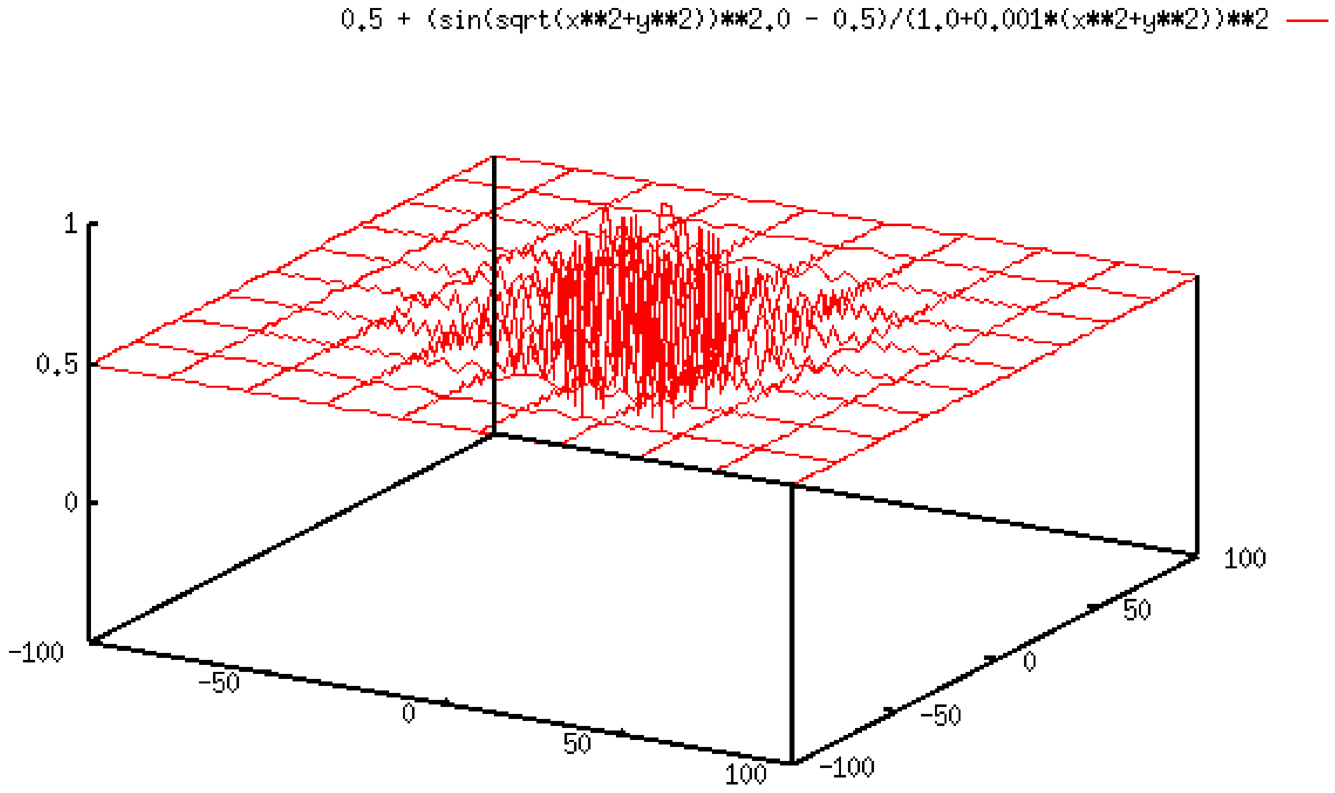
# A Simplified GA

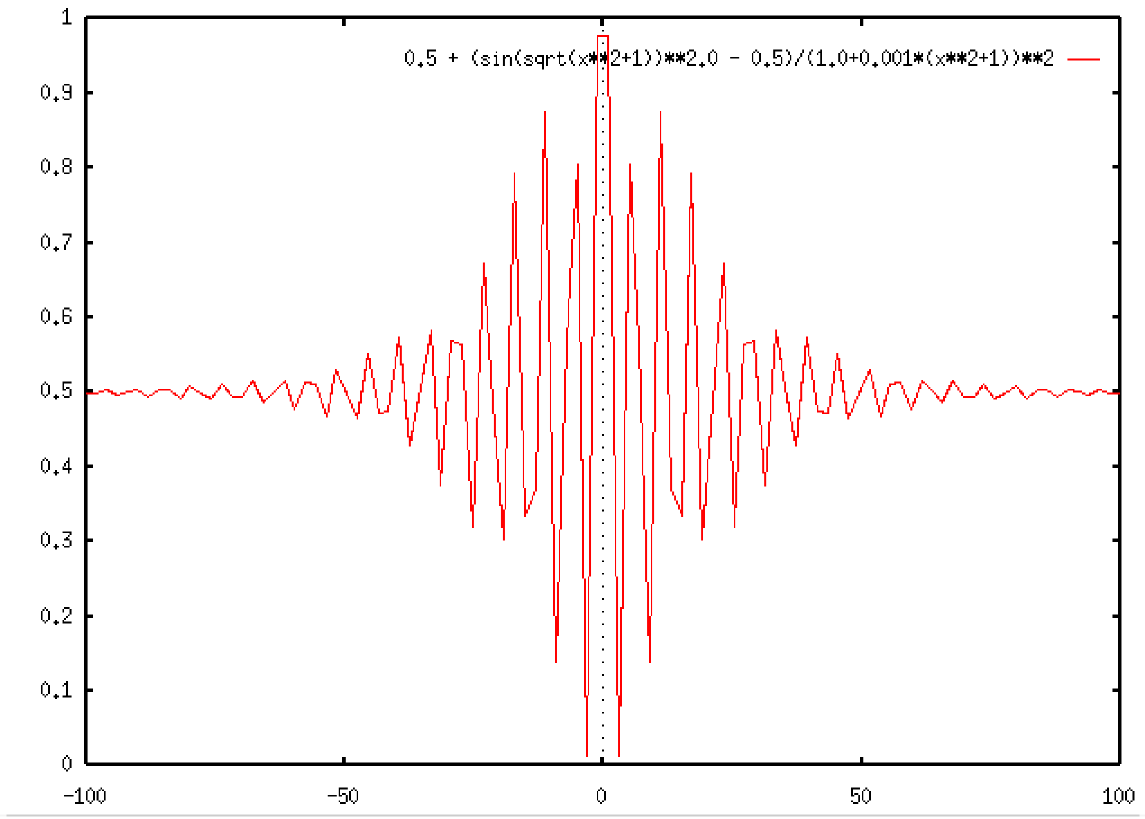
# A Simple Example

- Let’s walk through a simple example!
- Let’s say you were asked to solve the following problem:
  - Maximize:
  - $f(x,y) = 0.5 + (\sin(\mathrm{sqrt}(x^2+y^2))^2 - 0.5)/(1.0 + 0.001(x^2+y^2))^2$
  - Where x and y are take from [-100.0,100.0]
  - You must find a solution that is greater than 0.99754, and you can only evaluate a total of 4000 candidate solutions (CSs)
- This seems like a difficult problem.
  - It would be nice if we could see what it looks like!
  - This may help us determine a good algorithm for solving it.

# A 3D view of f(x,y)



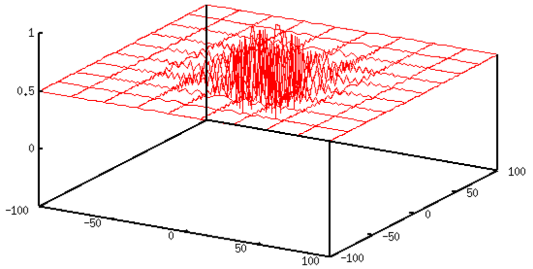
# Looking at only one dimension f(x,1.0)

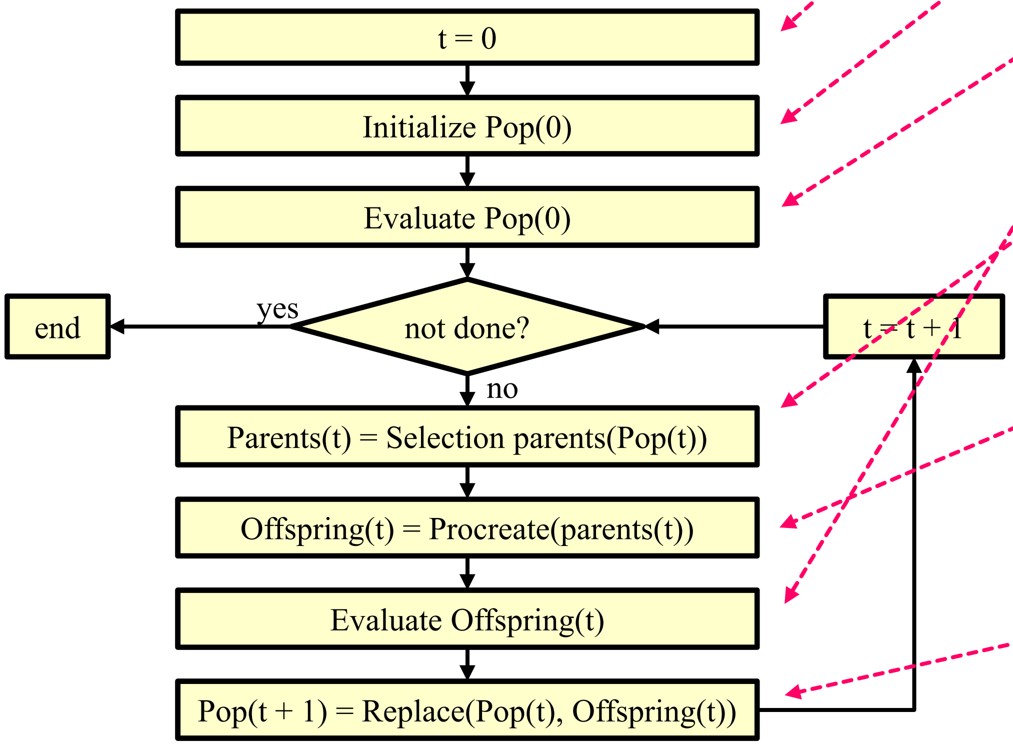


# 基因演算法流程圖

Max f(x,y)

$$= 0.5 + (\sin(\mathrm{sqrt}(x^2+y^2))^2 - 0.5)/(1.0 + 0.001(x^2+y^2))^2$$





t = 0

Initialize Pop(0)

Evaluate Pop(0)

not done?

yes

end

no

Parents(t) = Selection parents(Pop(t))

Offspring(t) = Procreate(parents(t))

Evaluate Offspring(t)

Pop(t + 1) = Replace(Pop(t), Offspring(t))

t = t + 1

**1. 編碼**： $(x_i,y_i)$

**初始化**：$(x_i,y_i)$隨機從[-100,100] × [-100,100]中取一點

**2. 適應度**： $fit_i = f(x_i,y_i)$

**3. 選擇**： 隨機從目前的population中選出一對父母$(x_{mom}, y_{mon})$和$(x_{dad}, y_{dad})$

**4. 繁衍**：

$x_{kid} = rnd(x_{mom}, x_{dad}) + N_x(0,\sigma)$

$y_{kid} = rnd(y_{mom}, y_{dad}) + N_y(0,\sigma)$

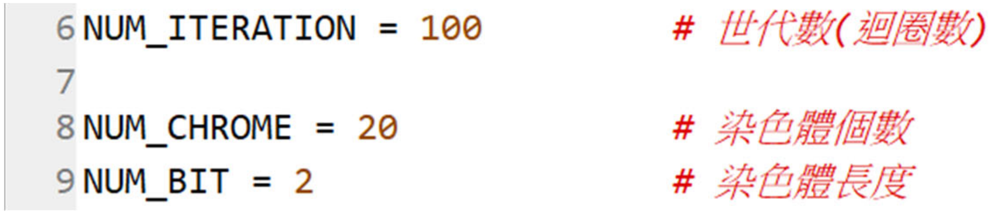
**5. 取代**：

Pop(t+1)

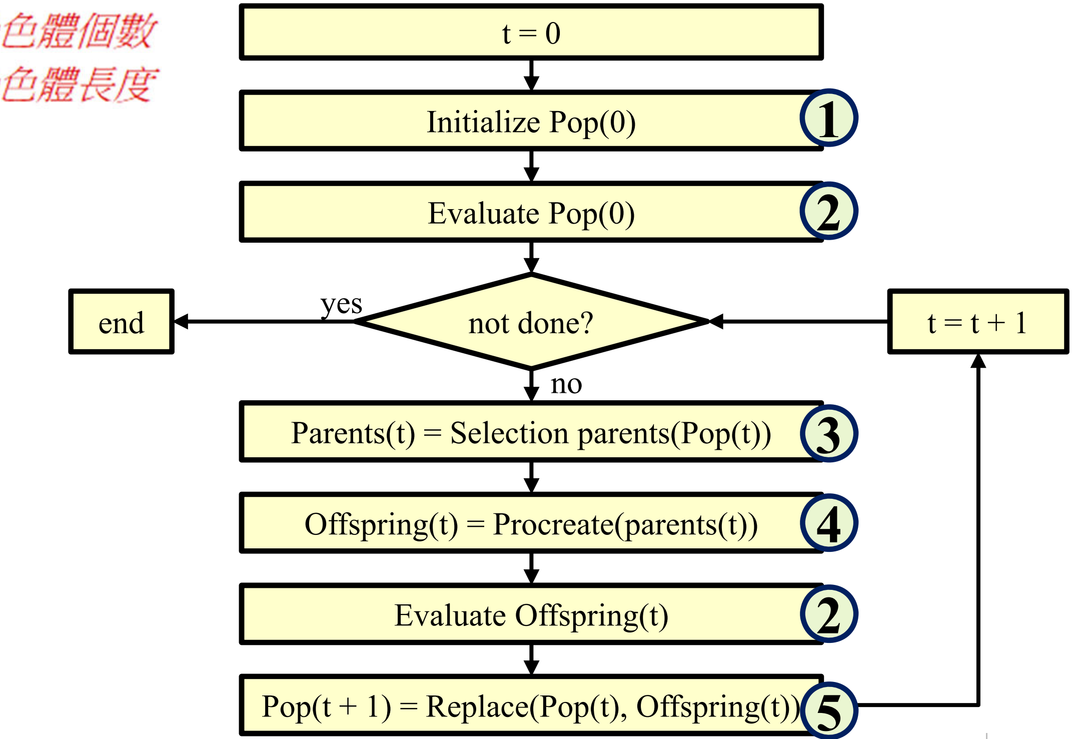
= {Pop(t) - {worst}} ∪ {kid}

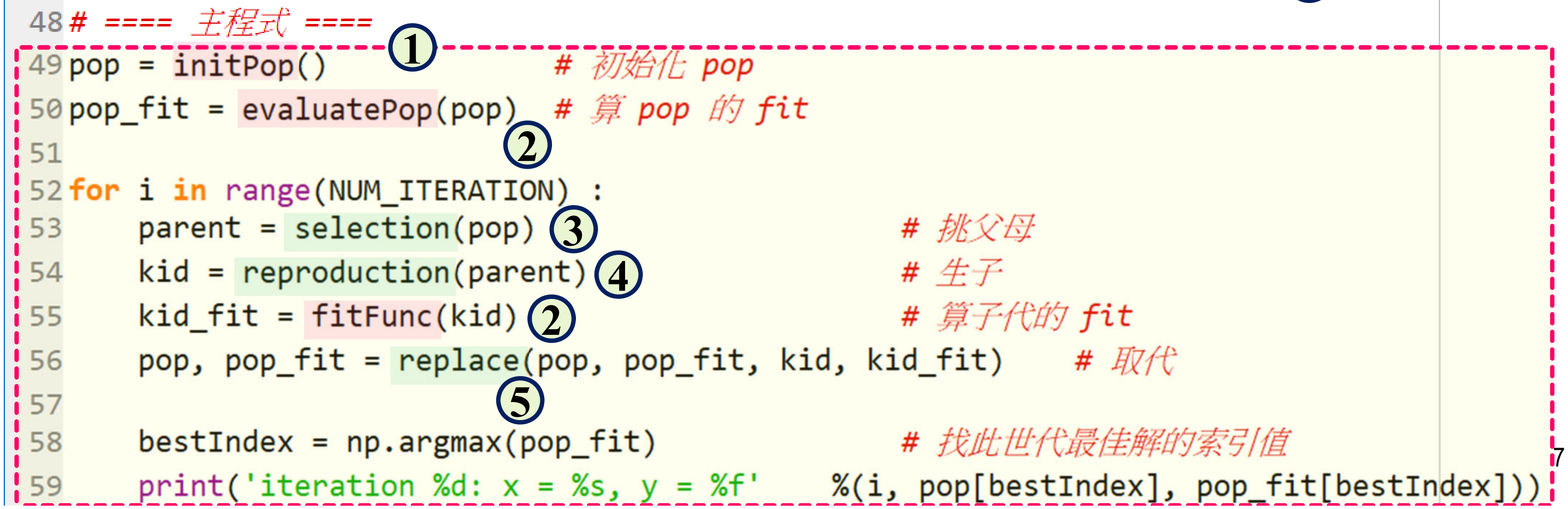
# A Simple Example

```
t = 0;
Initialize Pop(t); /* of P individuals */
Evaluate Pop(t);
while (t <= 4000){
  Select_Parent(<$x_{mom}$,$y_{mom}$>); /* Randomly */
  Select_Parent(<$x_{dad}$,$y_{dad}$>); /* Randomly */
  Create_Offspring(<$x_{kid}$,$y_{kid}$>):
  $x_{kid}$ = rnd($x_{mom}$, $x_{dad}$) + $N_x(0,\sigma)$;
  $y_{kid}$ = rnd($y_{mom}$, $y_{dad}$) + $N_y(0,\sigma)$;
  $fit_{kid}$ = Evaluate(<$x_{kid}$,$y_{kid}$>);
  Pop(t+1) = Replace(worst,kid);{Pop(t)-{worst}}$\cup${kid}
  t = t + 1;
}
```

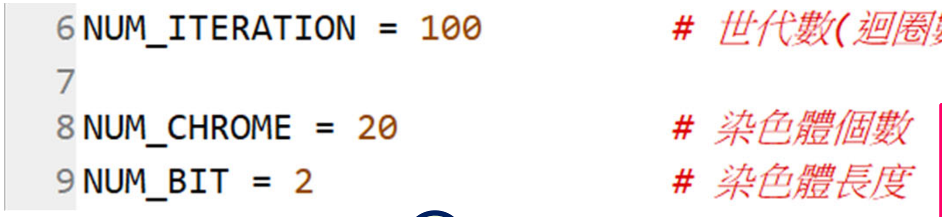


```
NUM_ITERATION = 100             # 世代數(迴圈數)

NUM_CHROME = 20                 # 染色體個數
NUM_BIT = 2                     # 染色體長度
```


t = 0
Initialize Pop(0)
1
Evaluate Pop(0)
2
not done?
yes
end
no
t = t + 1
Parents(t) = Selection parents(Pop(t))
3
Offspring(t) = Procreate(parents(t))
4
Evaluate Offspring(t)
2
Pop(t + 1) = Replace(Pop(t), Offspring(t))
5

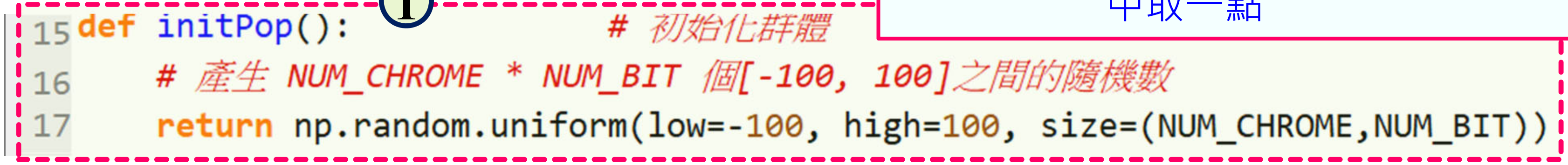


```
# ==== 主程式 ====
pop = initPop()                 # 初始化 pop
pop_fit = evaluatePop(pop)      # 算 pop 的 fit

for i in range(NUM_ITERATION) :
    parent = selection(pop)                         # 挑父母
    kid = reproduction(parent)                      # 生子
    kid_fit = fitFunc(kid)                          # 算子代的 fit
    pop, pop_fit = replace(pop, pop_fit, kid, kid_fit)     # 取代

    bestIndex = np.argmax(pop_fit)                  # 找此世代最佳解的索引值
    print('iteration %d: x = %s, y = %f'     %(i, pop[bestIndex], pop_fit[bestIndex]))
```



```
NUM_ITERATION = 100            # 世代數(迴圈數)

NUM_CHROME = 20                # 染色體個數
NUM_BIT = 2                    # 染色體長度
```

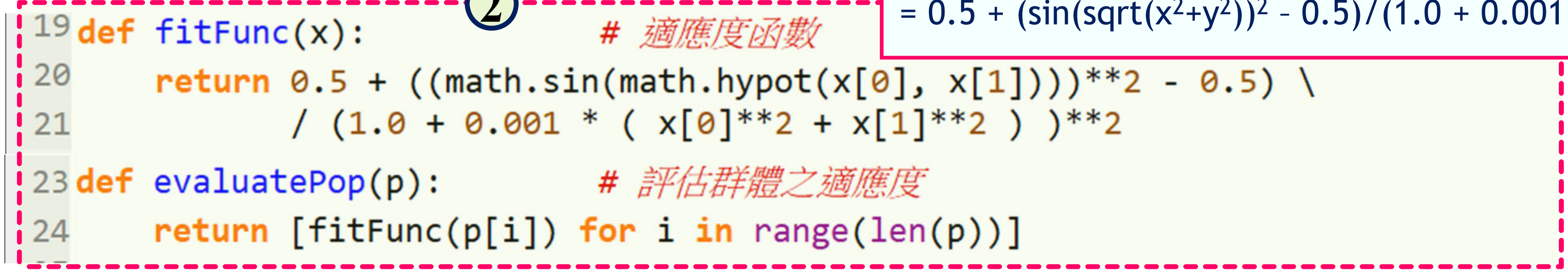
1. 編碼： $(x_i,y_i)$
初始化：$(x_i,y_i)$隨機從[-100,100] × [-100,100]
中取一點



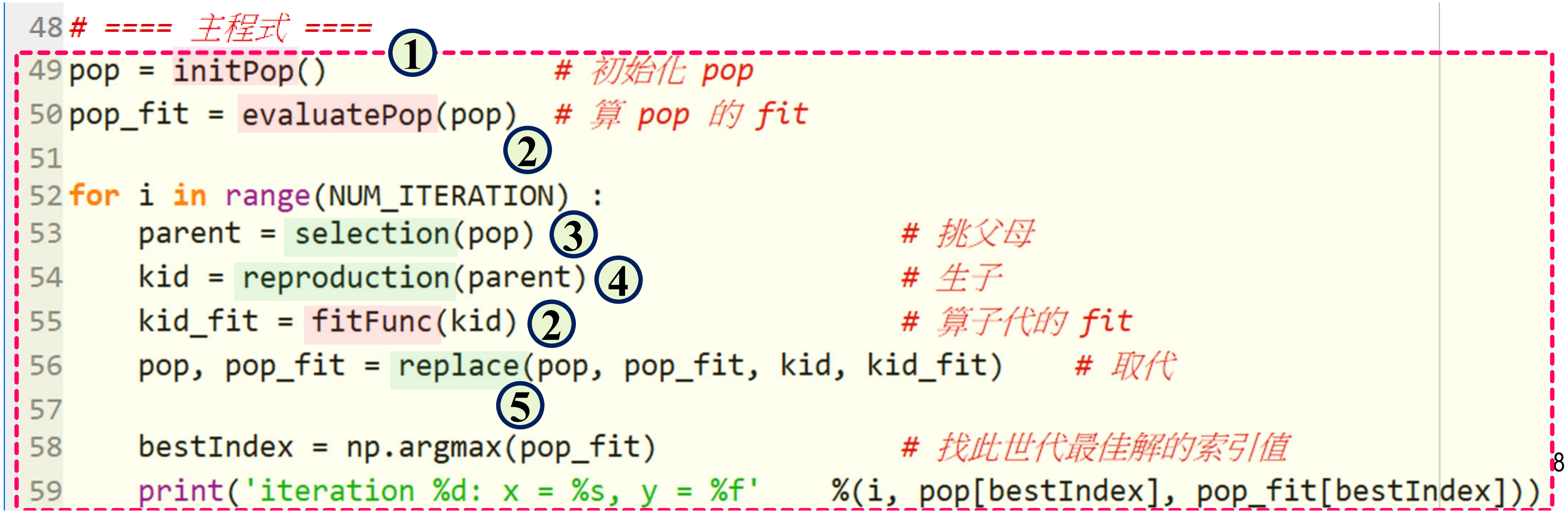
```
def initPop():  ①          # 初始化群體
    # 產生 NUM_CHROME * NUM_BIT 個[-100, 100]之間的隨機數
    return np.random.uniform(low=-100, high=100, size=(NUM_CHROME,NUM_BIT))
```

2. 適應度： Max f(x,y)

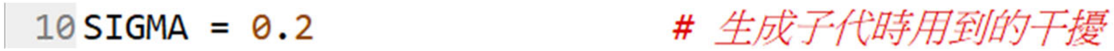
$= 0.5 + (\sin(\mathrm{sqrt}(x^2+y^2))^2 - 0.5)/(1.0 + 0.001(x^2+y^2))^2$



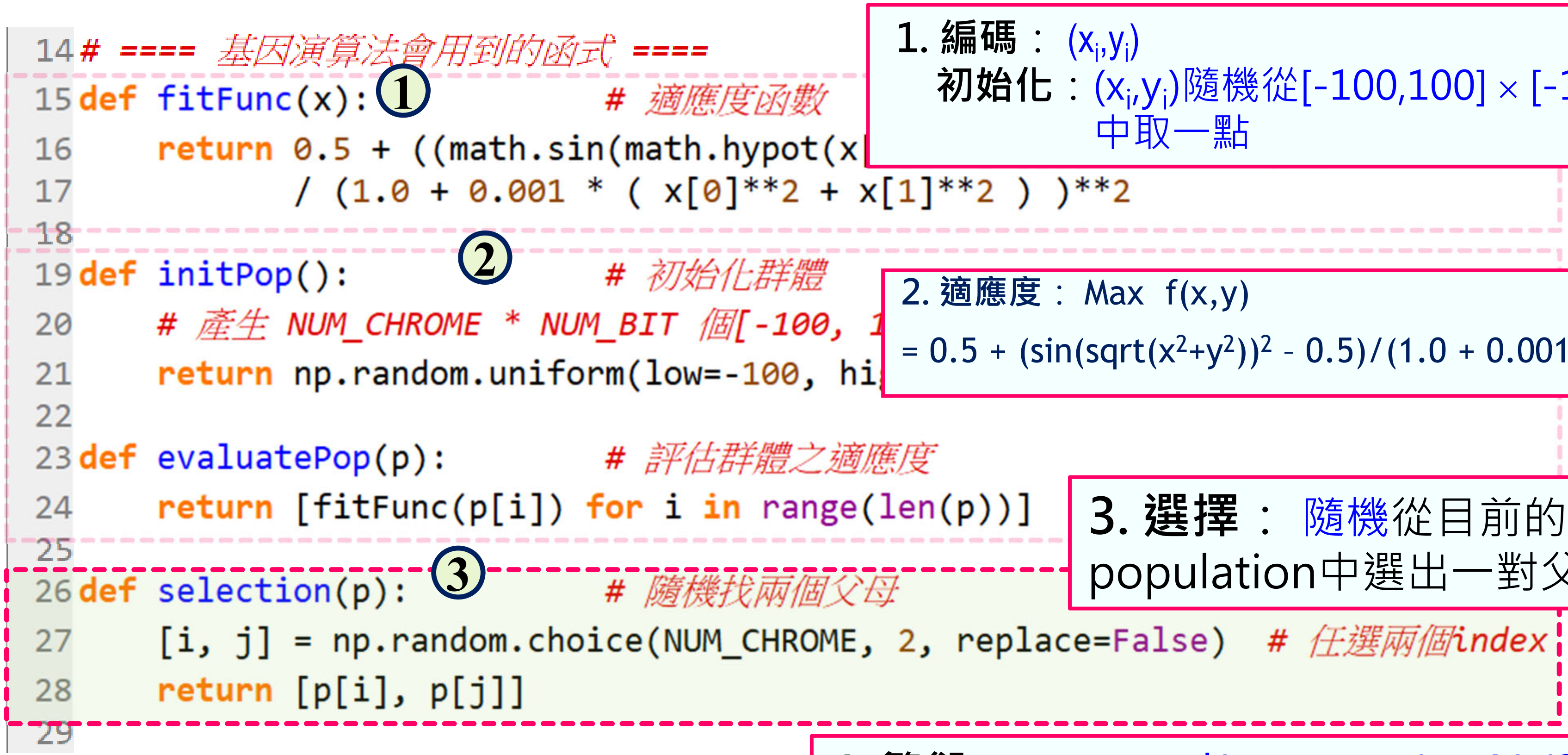
```
def fitFunc(x):  ②         # 適應度函數
    return 0.5 + ((math.sin(math.hypot(x[0], x[1])))**2 - 0.5) \
          / (1.0 + 0.001 * ( x[0]**2 + x[1]**2 ) )**2

def evaluatePop(p):         # 評估群體之適應度
    return [fitFunc(p[i]) for i in range(len(p))]
```



```
# ==== 主程式 ====
pop = initPop()  ①             # 初始化 pop
pop_fit = evaluatePop(pop)  ②  # 算 pop 的 fit

for i in range(NUM_ITERATION) :
    parent = selection(pop)  ③                 # 挑父母
    kid = reproduction(parent)  ④              # 生子
    kid_fit = fitFunc(kid)  ②                  # 算子代的 fit
    pop, pop_fit = replace(pop, pop_fit, kid, kid_fit)  ⑤    # 取代

    bestIndex = np.argmax(pop_fit)               # 找此世代最佳解的索引值
    print('iteration %d: x = %s, y = %f'    %(i, pop[bestIndex], pop_fit[bestIndex]))
```



```
SIGMA = 0.2                    # 生成子代時用到的干擾
```



```
# ==== 基因演算法會用到的函式 ====
def fitFunc(x): ①              # 適應度函數
    return 0.5 + ((math.sin(math.hypot(x
            / (1.0 + 0.001 * ( x[0]**2 + x[1]**2 ) )**2

def initPop(): ②               # 初始化群體
    # 產生 NUM_CHROME * NUM_BIT 個[-100, 1
    return np.random.uniform(low=-100, hi

def evaluatePop(p):            # 評估群體之適應度
    return [fitFunc(p[i]) for i in range(len(p))]

def selection(p): ③            # 隨機找兩個父母
    [i, j] = np.random.choice(NUM_CHROME, 2, replace=False)  # 任選兩個index
    return [p[i], p[j]]

```

1. **編碼**：$(x_i,y_i)$

**初始化**：$(x_i,y_i)$隨機從[-100,100] × [-100,100]中取一點

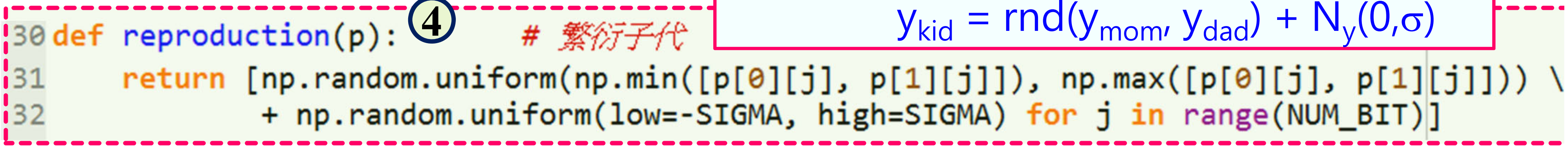
2. **適應度**：Max f(x,y)

$$= 0.5 + (\sin(\sqrt{x^2+y^2}))^2 - 0.5)/(1.0 + 0.001(x^2+y^2))^2$$

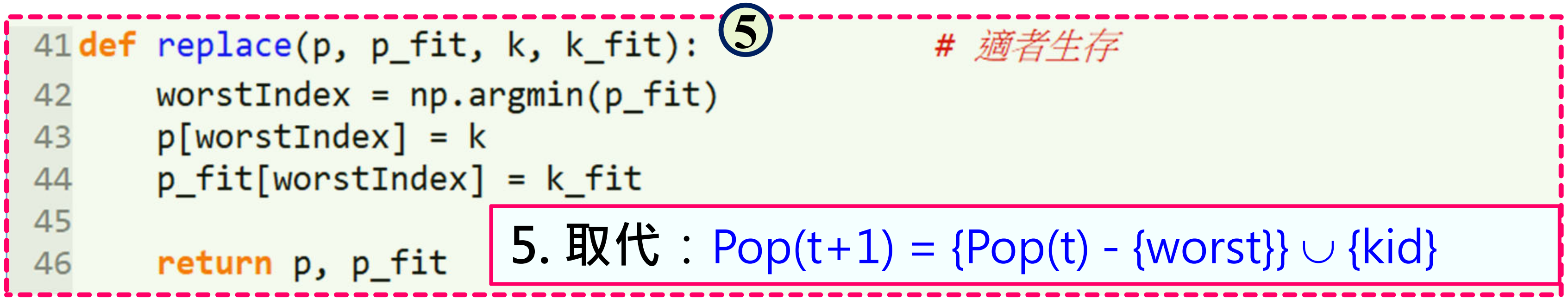
3. **選擇**：隨機從目前的population中選出一對父母

4. **繁衍**：$x_{kid} = rnd(x_{mom}, x_{dad}) + N_x(0,\sigma)$

$y_{kid} = rnd(y_{mom}, y_{dad}) + N_y(0,\sigma)$



```
def reproduction(p): ④       # 繁衍子代
    return [np.random.uniform(np.min([p[0][j], p[1][j]]), np.max([p[0][j], p[1][j]])) \
            + np.random.uniform(low=-SIGMA, high=SIGMA) for j in range(NUM_BIT)]
```

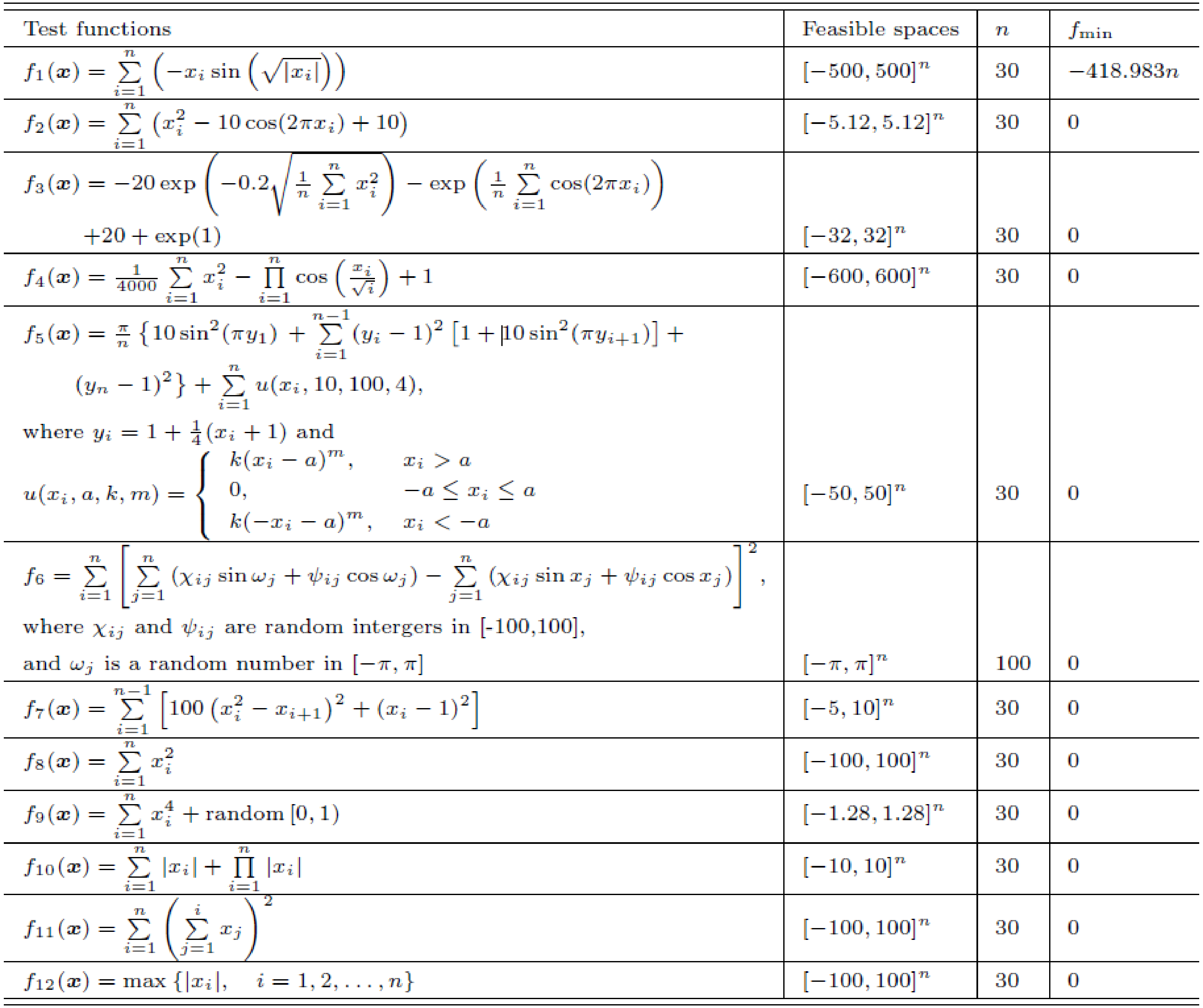


```
def replace(p, p_fit, k, k_fit): ⑤          # 適者生存
    worstIndex = np.argmin(p_fit)
    p[worstIndex] = k
    p_fit[worstIndex] = k_fit

    return p, p_fit
```

5. **取代**：Pop(t+1) = {Pop(t) - {worst}} ∪ {kid}

# Exercise

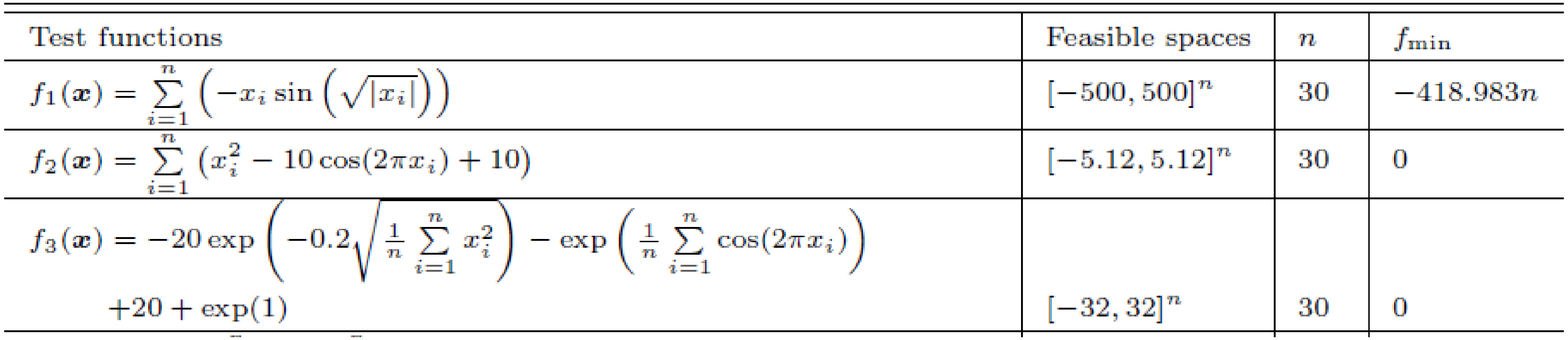
- Benchmark functions:



| Test functions | Feasible spaces | $n$ | $f_{\min}$ |
|---|---|---|---|
| $f_1(\boldsymbol{x}) = \sum_{i=1}^{n}\left(-x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$ | $[-500, 500]^n$ | 30 | $-418.983n$ |
| $f_2(\boldsymbol{x}) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$ | $[-5.12, 5.12]^n$ | 30 | 0 |
| $f_3(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$<br>$+20 + \exp(1)$ | $[-32, 32]^n$ | 30 | 0 |
| $f_4(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | $[-600, 600]^n$ | 30 | 0 |
| $f_5(\boldsymbol{x}) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + \right.$<br>$\left.(y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$<br>where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and<br>$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$ | $[-50, 50]^n$ | 30 | 0 |
| $f_6 = \sum_{i=1}^{n}\left[\sum_{j=1}^{n}(\chi_{ij}\sin\omega_j + \psi_{ij}\cos\omega_j) - \sum_{j=1}^{n}(\chi_{ij}\sin x_j + \psi_{ij}\cos x_j)\right]^2,$<br>where $\chi_{ij}$ and $\psi_{ij}$ are random intergers in [-100,100],<br>and $\omega_j$ is a random number in $[-\pi, \pi]$ | $[-\pi, \pi]^n$ | 100 | 0 |
| $f_7(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right]$ | $[-5, 10]^n$ | 30 | 0 |
| $f_8(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$ | $[-100, 100]^n$ | 30 | 0 |
| $f_9(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^4 + \text{random}\,[0, 1)$ | $[-1.28, 1.28]^n$ | 30 | 0 |
| $f_{10}(\boldsymbol{x}) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$ | $[-10, 10]^n$ | 30 | 0 |
| $f_{11}(\boldsymbol{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$ | $[-100, 100]^n$ | 30 | 0 |
| $f_{12}(\boldsymbol{x}) = \max\left\{\lvert x_i\rvert, \quad i = 1, 2, \ldots, n\right\}$ | $[-100, 100]^n$ | 30 | 0 |

# Exercise (cont.)

- Use the GA sample code “GA-simplified” to find the optimal solutions of the following three functions:



| Test functions | Feasible spaces | $n$ | $f_{\min}$ |
|---|---|---|---|
| $f_1(\boldsymbol{x}) = \sum_{i=1}^{n} \left(-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$ | $[-500, 500]^n$ | 30 | $-418.983n$ |
| $f_2(\boldsymbol{x}) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$ | $[-5.12, 5.12]^n$ | 30 | 0 |
| $f_3(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$ $+20 + \exp(1)$ | $[-32, 32]^n$ | 30 | 0 |

- Note that it is very hard to find the optimal solutions when n = 30. Hence, when testing your program, you can check whether it can find the optimal solution when n = 1.

- Hint
  - 1. (編碼) 改成30維度
  - 2. (編碼) 改初始化群組的範圍
  - 3. (解碼) 改適應度函數成$f_1$, $f_2$, $f_3$
  - 4. (輸出) 不要輸出x，只要看y = f(x)