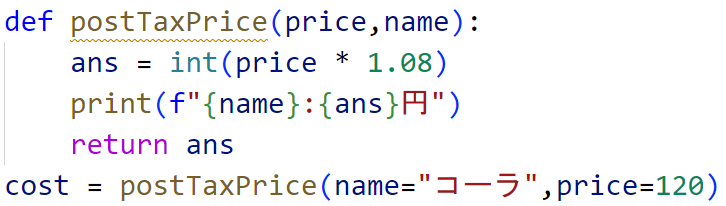
def 関数名(引数1,引数2,…):

処理

return 戻り値



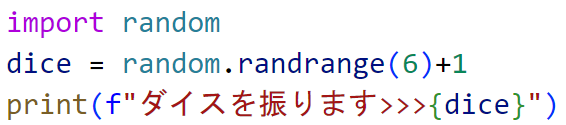
※補足：引数が複数ある場合



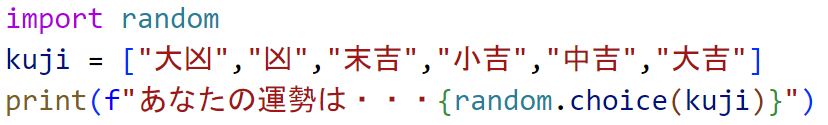
引数を指定することが可能

Pythonでプログラミング

■サイコロを作ってみよう



■おみくじを作ってみよう



■6.オブジェクト

Pythonではあらゆるデータをオブジェクトとして扱います。

例えばname = “ECC” という変数を作成した場合

nameはstr型のオブジェクトとして扱い、

str.format()やstr.strip()など関数もオブジェクトに含まれる要素になります。

­­

では、値(オブジェクト)がどのデータ型として生成されるのか

それは生成時に指示をしています

|  |  |
| --- | --- |
| 小数点を含まない数字 | intクラスのオブジェクト |
| 小数点を含む数字 | floatクラスのオブジェクト |
| ‘や”で囲まれた文字 | strクラスのオブジェクト |
| [　]で囲まれた文字 | listクラスのオブジェクト |
| { }で囲まれた文字 | {～:～}形式ならdictクラス、  {～}ならsetクラスのオブジェクト |

つまり、オブジェクトは参照しているだけ

**■7.モジュール**

モジュールとは開発されたライブラリのことを指す

|  |
| --- |
| 標準ライブラリ |
| Pythonが公式に用意したモジュールのまとまり |
| 外部ライブラリ |
| 別の組織や個人が用意したモジュールのまとまり |

明示的に宣言することにより、

他の開発者が作成した変数や関数をプログラムに取り入れる事が可能となる。

標準ライブラリ(代表例)

|  |  |
| --- | --- |
| モジュール | 用途 |
| mathモジュール | 数学計算に関する処理 |
| randomモジュール | 乱数に関する処理 |
| datetimeモジュール | 日付と時間に関する |
| emailモジュール | 電子メールに関する処理 |
| csvモジュール | CSVファイルに関する処理 |
| jsonモジュール | JSONファイルに関する処理 |
| osモジュール | OS操作に関する処理 |

外部ライブラリ(代表例)

|  |  |
| --- | --- |
| モジュール名(パッケージ名) | 主な用途 |
| matplotlib | データの可視化 |
| pandas | データ解析 |
| numpy | ベクトル・行列計算 |
| sciPy | 科学技術計算 |
| symPy | 代数計算 |
| scikit-lean | 機械学習 |
| tensorFlow | 深層学習 |
| pygame | グラフィックス、音声 |
| dateutil | 日付、時間 |
| simplejson | JSONファイル |
| pyYAML | YAMLファイル |
| requests | Webアクセス |

**※Pythonは出来ることがとても多いです。**

**そして外部ライブラリは学習コストが高い傾向にあります。**

**その為、基本的に「実現したい目的を明確」にしてから、外部ライブラリを学びましょう。**

**やみくもに勉強しても、「掛けた時間を無駄にする」可能性が高いです。**

■次回以降の予告

Python超入門では有名な外部ライブラリとその使い方を一部だけ紹介します。

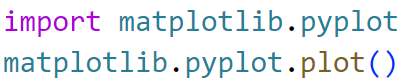
モジュールの取り込み

import モジュール名

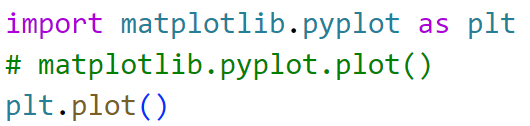


モジュール内の関数を呼び出す

モジュール名.関数名(引数)



モジュールに別名を付けて取り込む



外部ライブラリ　matplotlib　データの可視化に関数を提供

