```
import numpy as np

class Bandits():
    def __init__(self,k,mu,sigma):
        self.k = k
        self.means = np.random.normal(mu,sigma,k)
        self.variances = np.ones(k)
        self._step = 0
        self.state = None


    def reset(self):
        """
        return (observation, reward, terminated, truncated, info)
        """
        self._step = 0
        self.state = None
        return self._get_obs(), 0, False,False,  self._get_info(), # observation,
reward, terminated, truncated, info


    def _get_obs(self):
        return self.state

    def _get_info(self):
        return {"steps": self._step}

    def get_optimal_action(self):
        return np.argmax(self.means)


    def step(self,action:int):
        """
        input: action
        return (observation, reward, terminated, truncated, info)
        """
        self._step +=1
        reward = np.random.normal(self.means[action],self.variances[action])
        return self._get_obs(), reward, True, False, self._get_info()


class Bandits_one(Bandits):
    def __init__(self):
        self.k = 3
        self.means = np.array([1, 2, 3])
        self.variances = np.ones(len(self.means))
        self._step = 0
        self.state = None

class Bandits_two(Bandits):
    def __init__(self):
        self.k = 4
        self.means = np.array([1, 2, 3, 2.5])
        self.variances = np.ones(len(self.means))
        self._step = 0
        self.state = None
```

```
class Bandits_three(Bandits):
    def __init__(self):
        self.k = 3
        self.means = np.array([2, 1.5, 2.2])
        self.variances = np.array([1, 1, 3])
        self._step = 0
        self.state = None

class Bandits_four(Bandits):
    def __init__(self, gene:int = 0):
        self.k = 3
        self.means = {0: np.array([1, 2, 2.2]),
                      1: gene*np.array([3, 1, 2.2])}
        self.variances = np.array([1, 1, 3])
        self.state = gene

    def reset(self):
        """
        return (observation, reward, terminated, truncated, info)
        """
        self._step = 0
        return self._get_obs(), 0, False,False,  self._get_info(), # observation,
reward, terminated, truncated, info


    def step(self,action:int):
        """
        input: action
        return (observation, reward, terminated, truncated, info)
        """
        self._step +=1
        reward = np.random.normal(self.means[self.state]
[action],self.variances[action])
        return self._get_obs(), reward, True, False, self._get_info()

    def get_optimal_action(self):
        return np.argmax(self.means[self.state])
```