LAB 4

Welcome!😊

Outline

Constrained Optimization

- Toy Problem
- Water Filling Problem
- Entropy Maximization Problem

Toy Problem

Toy Problem: Formulation

$$\min x_2$$

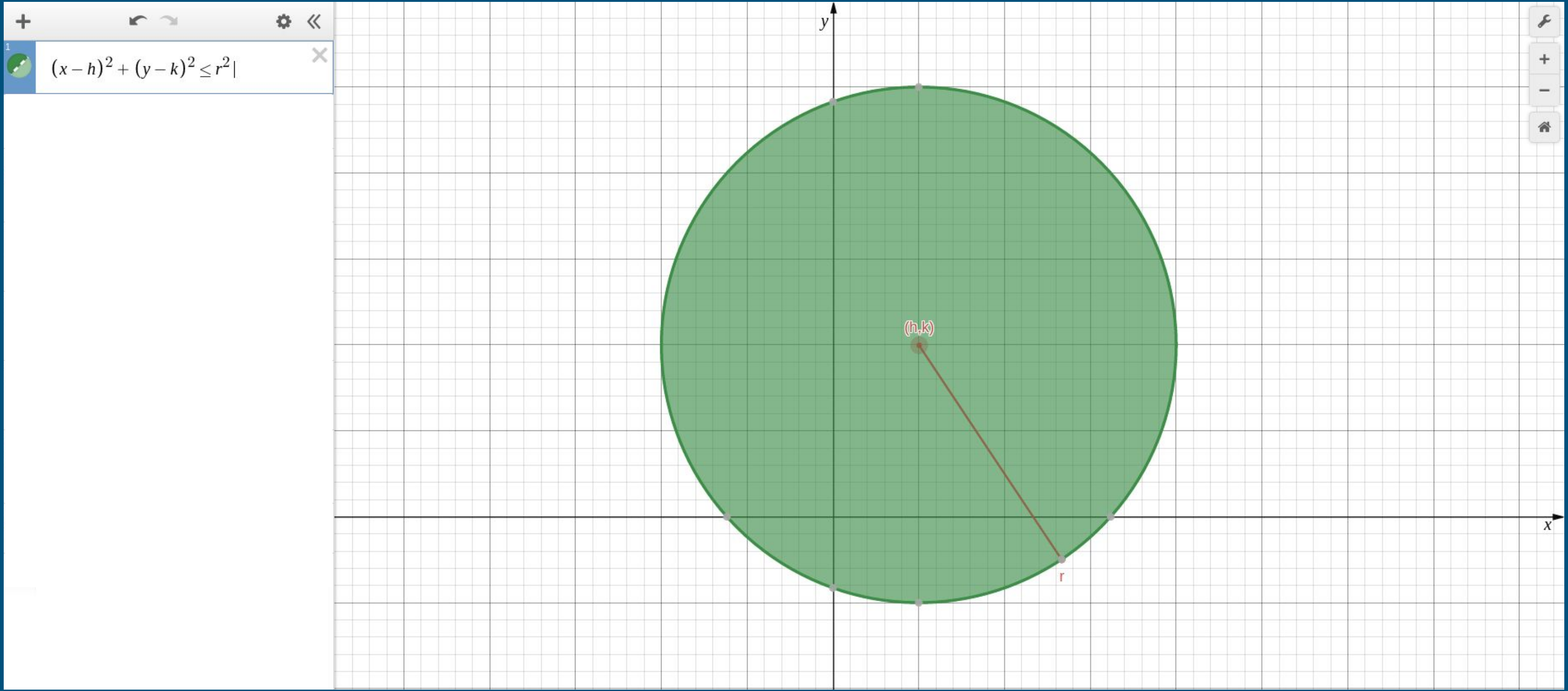
$$\text{subject to} \quad 9 \geq (x_1 - 1)^2 + (x_2 - 2)^2$$

$$x_2 \leq 0$$

Toy Problem: Formulation

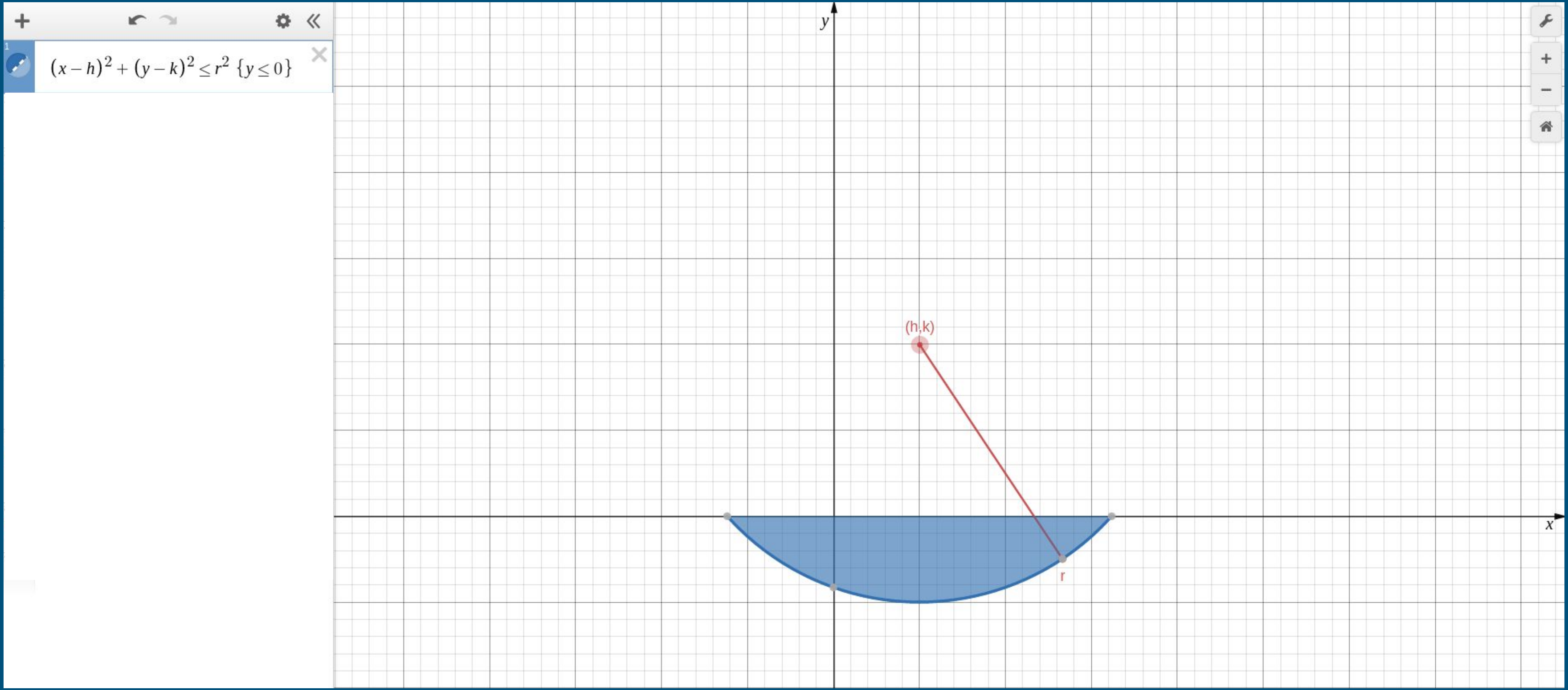
$$\begin{aligned} &\min x_2 \\ &\text{subject to} \quad (x_1 - 1)^2 + (x_2 - 2)^2 \leq 9 \\ &\phantom{\text{subject to} \quad} x_2 \leq 0 \end{aligned}$$

Toy Problem: Visualization


$(x-h)^2+(y-k)^2 \le r^2$

Source: Desmos

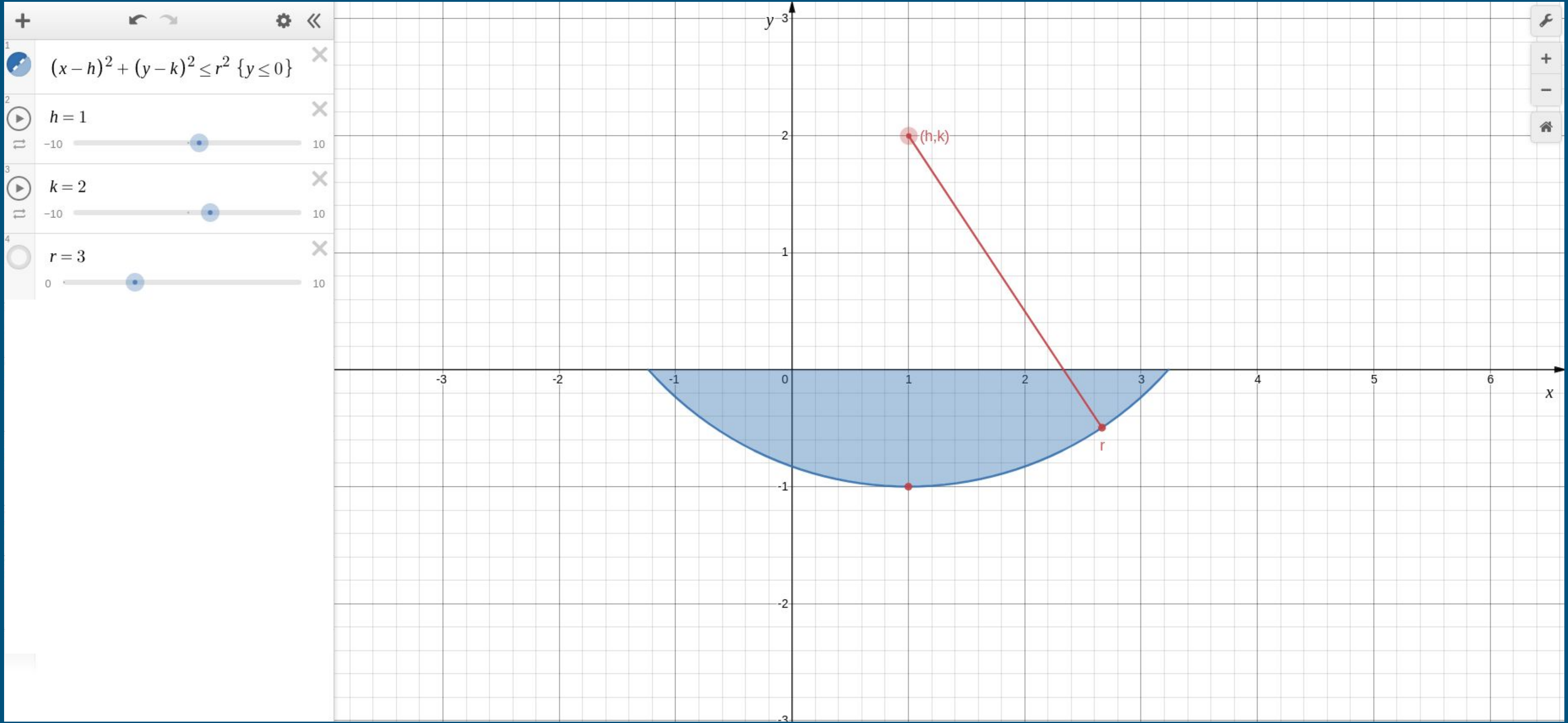
Toy Problem: Visualization



$(x-h)^2+(y-k)^2 \le r^2 \ \{y \le 0\}$

Source: Desmos

Toy Problem: Sketch the solution?



$(x-h)^2 + (y-k)^2 \le r^2 \; \{y \le 0\}$

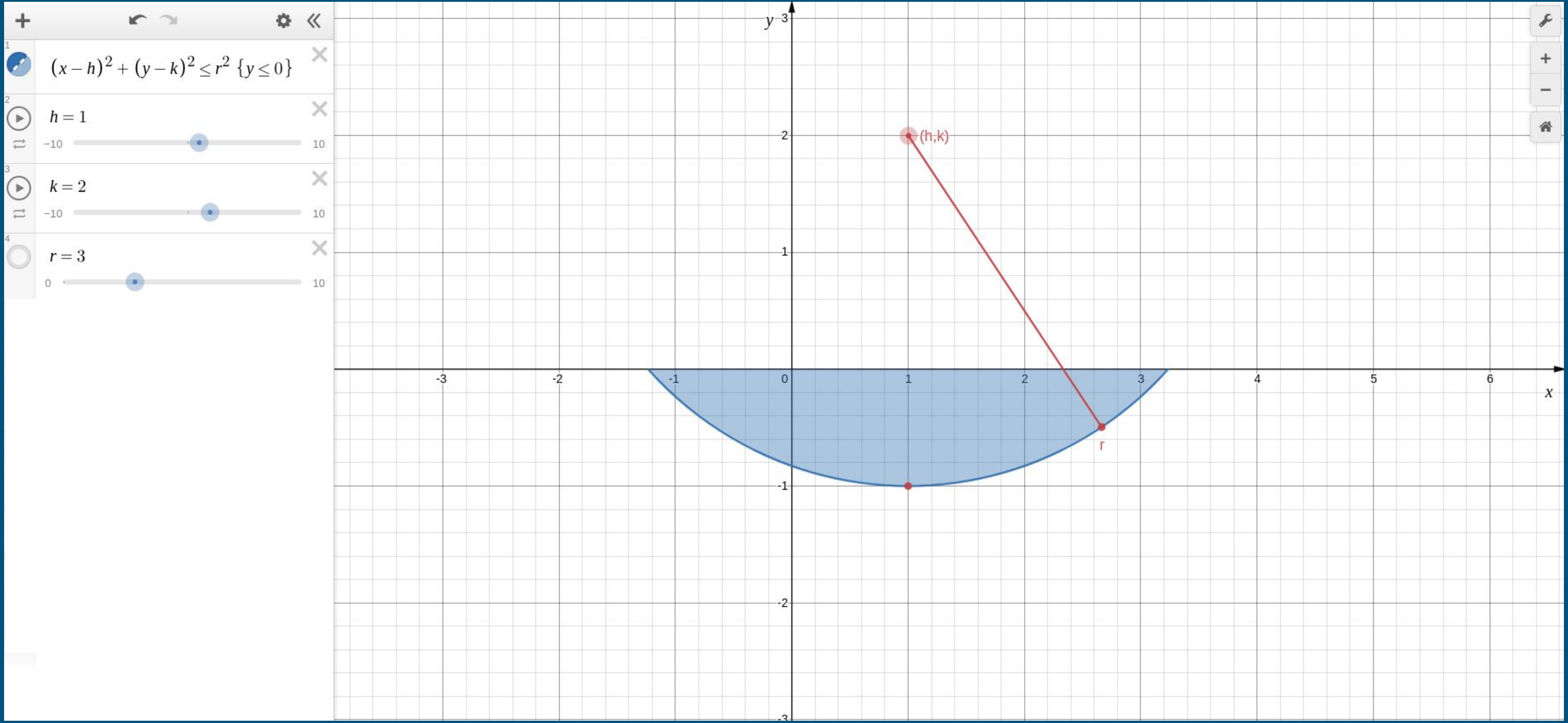
$h = 1$

$k = 2$

$r = 3$

Source: Desmos

Toy Problem: Guess the solution?



$(x-h)^2 + (y-k)^2 \le r^2 \; \{y \le 0\}$

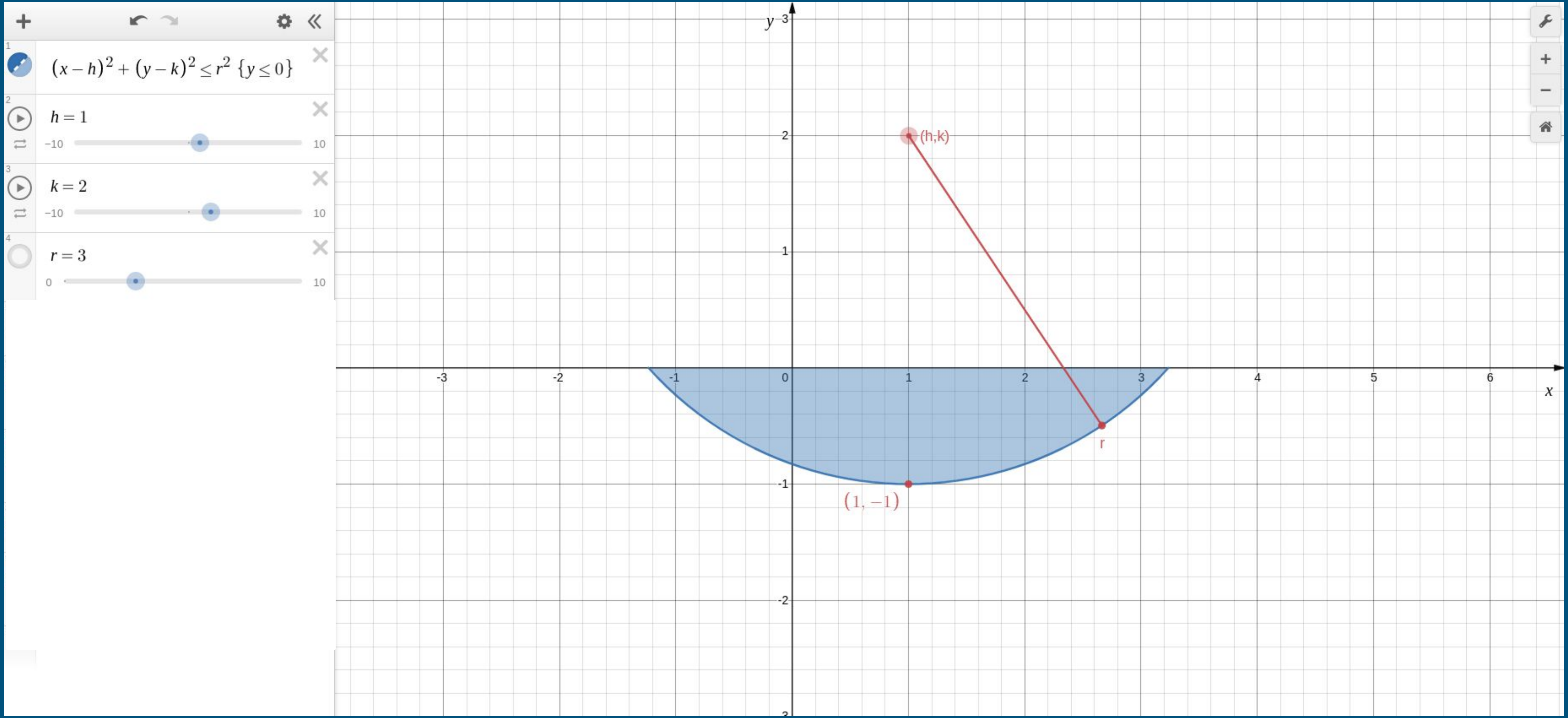
$h = 1$

$k = 2$

$r = 3$

Source: Desmos

Toy Problem: Guess the solution?


$(x-h)^2+(y-k)^2 \le r^2 \; \{y \le 0\}$

$h = 1$

$k = 2$

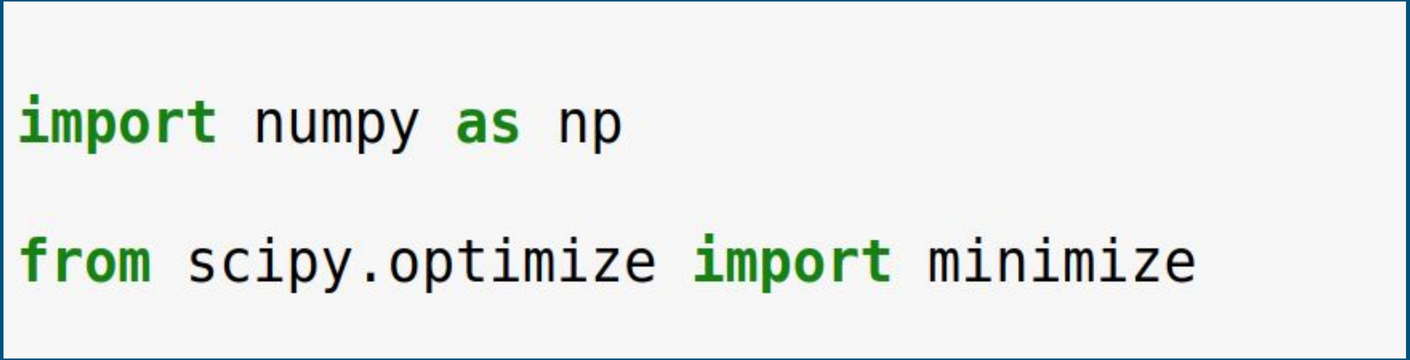
$r = 3$

Source: Desmos

Toy Problem: Solution

- Import required libraries
- Define the problem in Python
- Solve!

Toy Problem: Solution: Libraries!



```python
import numpy as np

from scipy.optimize import minimize
```

Toy Problem: Solution: Functions!

- Objective function:

$$f(x_1, x_2) = x_2$$

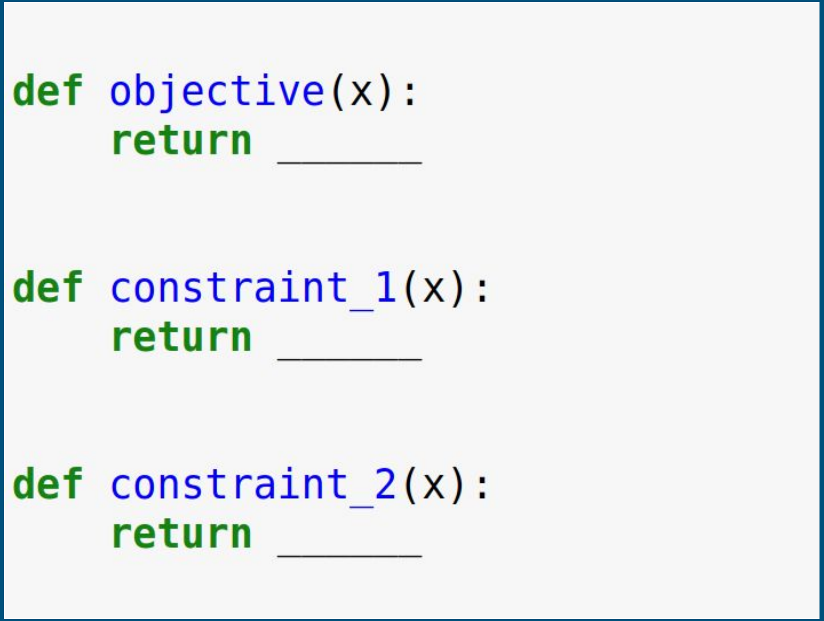
- Constraint 1:

$$c_1(x) = -(x_1 - 1)^2 - (x_2 - 2)^2 + 9 \geq 0$$

- Constraint 2:

$$c_2(x) = -x_2 \geq 0$$

Toy Problem: Solution: Functions!



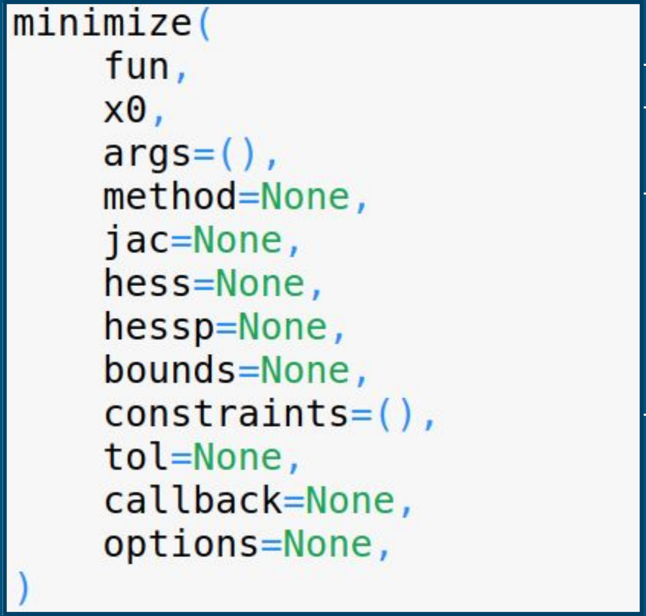
```
def objective(x):
    return ______


def constraint_1(x):
    return ______


def constraint_2(x):
    return ______
```

NB:

x is a vector!

Toy Problem: Solution: Solve!

minimize?



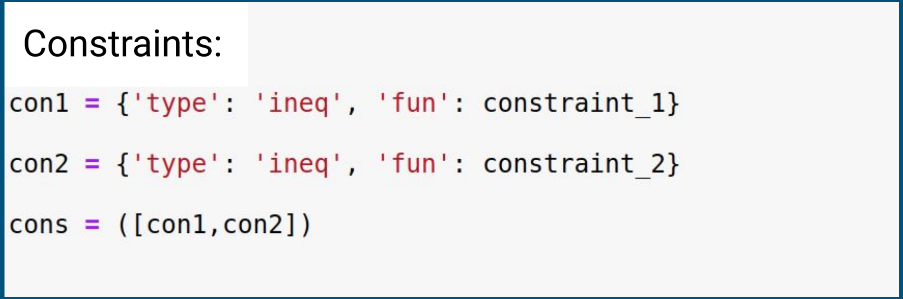
```
minimize(
    fun,
    x0,
    args=(),
    method=None,
    jac=None,
    hess=None,
    hessp=None,
    bounds=None,
    constraints=(),
    tol=None,
    callback=None,
    options=None,
)
```

→ Objective function

→ First guess

→ **SLSQP**

→

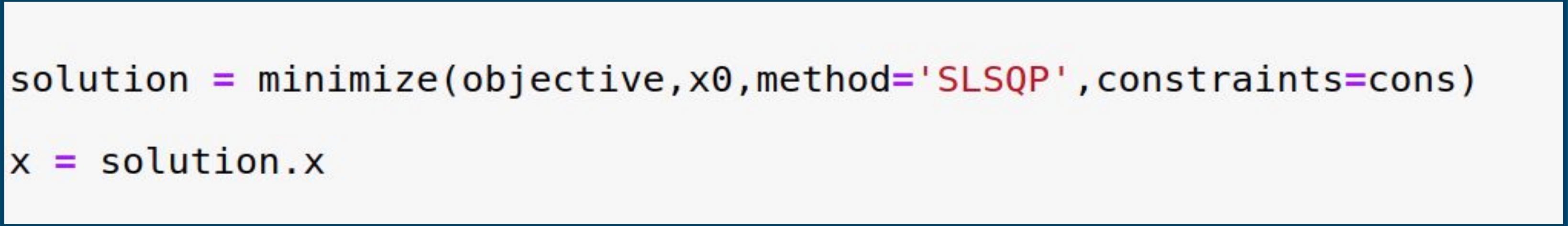


Constraints:

```
con1 = {'type': 'ineq', 'fun': constraint_1}

con2 = {'type': 'ineq', 'fun': constraint_2}

cons = ([con1,con2])
```

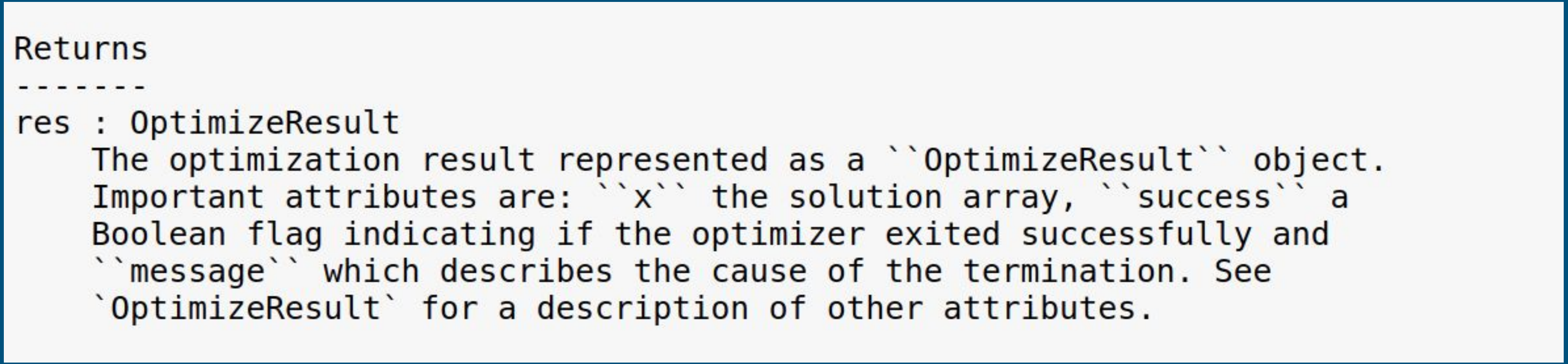
Source: Jupyter notebook

Toy Problem: Solution: Solve!



```python
solution = minimize(objective,x0,method='SLSQP',constraints=cons)

x = solution.x
```

minimize?



```
Returns
-------
res : OptimizeResult
    The optimization result represented as a ``OptimizeResult`` object.
    Important attributes are: ``x`` the solution array, ``success`` a
    Boolean flag indicating if the optimizer exited successfully and
    ``message`` which describes the cause of the termination. See
    `OptimizeResult` for a description of other attributes.
```

Source: Jupyter notebook

Toy Problem: KKT Optimality Conditions

$$
\begin{aligned}
\nabla_x \mathcal{L}(x_0, \lambda^0) &= 0 & & \text{dual feasibility} \\
c_i(x_0) &\geq 0 & \forall i & \quad \text{primal feasibility} \\
d_j(x_0) &= 0 & \forall j & \quad \text{primal feasibility} \\
\lambda_i^0 &\geq 0 & \forall i & \quad \text{dual positivity} \\
\lambda_i^0 c_i(x_0) &= 0 & \forall i & \quad \text{complementary slackness}
\end{aligned}
$$

(*for* $1 \leq i \leq k$ *and* $1 \leq j \leq r$).

Toy Problem: KKT Optimality Conditions

$$\mathcal{L}(x_1, x_2, \lambda_1, \lambda_2) = f(x_1, x_2) - \lambda_1 c_1(x) - \lambda_2 c_2(x)$$

Toy Problem: KKT Optimality Conditions

$$\begin{aligned}\mathcal{L}(x_1, x_2, \lambda_1, \lambda_2) &= f(x_1, x_2) - \lambda_1 c_1(x) - \lambda_2 c_2(x) \\ &= x_2 - \lambda_1[9 - (x_1 - 1)^2 - (x_2 - 2)^2] - \lambda_2[-x_2]\end{aligned}$$

Toy Problem: KKT Optimality Conditions

$$\begin{aligned}\mathcal{L}(x_1, x_2, \lambda_1, \lambda_2) &= f(x_1, x_2) - \lambda_1 c_1(x) - \lambda_2 c_2(x) \\ &= x_2 - \lambda_1[9 - (x_1 - 1)^2 - (x_2 - 2)^2] - \lambda_2[-x_2]\end{aligned}$$

$$\nabla_x \mathcal{L}(x_1, x_2, \lambda_1, \lambda_2) = \begin{bmatrix} 2\lambda_1(x_1 - 1) \\ 1 + 2\lambda_1(x_2 - 2) + \lambda_2 \end{bmatrix}$$

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1 - 1) \\ 1 + 2\lambda_1(x_2 - 2) + \lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = 9 - (x_1 - 1)^2 - (x_2 - 2)^2 \geq 0 \\ c_2(x) = -x_2 \geq 0 \\ \lambda_1, \lambda_2 \geq 0 \\ \lambda_1 c_1(x) = \lambda_1(9 - (x_1 - 1)^2 - (x_2 - 2)^2) = 0 \\ \lambda_2 c_2(x) = \lambda_2(-x_2) = 0 \end{cases}$$

$(\text{for } x_1 = x_{10}, x_2 = x_{20}, \lambda_1 = \lambda_1^0, \lambda_2 = \lambda_2^0).$

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1 - 1) \\ 1 + 2\lambda_1(x_2 - 2) + \lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = 9 - (x_1 - 1)^2 - (x_2 - 2)^2 \geq 0 \\ c_2(x) = -x_2 \geq 0 \\ \lambda_1, \lambda_2 \geq 0 \\ \lambda_1 c_1(x) = \lambda_1(9 - (x_1 - 1)^2 - (x_2 - 2)^2) = 0 \\ \lambda_2 c_2(x) = \lambda_2(-x_2) = 0 \end{cases}$$

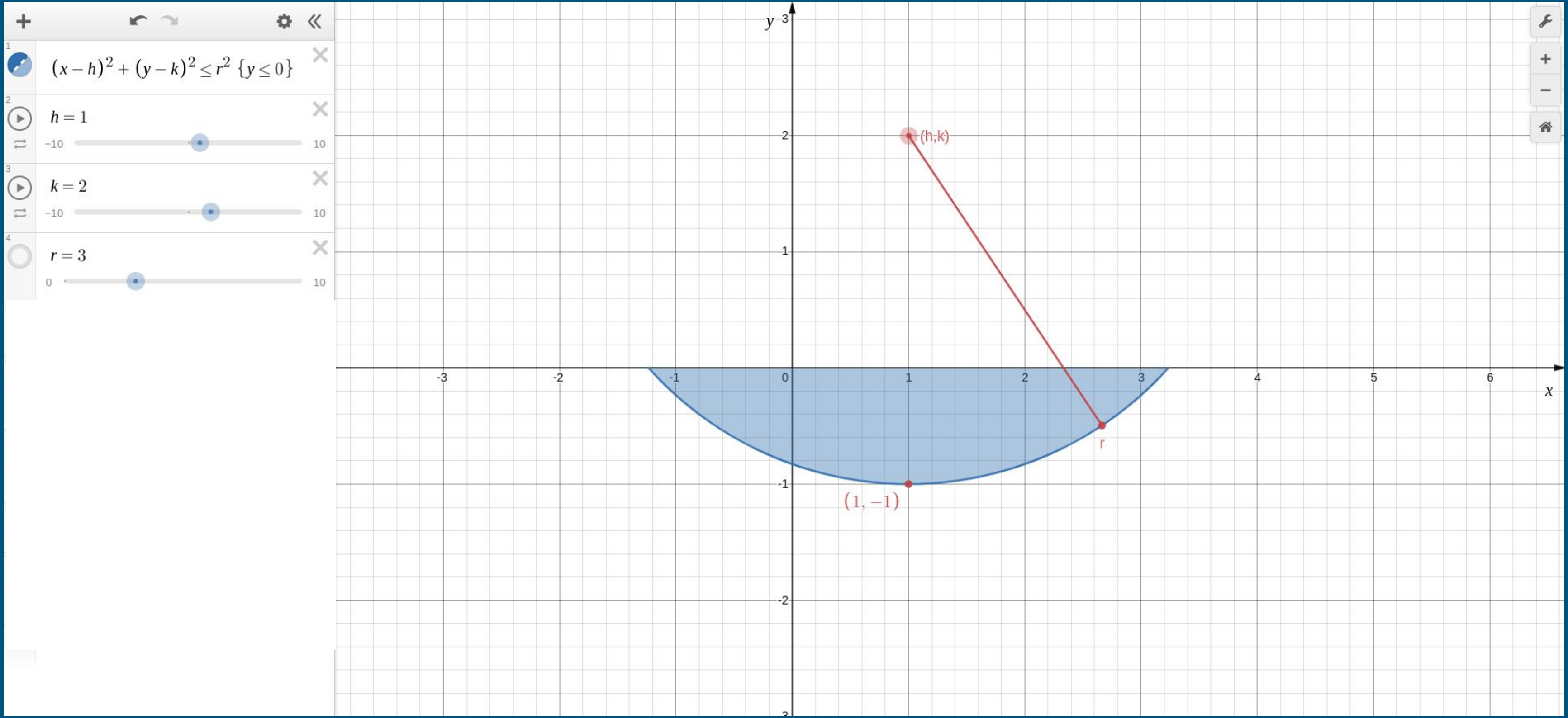
$(\text{for } x_1 = 1, x_2 = -1, \lambda_1 = \lambda_1^0, \lambda_2 = \lambda_2^0).$

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1-1) \\ 1+2\lambda_1(x_2-2)+\lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = 9-(x_1-1)^2-(x_2-2)^2 \geq 0 \\ c_2(x) = \boxed{-x_2 \geq 0} \\ \lambda_1, \lambda_2 \geq 0 \\ \lambda_1 c_1(x) = \lambda_1(9-(x_1-1)^2-(x_2-2)^2) = 0 \\ \lambda_2 c_2(x) = \lambda_2(-x_2) = 0 \end{cases}$$

$(\text{for } x_1 = 1, x_2 = -1, \lambda_1 = \lambda_1^0, \lambda_2 = \lambda_2^0).$

Toy Problem: Visualization



$(x-h)^2 + (y-k)^2 \leq r^2 \ \{y \leq 0\}$

$h = 1$

$k = 2$

$r = 3$

Source: Desmos

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1-1) \\ 1+2\lambda_1(x_2-2)+\lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = \boxed{9-(x_1-1)^2-(x_2-2)^2} \geq 0 \xrightarrow{} \boxed{0} \\ c_2(x) = \boxed{-x_2 \geq 0} \\ \lambda_1, \lambda_2 \geq 0 \\ \lambda_1 c_1(x) = \lambda_1(9-(x_1-1)^2-(x_2-2)^2) = 0 \\ \lambda_2 c_2(x) = \lambda_2(-x_2) = 0 \end{cases}$$

$$(\text{for } x_1 = 1, x_2 = -1, \lambda_1 = \lambda_1^0, \lambda_2 = \lambda_2^0).$$

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1-1) \\ 1 + 2\lambda_1(x_2-2) + \lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = \boxed{9-(x_1-1)^2-(x_2-2)^2} \geq 0 \\ c_2(x) = \boxed{-x_2 \geq 0} \\ \boxed{\lambda_1, \lambda_2 \geq 0} \\ \lambda_1 c_1(x) = \lambda_1(9-(x_1-1)^2-(x_2-2)^2) = 0 \\ \lambda_2 c_2(x) = \lambda_2(-x_2) = 0 \end{cases}$$

$$\xrightarrow{} \boxed{0}$$

$(\text{for } x_1 = 1, x_2 = -1, \lambda_1 = \lambda_1^0, \lambda_2 = \lambda_2^0).$

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1-1) \\ 1+2\lambda_1(x_2-2)+\lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = \boxed{9-(x_1-1)^2-(x_2-2)^2} \geq 0 \\ c_2(x) = \boxed{-x_2 \geq 0} \\ \boxed{\lambda_1, \lambda_2 \geq 0} \\ \lambda_1 c_1(x) = \lambda_1(9-(x_1-1)^2-(x_2-2)^2) = 0 \\ \lambda_2 c_2(x) = \lambda_2(-x_2) = 0 \end{cases}$$

$\boxed{\lambda_1 > 0, \lambda_2 = 0}$ $\boxed{0}$

$(\text{for } x_1 = 1, x_2 = -1, \lambda_1 = \lambda_1^0, \lambda_2 = \lambda_2^0).$

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1-1) \\ 1+2\lambda_1(x_2-2)+\lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = \boxed{9-(x_1-1)^2-(x_2-2)^2} \geq 0 \quad \longrightarrow \quad \boxed{0} \\ c_2(x) = \boxed{-x_2 \geq 0} \\ \boxed{\lambda_1, \lambda_2 \geq 0} \quad \longleftarrow \quad \boxed{\lambda_1 > 0, \lambda_2 = 0} \\ \lambda_1 c_1(x) = \boxed{\lambda_1(9-(x_1-1)^2-(x_2-2)^2) = 0} \\ \lambda_2 c_2(x) = \lambda_2(-x_2) = 0 \end{cases}$$

$$(\text{for } x_1 = 1, x_2 = -1, \lambda_1 = \lambda_1^0, \lambda_2 = \lambda_2^0).$$

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1-1) \\ 1+2\lambda_1(x_2-2)+\lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = \boxed{9-(x_1-1)^2-(x_2-2)^2} \geq 0 \quad \longrightarrow \boxed{0} \\ c_2(x) = \boxed{-x_2 \geq 0} \\ \boxed{\lambda_1, \lambda_2 \geq 0} \longleftarrow \boxed{\lambda_1 > 0, \lambda_2 = 0} \\ \lambda_1 c_1(x) = \boxed{\lambda_1(9-(x_1-1)^2-(x_2-2)^2) = 0} \\ \lambda_2 c_2(x) = \boxed{\lambda_2(-x_2) = 0} \end{cases}$$

$(\text{for } x_1 = 1, x_2 = -1, \lambda_1 = \lambda_1^0, \lambda_2 = \lambda_2^0).$

Toy Problem: KKT Optimality Conditions

$x_1 = 1, x_2 = -1, \lambda_2 = 0$

Toy Problem: KKT Optimality Conditions

$x_1 = 1, x_2 = -1, \lambda_2 = 0$

$$\nabla_x \mathcal{L}(x_1, x_2, \lambda_1, \lambda_2) = \begin{pmatrix} 2\lambda_1(x_1 - 1) \\ 1 + 2\lambda_1(x_2 - 2) + \lambda_2 \end{pmatrix}$$

Toy Problem: KKT Optimality Conditions

$x_1 = 1, x_2 = -1, \lambda_2 = 0$

$$\nabla_x \mathcal{L}(x_1, x_2, \lambda_1, \lambda_2) = \begin{pmatrix} 2\lambda_1(x_1 - 1) \\ 1 + 2\lambda_1(x_2 - 2) + \lambda_2 \end{pmatrix}$$

$$= \begin{pmatrix} 2\lambda_1(1 - 1) \\ 1 + 2\lambda_1(-1 - 2) \end{pmatrix}$$

Toy Problem: KKT Optimality Conditions

$x_1 = 1, x_2 = -1, \lambda_2 = 0$

$$\nabla_x \mathcal{L}(x_1, x_2, \lambda_1, \lambda_2) = \begin{pmatrix} 2\lambda_1(x_1 - 1) \\ 1 + 2\lambda_1(x_2 - 2) + \lambda_2 \end{pmatrix}$$

$$= \begin{pmatrix} 2\lambda_1(1 - 1) \\ 1 + 2\lambda_1(-1 - 2) \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 1 - 6\lambda_1 \end{pmatrix}$$

Toy Problem: KKT Optimality Conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = \begin{pmatrix} 2\lambda_1(x_1 - 1) \\ 1 + 2\lambda_1(x_2 - 2) + \lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ c_1(x) = 9 - (x_1 - 1)^2 - (x_2 - 2)^2 \geq 0 \\ c_2(x) = -x_2 \geq 0 \\ \lambda_1, \lambda_2 \geq 0 \\ \lambda_1 c_1(x) = \lambda_1(9 - (x_1 - 1)^2 - (x_2 - 2)^2) = 0 \\ \lambda_2 c_2(x) = \lambda_2(-x_2) = 0 \end{cases}$$

$$(\text{for } x_1 = 1, x_2 = -1, \lambda_1 = \frac{1}{6}, \lambda_2 = 0).$$

Questions?

Water Filling Problem

Water Filling Problem: Formulation

Problem statement:

How to allocate a total power of 1 to the channels in order to maximize the total communication rate.

Water Filling Problem: Formulation

Variables:

x_i - Transmitter power allocated to the ith channel

α_i - Noise

Water Filling Problem: Formulation

Goal:

$$\max_{x \in C} \sum_{i=1}^{n} \log(\alpha_i + x_i)$$

What are the constraints?

Water Filling Problem: Formulation

Minimization Problem:

$$\min_{x \in R^n} -\sum_{i=1}^{n} \log(\alpha_i + x_i)$$

$$\text{s.t.} \quad x \geq 0$$

$$\text{and} \quad 1^T x - 1 = 0$$

Water Filling Problem: Task 1

Cases:

- Random Noise:

```
alpha = np.random.random(4)*0.25
```

- Equal Noise:

```
alpha = np.array([0.25]*4)
```

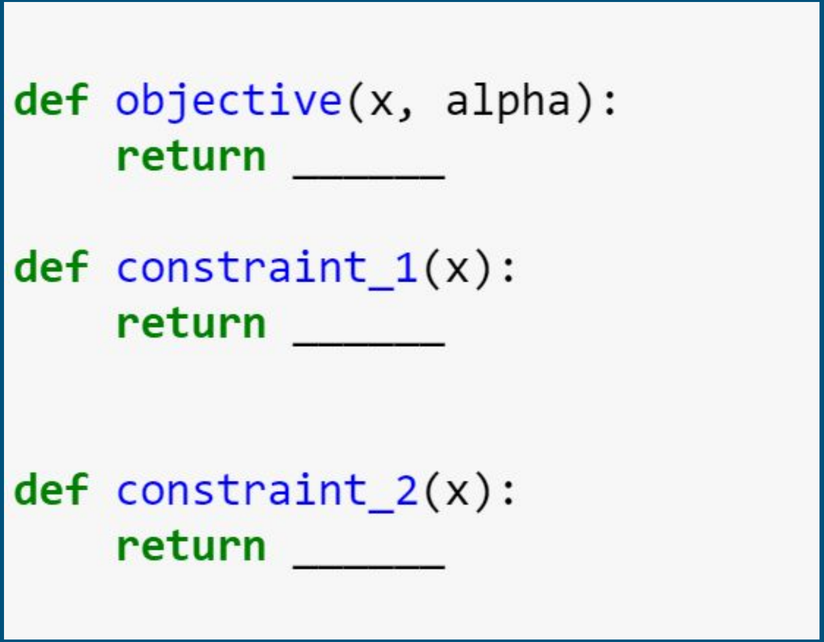
Water Filling Problem: Solution

- Import required libraries
- Define the problem in Python
- Solve!

Water Filling Problem: Solution: Functions!

- Objective function: $$f(x) = -\sum_{i=1}^{n} \log(\alpha_i + x_i)$$
- Constraint 1: $$c_1(x) = x \geq 0$$
- Constraint 2: $$d_1(x) = 1^T x - 1 = 0$$

Water Filling Problem: Solution: Functions!



```
def objective(x, alpha):
    return ______

def constraint_1(x):
    return ______


def constraint_2(x):
    return ______
```

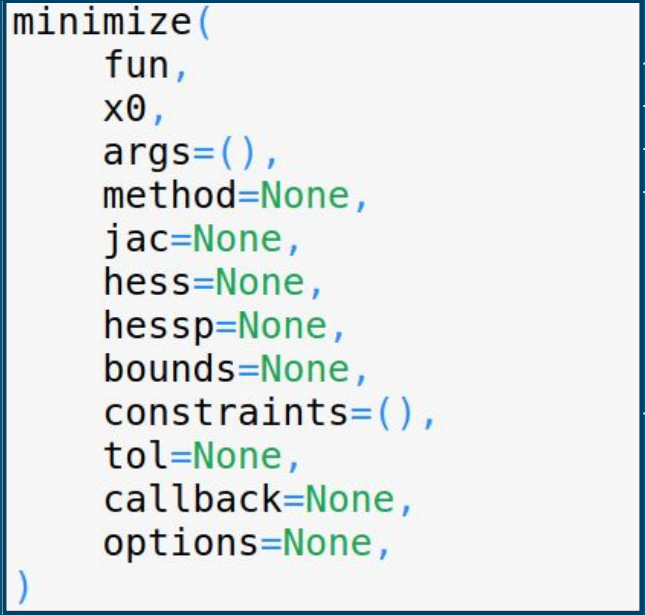
NB:

x is a vector!

alpha (α) is a vector!

Water Filling Problem: Solution: Solve!

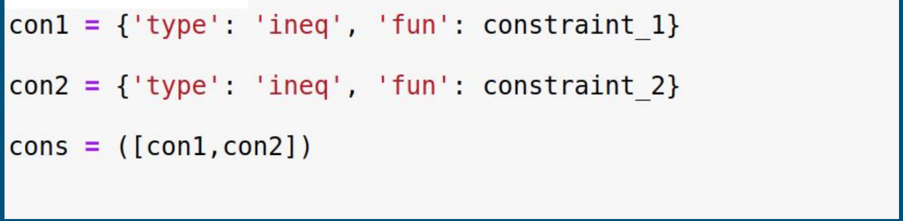
minimize?



```
minimize(
    fun,
    x0,
    args=(),
    method=None,
    jac=None,
    hess=None,
    hessp=None,
    bounds=None,
    constraints=(),
    tol=None,
    callback=None,
    options=None,
)
```

Objective function
First guess
Extra arguments passed to the objective function
SLSQP

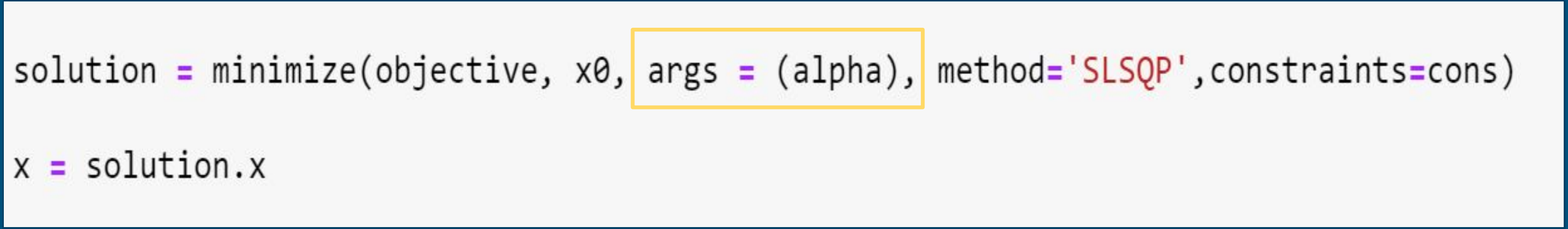
Constraints:



```
con1 = {'type': 'ineq', 'fun': constraint_1}

con2 = {'type': 'ineq', 'fun': constraint_2}

cons = ([con1,con2])
```

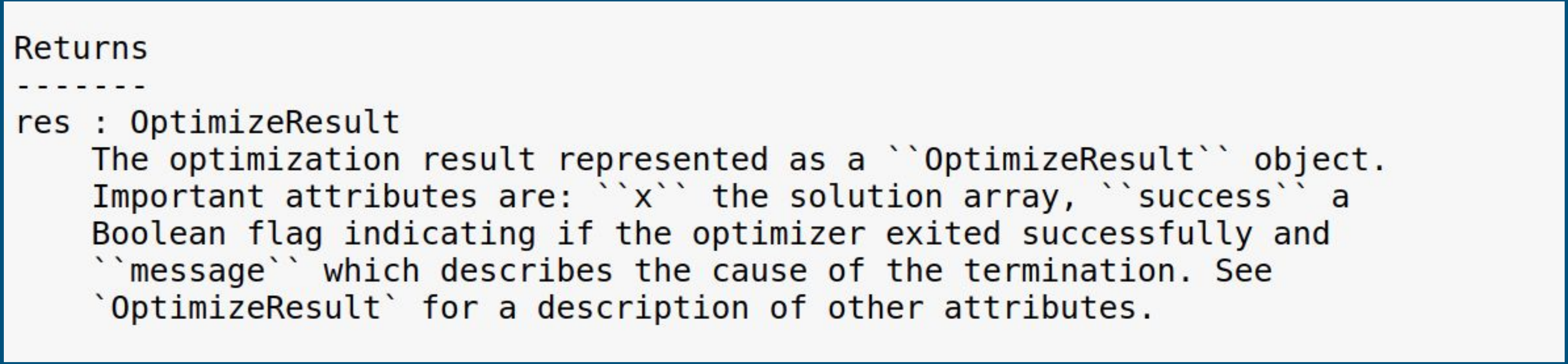
Source: Jupyter notebook

Water Filling Problem: Solution: Solve!



```
solution = minimize(objective, x0, args = (alpha), method='SLSQP',constraints=cons)

x = solution.x
```

minimize?

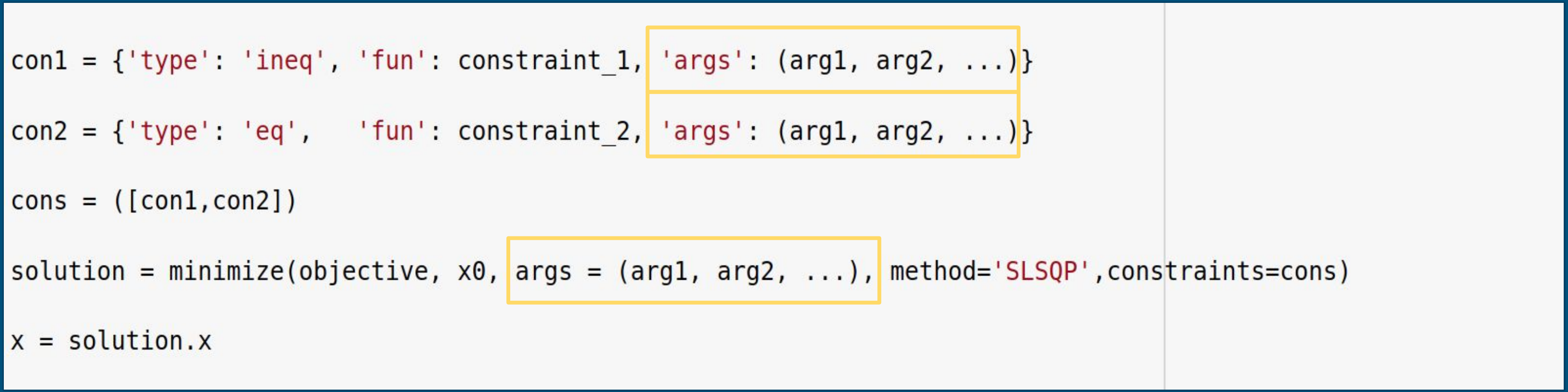


```
Returns
-------
res : OptimizeResult
    The optimization result represented as a ``OptimizeResult`` object.
    Important attributes are: ``x`` the solution array, ``success`` a
    Boolean flag indicating if the optimizer exited successfully and
    ``message`` which describes the cause of the termination. See
    `OptimizeResult` for a description of other attributes.
```

Source: Jupyter notebook

Water Filling Problem: Solution: Solve!

If the constraints require additional arguments?



```
con1 = {'type': 'ineq', 'fun': constraint_1, 'args': (arg1, arg2, ...)}

con2 = {'type': 'eq',    'fun': constraint_2, 'args': (arg1, arg2, ...)}

cons = ([con1,con2])

solution = minimize(objective, x0, args = (arg1, arg2, ...), method='SLSQP',constraints=cons)

x = solution.x
```

Entropy Maximization Problem

Entropy Maximization Problem

- Standard form:

$$H(x) = -\sum_{i=1}^{n} x_i \log x_i$$

Entropy Maximization Problem: Functions!

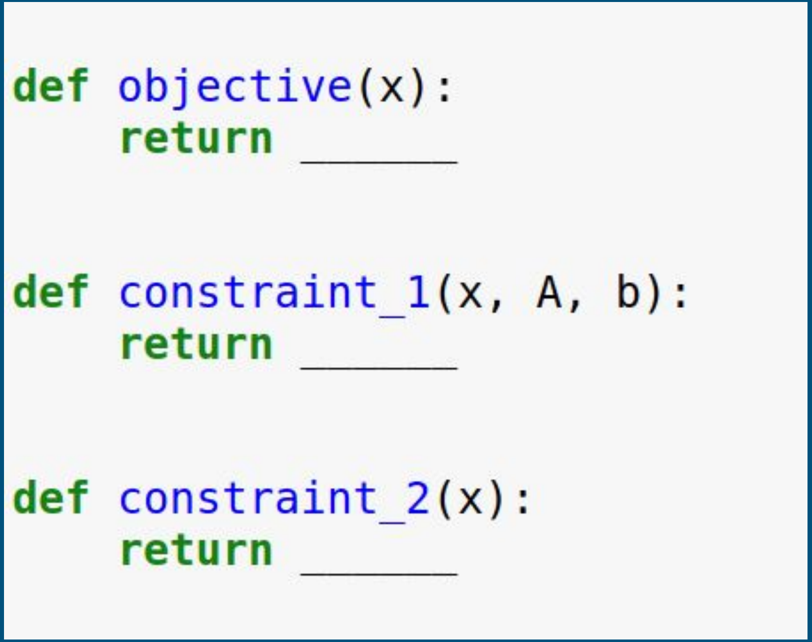
- Objective function: $f(x) = \sum_{i=1}^{n} x_i \log x_i$
- Constraint 1:

$$c_1(x) = b - Ax \geq 0$$

- Constraint 2:

$$d_1(x) = 1^T x - 1 = 0$$

Entropy Maximization Problem: Functions!



```
def objective(x):
    return ______


def constraint_1(x, A, b):
    return ______


def constraint_2(x):
    return ______
```

NB:

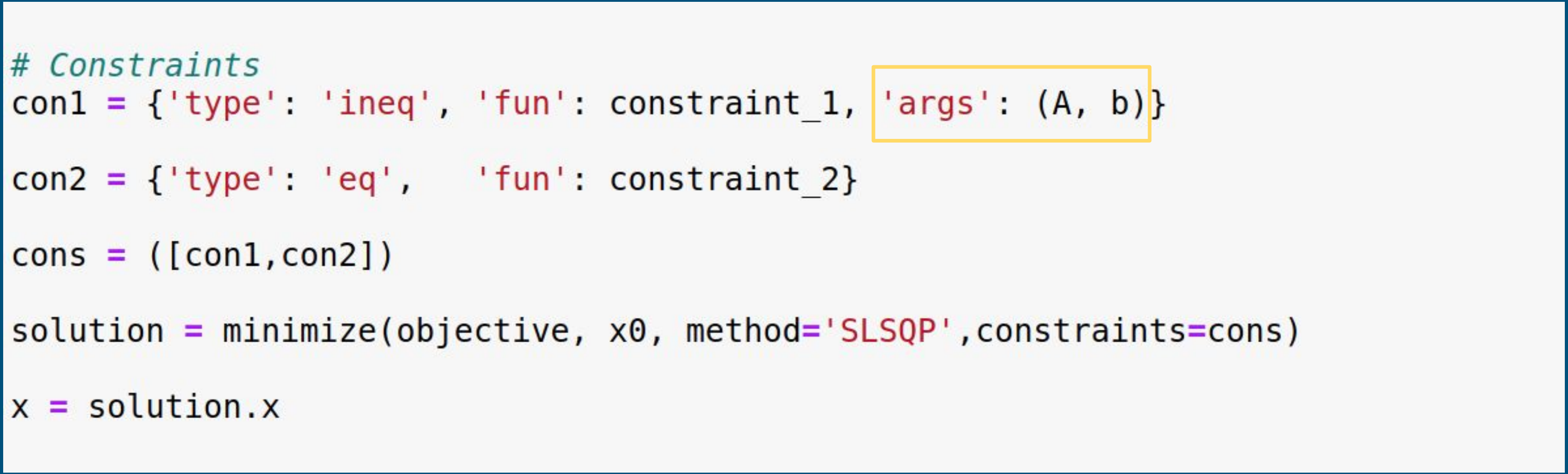
x is a vector!

A is a matrix!

b is a vector!

Output of const_1 should be a vector!

Entropy Maximization Problem: Solution!



```
# Constraints
con1 = {'type': 'ineq', 'fun': constraint_1, 'args': (A, b)}

con2 = {'type': 'eq',    'fun': constraint_2}

cons = ([con1,con2])

solution = minimize(objective, x0, method='SLSQP',constraints=cons)

x = solution.x
```

Questions?

Thanks!