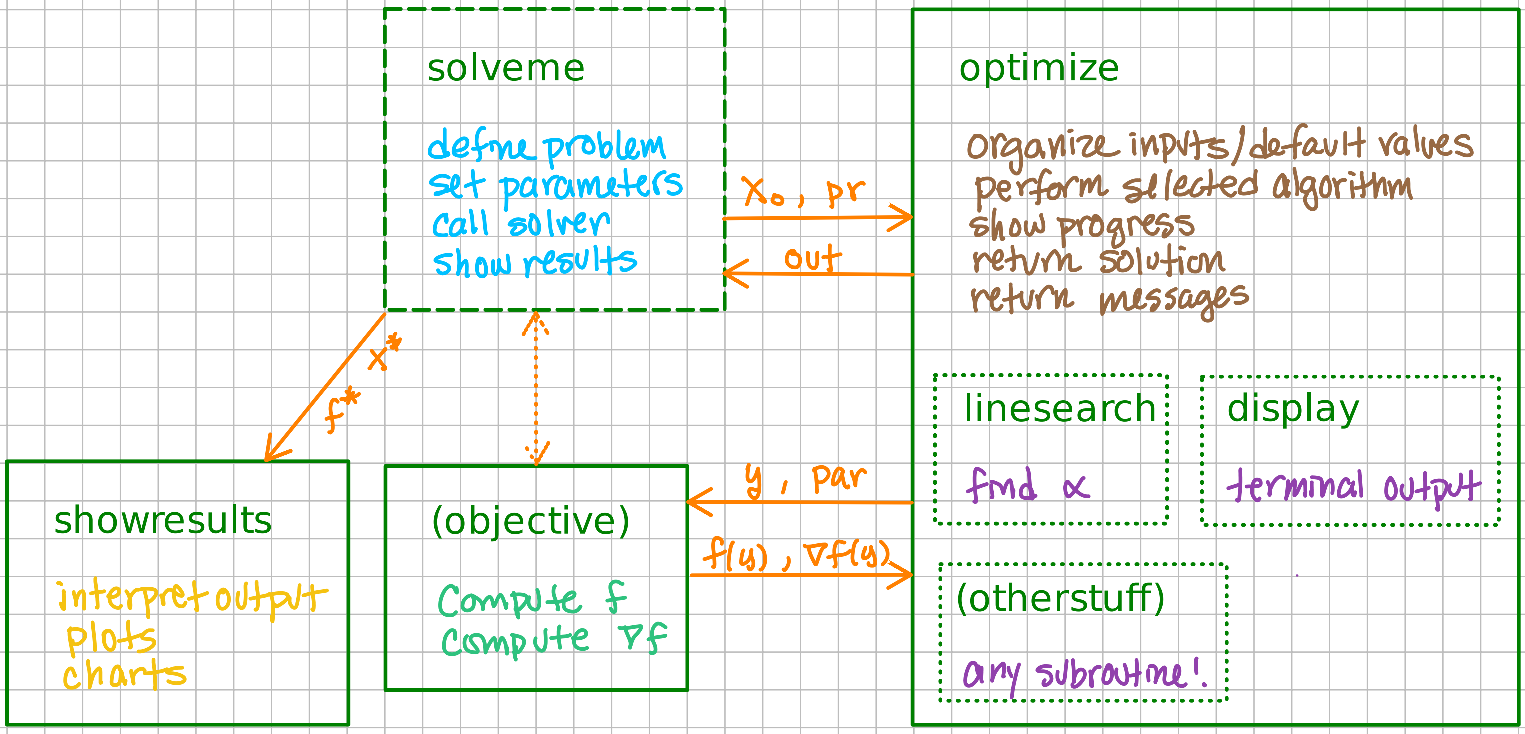
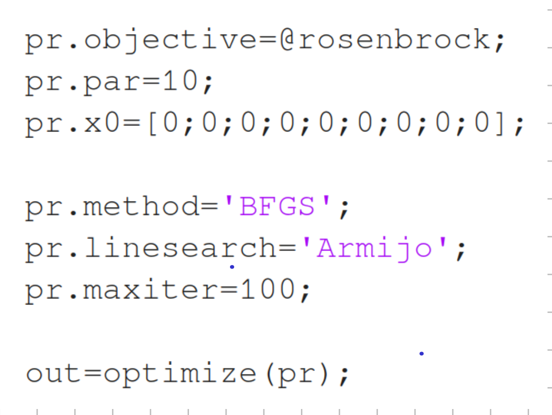
Suggestions on How to Organize your Code



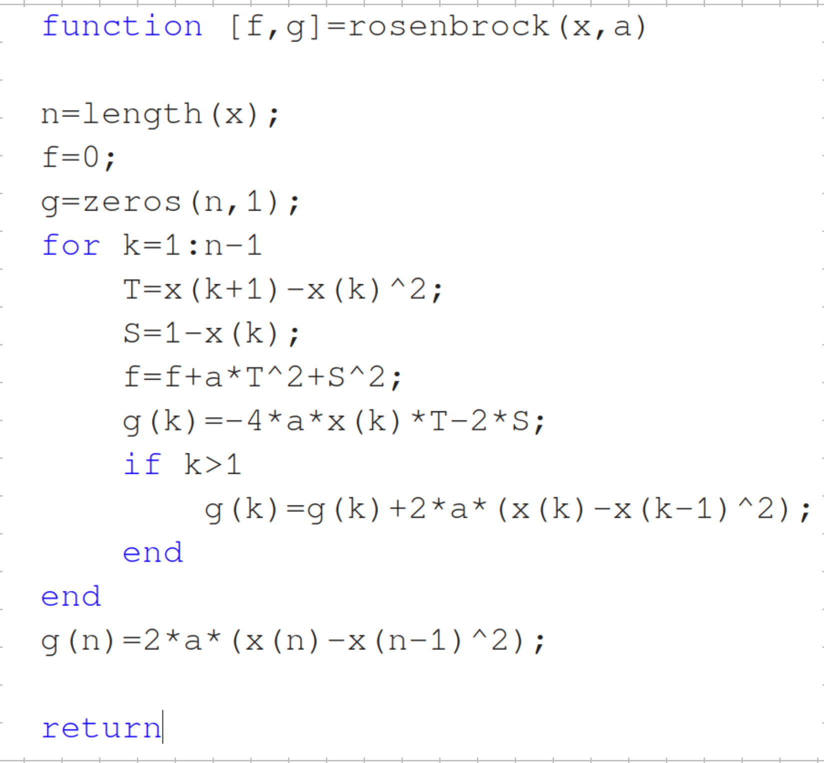
□	function with inputs/outputs or python module
⬚	script file set of commands
⋯	subfunctions
x_0	initial vector in $\mathbb{R}^n$
pr	problem definition variable (structure in matlab, dictionary in python)
out	problem results variable (structure or dictionary)
par	Parameter variable required by objective (structure or dictionary)
y	test point in $\mathbb{R}^n$
$f(y)$	objective value at y
$\nabla f(y)$	gradient vector at y

solveme.m



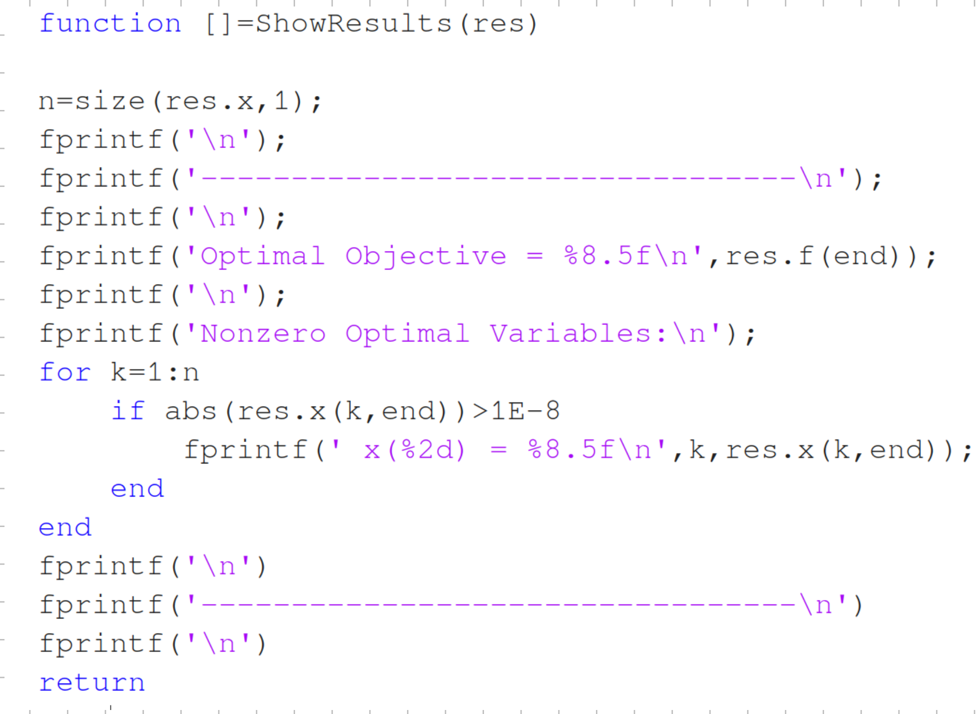
```
pr.objective=@rosenbrock;
pr.par=10;
pr.x0=[0;0;0;0;0;0;0;0;0];

pr.method='BFGS';
pr.linesearch='Armijo';
pr.maxiter=100;

out=optimize(pr);
```

rosenbrock.m



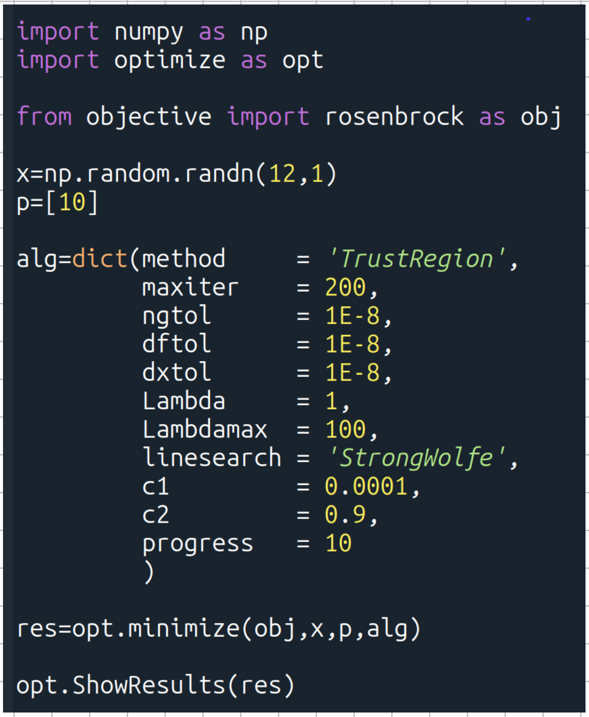
```
function [f,g]=rosenbrock(x,a)

n=length(x);
f=0;
g=zeros(n,1);
for k=1:n-1
    T=x(k+1)-x(k)^2;
    S=1-x(k);
    f=f+a*T^2+S^2;
    g(k)=-4*a*x(k)*T-2*S;
    if k>1
        g(k)=g(k)+2*a*(x(k)-x(k-1)^2);
    end
end
g(n)=2*a*(x(n)-x(n-1)^2);

return
```

ShowResults.m



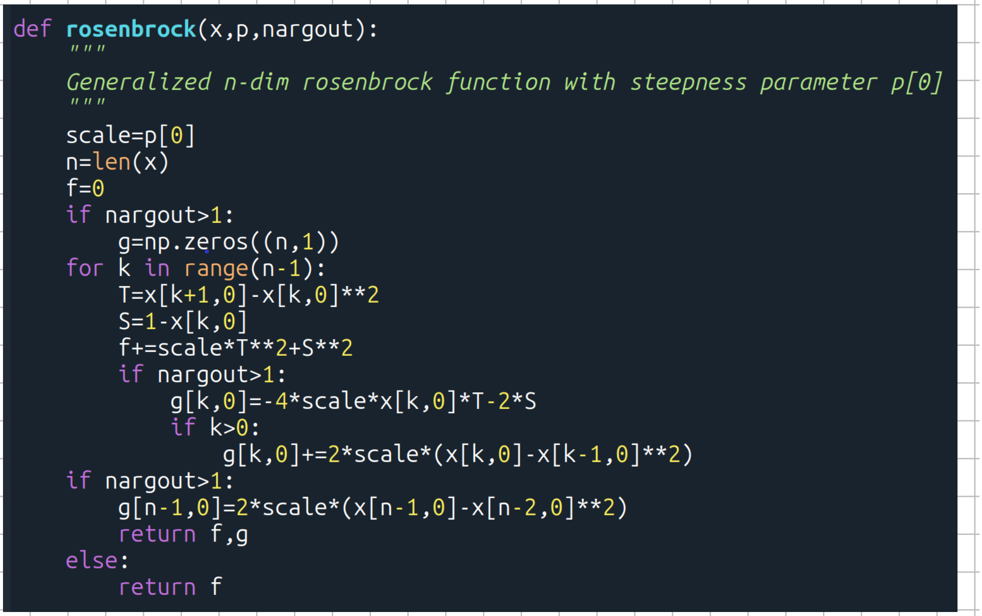
```
function []=ShowResults(res)

n=size(res.x,1);
fprintf('\n');
fprintf('---------------------------------\n');
fprintf('\n');
fprintf('Optimal Objective = %8.5f\n',res.f(end));
fprintf('\n');
fprintf('Nonzero Optimal Variables:\n');
for k=1:n
    if abs(res.x(k,end))>1E-8
        fprintf(' x(%2d) = %8.5f\n',k,res.x(k,end));
    end
end
fprintf('\n')
fprintf('---------------------------------\n')
fprintf('\n')
return
```

Solveme.py



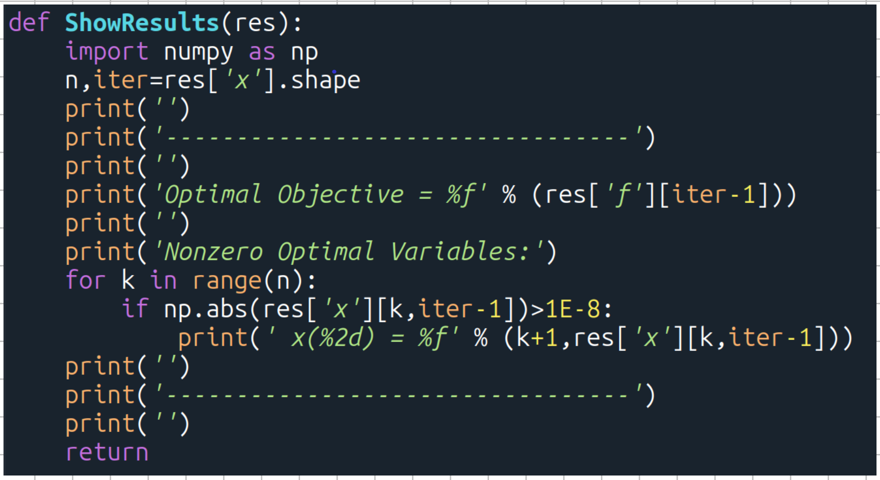
```
import numpy as np
import optimize as opt

from objective import rosenbrock as obj

x=np.random.randn(12,1)
p=[10]

alg=dict(method     = 'TrustRegion',
         maxiter    = 200,
         ngtol      = 1E-8,
         dftol      = 1E-8,
         dxtol      = 1E-8,
         Lambda     = 1,
         Lambdamax  = 100,
         linesearch = 'StrongWolfe',
         c1         = 0.0001,
         c2         = 0.9,
         progress   = 10
         )

res=opt.minimize(obj,x,p,alg)

opt.ShowResults(res)
```

function rosenbrock in module objective



```
def rosenbrock(x,p,nargout):
    """
    Generalized n-dim rosenbrock function with steepness parameter p[0]
    """
    scale=p[0]
    n=len(x)
    f=0
    if nargout>1:
        g=np.zeros((n,1))
    for k in range(n-1):
        T=x[k+1,0]-x[k,0]**2
        S=1-x[k,0]
        f+=scale*T**2+S**2
        if nargout>1:
            g[k,0]=-4*scale*x[k,0]*T-2*S
            if k>0:
                g[k,0]+=2*scale*(x[k,0]-x[k-1,0]**2)
    if nargout>1:
        g[n-1,0]=2*scale*(x[n-1,0]-x[n-2,0]**2)
        return f,g
    else:
        return f
```

function ShowResults in module optimize



```
def ShowResults(res):
    import numpy as np
    n,iter=res['x'].shape
    print('')
    print('---------------------------------')
    print('')
    print('Optimal Objective = %f' % (res['f'][iter-1]))
    print('')
    print('Nonzero Optimal Variables:')
    for k in range(n):
        if np.abs(res['x'][k,iter-1])>1E-8:
            print(' x(%2d) = %f' % (k+1,res['x'][k,iter-1]))
    print('')
    print('---------------------------------')
    print('')
    return
```

Suggested Methods for reading .csv files

suppose the data file is an array of numerical valves

Matlab:

```
A = readmatrix('datafile.csv');   % read in as matrix A
col1 = A(:,1);          % maybe the first column is later used for something
row3 = A(3,:);          % or perhaps the third row.
```

Python:

```
import pandas as pd
A= pd.read_csv('datafile.csv').to_numpy()
col1= A[:,[0,]]
row3 = A[[2,],:]
```