Python tips: print()の出力先を変える方法

https://ja.stackoverflow.com/questions/42919/python3-%E6%A8%99%E6%BA%96%E9%96%A2%E6%95%B0%E3%81%AB%E5%89%B2%E3%82%8A%E8%BE%BC%E3%81%BF%E5%87%A6%E7%90%86%E3%82%92%E5%85%A5%E3%82%8C%E3%81%9F%E3%81%84

builtinsモジュールに組み込み関数が入っているので、置き換える
ただし、import builtins を行ってbuiltins.printを置き換えたファイルにだけ有効

```
import builtins

# もとの組込みprint関数の値を保存しておく
print_original = print

# 自前のprint用関数を定義する
def my_print(*args, **kwargs):
 original _print("# カスタム版printが呼ばれました")     # 追加したい処理
    return orig_ original(*args , **kwargs)                    # 元のprintの呼び出し

# 組込み識別子のprintをmy_printに変える
builtins.print = my_print

print("hello, world")
```

Matplitlib グラフの特殊動作

```
# グラフ中のデータ点をマウスでクリックしたときの動作

# クリック点に一番近いデータを探す
    def find_nearest_data(x, y, xlist, ylist):
        return ihit, minr2
# クリック時の応答関数 (callback)
    def onclick(event):
        xe, ye  = event.xdata, event.ydata  # event.xdata, .ydataに
                                            # クリックした位置のデータ値が入っている

# matplotlibのfigオブジェクトと、tkinterのイベントを結合 (bind)
    fig.canvas.mpl_connect("button_press_event", onclick)
```

scipy.optimize.minimize()で使えるアルゴリズム

[tkProg]¥tkprog_tutorial¥optimize¥minimize_tutorial2.py

#nelder-mead　Downhill simplex
#powell　　Modified Powell
#cg　　conjugate gradient (Polak-Ribiere method)
#bfgs　　BFGS法
#newton-cg　　Newton-CG
#trust-ncg　　信頼領域 Newton-CG 法
#dogleg　　信頼領域 dog-leg 法
#L-BFGS-B’ (see here)
#TNC’ (see here)
#COBYLA’ (see here)
#SLSQP’ (see here)
#trust-constr’(see here)
#dogleg’ (see here)
#trust-exact’ (see here)
#trust-krylov’ (see here)

Program: peakfit-scipy-minimize.py

Callback関数の使い方

```
# callback関数: ある関数から呼び出される関数
# scipy.minimize()のcallback関数では、更新されたパラメータのリストが引数 xk として渡される
# 他の引数は渡されないので、global変数にするか、callbackの指定でlambda関数を定義する
iter = 0
xiter  = []
yfmin  = []
def callback(xk):
# global変数を変更する場合は、global宣言が必要。そうでないと、local変数として扱われる
    global iter, xiter, yfmin

    fmin = CalS2(xk)
    print("callback  {}: xk={}".format(iter, xk))
    print("    fmin={}".format(fmin))
    iter += 1
    xiter.append(iter)
    yfmin.append(fmin)


def main():
…
    res = minimize(CalS2, x0, jac = diff1, method = method, tol = tol, callback = callback,
            options = {'maxiter':maxiter, "disp":True})        # 初期値を与えて最適化。method = "cg"で共役購買法を使っている
```

Python tips: scipy.minimize()のcallback

Callback関数で繰り返し回数などを表示する方法

例: [tkProg]¥tkprog_tutorial¥optimize¥minimize_func1d.py

- **Global変数を使う**

```
iter = 0
def callback(xk):
    global iter                                    # 関数内でglobal変数を書き換える場合には global宣言が必要

    fmin = minimize_func(xk)
    print(f"callback {iter}: xk={xk}  func={fmin}")
    iter += 1

res = minimize(minimize_func, x0s, jac = diff1, method = method, tol = tol,
               callback = callback,
               options = {'maxiter':maxiter, "disp":True})
```

- **classを使う。__call__ を使うと、クラスのインスタンスを関数として呼び出せる**

```
class call_back():
    def __init__(self):
        self.iter = 0
    def __call__(self, xk):
        fmin = minimize_func(xk)
        print(f"callback {self.iter}: xk={xk}  func={fmin}")
        self.iter += 1

cb = call_back()
res = minimize(minimize_func, x0s, jac = diff1, method = method, tol = tol,
               callback = cb,                                  # callbackとして cb(x) が呼び出される
               options = {'maxiter':maxiter, "disp":True})
```

Python tips: Early Stopさせる方法

scikit-optimizeの場合: EarlyStopperで最適化を中断する
https://qiita.com/saiaron/items/8bf6cdf411b48a5ea59e

scipy.minimize() では、callback() が False を返したら Early Stopする

1. 割と簡単でお薦め: グラフウィンドウを閉じて終了させる
   例: [tkProg]¥tkprog_tutorial¥optimize¥lsq_func.py
   (1) 最初にグラフを表示したときにウインドウハンドルを保存
   callback.window = plt.get_current_fig_manager().window
   (2) callback()でウインドウハンドルを取得し、(1)と比較。
   異なっていたら False を返して Early Stopさせる

2. ちょっと複雑だけど正攻法: グラフウィンドウにボタンを表示、ボタンを押して終了させる
   https://note.com/evjunior/n/n73c43c1b546f などを参考にボタンとクリックイベント関数を設定
   参考: [tkProg]¥tkprog_tutorial¥optimize¥lsq_func.py で 'button' と 'subplots_adjust' を検索
   (1) ボタンを押したら stop_flag をTrueにする
   (2) callback()が呼ばれたら、stop_flag が Trueなら False を返して Early Stopさせる
   (3) tight_layout() を呼んだら subplots_adjust() を再設定する

3. ファイル検出: 面倒くさい
   callbackで os.path.exists('STOP') を実行し、STOPファイルがあったら中断
4. キーボード入力: 間違ってキー入力する可能性。入力するキーを覚えていないといけない
   keyboardモジュール keyboard.read_key()
   https://kuku81kuku81.hatenablog.com/entry/2022/06/16_python_keyboard_AvailableKeyTypes
5. 割り込み (Ctrl+C): scipyと共存しない
   https://kusanohitoshi.blogspot.com/2017/01/pythonctrl-c.html

scikit-learnで使える回帰アルゴリズム

線形回帰系:

多重線形回帰

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

多項式回帰

```
from sklearn.preprocessing import PolynomialFeatures
model = PolynomialFeatures(degree = 2, interaction_only = False, include_bias = True,  order = 'C’)
```

Ridge回帰

```
from sklearn.linear_model import Ridge
model = Ridge(alpha = 0.05)
```

LASSO回帰

```
from sklearn.linear_model import Lasso
model = Lasso(alpha = 0.05)
```

Elastic Net回帰

```
from sklearn.linear_model import ElasticNet
model = ElasticNet(alpha = 0.05)
```

ノンパラメトリック系:

Kernal Ridge回帰

```
from sklearn.kernel_ridge import KernelRidge
model = KernelRidge(alpha = 1.0, kernel = 'rbf’)
```

Gauss過程回帰

```
from sklearn.gaussian_process import GaussianProcessRegressor
model = GaussianProcessRegressor()
```

多層パーセプトロン (多層ニューラルネットワーク)

```
from sklearn.neural_network import MLPClassifier
model = MLPClassifier(max_iter = 1000, hidden_layer_sizes = (10,), activation = 'logistic', solver = 'sgd’,
                      learning_rate_init = 0.01)
```

scikit-learn: 回帰

分類器系:

Random Forest回帰

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
```

勾配ブースティング木 回帰

```
from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor()
```

サポートベクターマシーン 回帰

```
from sklearn.svm import SVR
model = SVR(kernel='linear', C=1, epsilon=0.1, gamma='auto’)
```

scikit-learnで使える回帰アルゴリズム (Web検索より)

```
# sklearnの回帰モデルを片っ端から試す
# https://qiita.com/futakuchi0117/items/72ce4afae9adcccd6e18

from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet, SGDRegressor
from sklearn.linear_model import PassiveAggressiveRegressor, ARDRegression, RidgeCV
from sklearn.linear_model import TheilSenRegressor, RANSACRegressor, HuberRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import SVR, LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, ExtraTreesRegressor, HistGradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor, GradientBoostingRegressor, VotingRegressor, StackingRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.cross_decomposition import PLSRegression

reg_dict = {"LinearRegression": LinearRegression(),
            "Ridge": Ridge(),
            "Lasso": Lasso(),
            "ElasticNet": ElasticNet(),
            "Polynomial_deg2": Pipeline([('poly', PolynomialFeatures(degree=2)),('linear', LinearRegression())]),
            "Polynomial_deg3": Pipeline([('poly', PolynomialFeatures(degree=3)),('linear', LinearRegression())]),
            "Polynomial_deg4": Pipeline([('poly', PolynomialFeatures(degree=4)),('linear', LinearRegression())]),
            "Polynomial_deg5": Pipeline([('poly', PolynomialFeatures(degree=5)),('linear', LinearRegression())]),
            "KNeighborsRegressor": KNeighborsRegressor(n_neighbors=3),
            "DecisionTreeRegressor": DecisionTreeRegressor(),
            "RandomForestRegressor": RandomForestRegressor(),
            "SVR": SVR(kernel='rbf', C=1e3, gamma=0.1, epsilon=0.1),
            "GaussianProcessRegressor": GaussianProcessRegressor(),
            "SGDRegressor": SGDRegressor(),
            "MLPRegressor": MLPRegressor(hidden_layer_sizes=(10,10), max_iter=100, early_stopping=True, n_iter_no_change=5),
            "ExtraTreesRegressor": ExtraTreesRegressor(n_estimators=100),
            "PLSRegression": PLSRegression(n_components=10),
            "PassiveAggressiveRegressor": PassiveAggressiveRegressor(max_iter=100, tol=1e-3),
            "TheilSenRegressor": TheilSenRegressor(random_state=0),
            "RANSACRegressor": RANSACRegressor(random_state=0),
            "HistGradientBoostingRegressor": HistGradientBoostingRegressor(),
            "AdaBoostRegressor": AdaBoostRegressor(random_state=0, n_estimators=100),
            "BaggingRegressor": BaggingRegressor(base_estimator=SVR(), n_estimators=10),
            "GradientBoostingRegressor": GradientBoostingRegressor(random_state=0),
            "VotingRegressor": VotingRegressor([('lr', LinearRegression()), ('rf', RandomForestRegressor(n_estimators=10))]),
            "StackingRegressor": StackingRegressor(estimators=[('lr', RidgeCV()), ('svr', LinearSVR())], final_estimator=RandomForestRegressor(n_estimators=10)),
            "ARDRegression": ARDRegression(),
            "HuberRegressor": HuberRegressor(),
                }
```