問題 1. 展開すると

$$
\begin{aligned}
\begin{bmatrix} \dfrac{\partial}{\partial \beta_1} \displaystyle\sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{p} \beta_k x_{i,k} \right)^2 \\ \vdots \\ \dfrac{\partial}{\partial \beta_p} \displaystyle\sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{p} \beta_k x_{i,k} \right)^2 \end{bmatrix} &= -2 \begin{bmatrix} \displaystyle\sum_{i=1}^{N} x_{i,1} \left(y_i - \sum_{k=1}^{p} x_{i,k}\beta_k \right) \\ \vdots \\ \displaystyle\sum_{i=1}^{N} x_{i,p} \left(y_i - \sum_{k=1}^{p} x_{i,k}\beta_k \right) \end{bmatrix} \\
&= -2 \begin{bmatrix} x_{1,1} & \cdots & x_{N,1} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,p} \end{bmatrix} \begin{bmatrix} y_1 - \sum_{k=1}^{p} x_{1,k}\beta_k \\ \vdots \\ y_N - \sum_{k=1}^{p} x_{N,k}\beta_k \end{bmatrix} \\
&= -2X^T(y - X\beta)
\end{aligned} \tag{1}
$$

となることから等式が成立する．さらに $X^T X$ が正則ならば，$\|y - X\beta\|_2^2$ を最小にする β は

$$
\begin{bmatrix} \frac{\partial}{\partial \beta_1} \|y - X\hat{\beta}\|^2 \\ \vdots \\ \frac{\partial}{\partial \beta_p} \|y - X\hat{\beta}\|^2 \end{bmatrix} = \mathbf{0}
$$

を満たすので，(1) 式より

$$
\begin{aligned}
-2X^t(y - X\hat{\beta}) &= 0 \\
X^T X \hat{\beta} &= X^T y \\
\therefore \hat{\beta} &= (X^T X)^{-1} X^T y
\end{aligned}
$$

となることがわかる．さらに，求める関数 liner を python で構成する際のソースコードは以下の通り．

```
def linear(X,y):
    p = X.shape[1]
    x_bar = np.zeros(p)
    for j in range(p):
        x_bar[j] = np.mean(X[:,j])
    for j in range(p):
        X[:,j] = X[:, j] - x_bar[j]
    y_bar=np.mean(y)
    y = y - y_bar
    "beta" "=" "np.dot("
        "np.linalg.inv(np.dot(X.T,X)),np.dot(X.T,y)"
    ")" #空欄(1)
```

```
    "beta_0␣=␣y_bar␣-␣np.dot(x_bar,beta)"#空欄(2)
    return beta, beta_0
```

□

問題 2.

(a) 実数 $x \in \mathbb{R}$ を任意にとって固定する．$x = x_0$ ならば等式

$$f(x) = f(x_0) + f'(x_0)(x - x_0)$$

が成立するので $x \neq x_0$ とする．このとき，凸関数の定義より任意の $0 < \alpha < 1$ に対して

$$f(\alpha x + (1-\alpha)x_0) \leq \alpha f(x) + (1-\alpha)f(x_0)$$

が成立するので上式を変形して

$$\begin{aligned} f(\alpha x + (1-\alpha)x_0) - f(x_0) &\leq \alpha(f(x) - f(x_0)) \\ f(x) - f(x_0) &\geq \frac{f(\alpha x + (1-\alpha)x_0) - f(x_0)}{\alpha} \\ \therefore f(x) &\geq f(x_0) + \frac{f(\alpha x + (1-\alpha)x_0) - f(x_0)}{\alpha(x - x_0)}(x - x_0) \end{aligned}$$

が得られる．$0 < \alpha < 1$ は任意であったので，$\alpha \searrow 0$ とすることで

$$f(x) \geq f(x_0) + f'(x_0)(x - x_0)$$

となることがわかる．上式で実数 x は任意にとれたので，求めたい不等式が示せた．

(b) まず $\partial f(x_0)$ は空集合でないことに注意する．実際 (a) の結果より，

$$f(x) \geq f(x_0) + f'(x_0)(x - x_0)$$

が成立するので $f'(x_0) \in \partial f(x_0)$ が成立している．

実数 z が $z \in \partial f(x_0)$ を満たしているとする．すなわち，任意の $x \in \mathbb{R}$ に対して

$$f(x) \geq f(x_0) + z(x - x_0) \tag{2}$$

が成立していたとする．このとき，上式を変形することで $x > x_0$ の範囲において

$$z \leq \frac{f(x) - f(x_0)}{x - x_0}$$

となることが要請されるので

$$z \leq \lim_{x \searrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \tag{3}$$

となる必要がある．一方で，(2) 式より $x < x_0$ の範囲において

$$z \geq \frac{f(x) - f(x_0)}{x - x_0}$$

となることが要請されるので

$$z \geq \lim_{x \nearrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \tag{4}$$

となる必要がある．以上 (3),(4) 式より

$$\lim_{x \nearrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \leq z \leq \lim_{x \searrow x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

が成立するので，これと上式の最左辺と最右辺を見ることで

$$z \in \partial f(x_0) \Rightarrow z = f'(x_0)$$

が得られる．$f'(x_0) \in \partial f(x_0)$ であったので，これより $\partial f(x_0) = \{f'(x_0)\}$ が示せた．□

問題 3.

(a) 各 $x \in \mathbb{R}$ について場合分けして考える．

$x > \lambda$ **のとき**　この場合 $x > 0$ かつ $|x| - \lambda \geq 0$ であることより $\mathrm{sign}(x) = 1$ かつ $(|x| - \lambda)_+ = x - \lambda$ が得られる．したがって

$$\mathcal{S}_\lambda(x) = x - \lambda = \mathrm{sign}(x)(|x| - \lambda)_+$$

となることがわかる．

$|x| \leq \lambda$ **のとき**　この場合 $|x| - \lambda \leq 0$ であることから $(|x| - \lambda)_+ = 0$ となるので

$$\mathcal{S}_\lambda(x) = \mathrm{sign}(x)(|x| - \lambda)_+ = 0$$

が得られる．

$x < -\lambda$ のとき　この場合 $x < 0$ かつ $|x| - \lambda \geq 0$ であることより $\mathrm{sign}(x) = -1$ かつ $(|x| - \lambda)_+ = -x - \lambda$ が得られる．したがって

$$\mathcal{S}_\lambda(x) = x + \lambda = -(-x - \lambda) = \mathrm{sign}(x)(|x| - \lambda)_+$$

となることがわかる．

以上より任意の $x \in \mathbb{R}$ に対して

$$\mathcal{S}_\lambda(x) = \mathrm{sign}(x)(|x| - \lambda)_+$$

とかけることが示せた．

(b) (a) で示した対応を Python に実行させるコードは以下の通り．

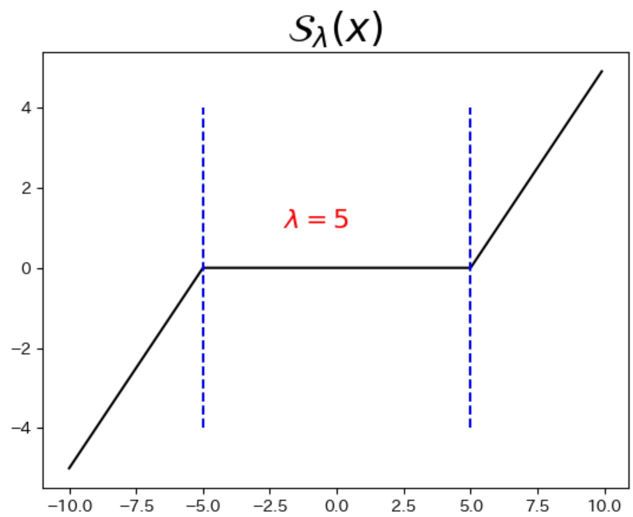
```
def soft_th(lam,x):
    return np.sign(x) * np.maximum(np.abs(x) - lam,0)
```

さらに $\lambda = 5$ の場合の関数 $\mathcal{S}_\lambda$ のグラフを出力すると下のようになる

入力

```
x = np.arange(-10,10,0.1)
y = soft_th(5,x)
plt.plot(x,y,c = "black")
plt.title(r"${\cal␣S}_\lambda(x)$",size = 24)
plt.plot([-5,-5],[-4,4],c = "blue",linestyle = "dashed")
plt.plot([5,5],[-4,4],c = "blue",linestyle = "dashed")
plt.text(-2,1,r"$\lambda␣=␣5$",c ="red",size = 16)
```

出力結果



確かに $\lambda = 5$ における関数 $\mathcal{S}_\lambda$ の動作と，定義した soft_th が一致していることが確認できる． □

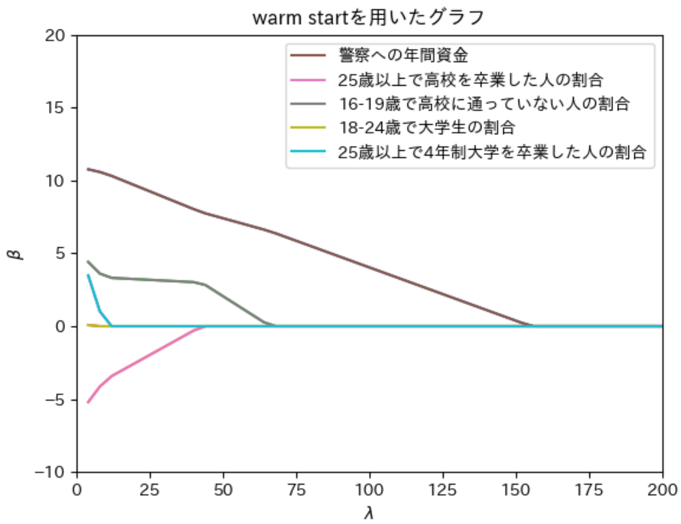
問題 4. 空欄を埋めたプログラムは以下の通り

```
def warm_start(X, y, lambda_max=100):
    dec = np.round(lambda_max / 50)
    lambda_seq = np.arange(lambda_max, 1, -dec)
    r = len(lambda_seq)
    p = X.shape[1]
    beta = np.zeros(p)
    coef_seq = np.zeros((r, p))
    for k in range(r):
        beta, _ = linear_lasso(X, y, lambda_seq[k], beta)
        coef_seq[k, :] = beta
    return coef_seq
```

```
df = np.loadtxt("crime.txt", delimiter="\t")
X = df[:, [i for i in range(2, 7)]]
p = X.shape[1]
y = df[:, 0]
coef_seq = warm_start(X, y, 200)
```

```
lambda_max = 200
dec = round(lambda_max / 50)
lambda_seq = np.arange(lambda_max, 1, -dec)
plt.ylim(np.min(coef_seq), np.max(coef_seq))
plt.xlabel(r"$\lambda$")
plt.ylabel("係数")
plt.xlim(0, 200)
plt.ylim(-10, 20)
for j in range(p):
    plt.plot(lambda_seq, coef_seq[:, j])
```

出力結果は下の通り



また λ の値が $\max_{1\le j\le p}\left|\frac{1}{N}\sum_{i=1}^{N} x_{i,j}y_i\right|$ よりも大きいとき，全ての $j=1,\cdots,p$ に対して

$$\beta_j = 0 \text{ かつ } r_{i,j} = y_i$$

が成立していることから

問題 5. 関数 ridge は以下のようにして構成できる

```
def ridge(X, y, lam=0):
    n, p = X.shape
    X, y, X_bar, X_sd, y_bar = centralize(X, y)
    beta = np.dot(
        np.linalg.inv(np.dot(X.T, X) + n * lam * np.eye(p)),
        np.dot(X.T, y)
```

```
    )
    beta = beta / X_sd
    beta_0 = y_bar - np.dot(X_bar, beta)
    return beta, beta_0
```

実行結果を確認する

```
df = np.loadtxt("crime.txt", delimiter="\t")
X = df[:, [i for i in range(2, 7)]]
y = df[:, 0]
linear(X, y)
```

```
(array([10.98067026, -6.08852939, 5.4803042 , 0.37704431, 5.50047122]),
    489.64859696903386)
```

```
ridge(X,y)
```

```
(array([10.98067026, -6.08852939, 5.4803042 , 0.37704431, 5.50047122]), 717.96)
```

```
ridge(X,y)
```

```
(array([ 0.0563518 , -0.01976397, 0.07786309, -0.0171218 , -0.0070393 ]), 717.96)
```

実行結果が一致していることが確認できた．

問題 6. 相異なる k, l について，X の第 k 列，第 l 列の N 個の成分がすべて等しいする．Ridge における損失関数 L は

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j) + \lambda\sum_{j=1}^{p}\beta_j^2$$

となるので，β_j, β_k で L を偏微分すると

$$\frac{\partial L}{\partial \beta_k} = -\frac{1}{N}\sum_{i=1}^{N} x_{i,k}(y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j) + \lambda\beta_k$$

$$\frac{\partial L}{\partial \beta_l} = -\frac{1}{N}\sum_{i=1}^{N} x_{i,l}(y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j) + \lambda\beta_l$$

となることがわかる．推定される β_k, β_l はこの偏微分の値が 0 になることから

$$\beta_k = \frac{1}{\lambda N}\sum_{i=1}^{N} x_{i,k}(y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j)$$
$$\beta_l = \frac{1}{\lambda N}\sum_{i=1}^{N} x_{i,l}(y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j)$$

を満たしており，仮定より $x_{i,k} = x_{i,l} (i = 1, \cdots, N)$ であることから

$$\begin{aligned}\beta_k &= \frac{1}{\lambda N}\sum_{i=1}^{N} x_{i,k}(y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j) \\ &= \frac{1}{\lambda N}\sum_{i=1}^{N} x_{i,l}(y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j) = \beta_l\end{aligned}$$

となり，推定される β_k, β_l が等しくなることがわかる． □

問題 7. elastic ネットは，例えば以下のようにして構成できる

```
def elastic_net(X, y, lam=0, alpha=1, beta=None): #
    n, p = X.shape
    if beta is None:
        beta = np.zeros(p)
    X, y, X_bar, X_sd, y_bar = centralize(X, y) # 中心化
    eps = 1
    beta_old = copy.copy(beta)
    while eps > 0.00001: # このループの収束を待つ
        for j in range(p):
            r = y
            for k in range(p):
                if j != k:
                    r = r - X[:, k] * beta[k]
            z = (np.dot(r, X[:, j]) / n) ##
            beta[j] = (soft_th(lam * alpha, z) ##
                       / (np.dot(X[:, j], X[:, j]) / n + (1-alpha) * lam)) ##
        eps = np.linalg.norm(beta - beta_old, 2)
        beta_old = copy.copy(beta)
    beta = beta / X_sd # 各変数の係数を正規化前のものに戻す
    beta_0 = y_bar - np.dot(X_bar, beta)
    return beta, beta_0
```