1．用加步探索法确定一维最优化问题$min_{t\geq0}\varphi(t)=t^3-2t+1$的搜索区间，要求选取$t_0=0,h_0=1,a=2$.

```
import numpy as np

t0 = 0
h0 = 1
a = 2
k = 0
t = t0
phi_t = lambda t: t**3 - 2*t + 1

while phi_t(t + h0) < phi_t(t) or k == 0 :
    if phi_t(t + h0) < phi_t(t):
        k += 1
        print("在第",k,"轮迭代中,t_",k,"=",t,",phi(t_",k,") =",phi_t(t),
              ",t_",k+1," = ",t + h0,",phi(t_" , k+1, ") = ",phi_t(t + h0),",h0 = ",h0)
        t0 = t
        t = t + h0
        h0 = a * h0
    else:
        h0 = -1 * h0
        t = t + h0

a = min(t0, t + h0)
b = max(t0, t + h0)
search_interval = [a, b]
print("最终的搜索区间为:",search_interval)
```

打印结果为:

在第 1 轮迭代中,t_ 1 = 0 ,phi(t_ 1) = 1 ,t_ 2 = 1 ,phi(t_ 2) = 0 ,h0 = 1

最终的搜索区间为: [0, 3]

2．用对分法求解$min\varphi(t) = t(t-3)$,已知初始单谷区间[a,b] = [-3,5]，按精度$\epsilon = 0.1$计算．

```
def phi_t(t):
    return t * (t - 3)

def phi_t_prime(t):
    return 2 * t - 3

a = -3
b = 5
epsilon = 0.1
count = 0
while abs(b - a) > epsilon:
    c = (a + b) / 2
    if phi_t_prime(c) == 0:
        count += 1
        print("第",count,"次迭代,搜索区间为:",[a, b])
        a = b = c
        break
    elif phi_t_prime(a) * phi_t_prime(c) < 0:
        count += 1
        print("第",count,"次迭代,搜索区间为:",[a, b])
        b = c
    else:
        count += 1
        print("第",count,"次迭代,搜索区间为:",[a, b])
        a = c

optimal_t = (a + b) / 2
optimal_t
```

打印结果为:

第 1 次迭代,搜索区间为: [-3, 5]

第 2 次迭代,搜索区间为: [1.0, 5]

第 3 次迭代,搜索区间为: [1.0, 3.0]

第 4 次迭代,搜索区间为: [1.0, 2.0]

t = 1.5

3. 用Newton法求解$min_{t\geq0}\varphi(t)=t^3-2t+1$,用第1题求得的区间，按精度$\epsilon=0.01$计算.

```python
def phi_t(t):
    return t**3 - 2*t + 1

def derivative_phi_t(t):
    return 3*t**2 - 2

def second_derivative_phi_t(t):
    return 6*t

a = search_interval[0]
b = search_interval[1]
epsilon = 0.01

t0 = (a + b) / 2
t = t0 - derivative_phi_t(t0) / second_derivative_phi_t(t0)
iteration = 1
print(f"Iteration {iteration} : t = {round(t,4)},phi(t) = {round(phi_t(t),5)}")

while abs(t - t0) > epsilon:
    t0 = t
    t = t - derivative_phi_t(t) / second_derivative_phi_t(t)
    iteration += 1
    print(f"Iteration {iteration} : t = {round(t,4)},phi(t) = {round(phi_t(t),5)}")

optimal_t = t
optimal_t
```

打印结果为:

Iteration 1 : t = 0.9722,phi(t) = -0.02548

Iteration 2 : t = 0.829,phi(t) = -0.08828

Iteration 3 : t = 0.8166,phi(t) = -0.08866

Iteration 4 : t = 0.8165,phi(t) = -0.08866

t = 0.8164965863169821

4．用黄金分割法法求解$min\varphi(t)=t(t+2)$,已知初始单谷区间[a,b] = [-3,5]，按精度$\epsilon=0.001$计算．

```
def phi_t(t):
    return t * (t + 2)

a = -3
b = 5
epsilon = 0.001

# Golden section formula
golden_ratio = (1 + 5 ** 0.5) / 2

t1 = a + (b - a) / golden_ratio
t2 = b - (b - a) / golden_ratio

iteration = 1

while abs(t1 - t2) > epsilon:
    print(f"Iteration {iteration}: a = {round(a,3)}, b = {round(b,3)},
    t2 = {round(t2,3)}, t1 = {round(t1,3)}")
    if phi_t(t1) <= phi_t(t2):
        a = t2
        t2 = t1
        t1 = a + (b - a) / golden_ratio
    else:
        b = t1
        t1 = t2
        t2 = b - (b - a) / golden_ratio

    iteration += 1

print(f"Final result: t = {round((t1+t2)/2,5)}")
```

打印结果为:

Iteration 1: a = -3, b = 5,t2 = 0.056, t1 = 1.944

Iteration 2: a = -3, b = 1.944,t2 = -1.111, t1 = 0.056

Iteration 3: a = -3, b = 0.056,t2 = -1.833, t1 = -1.111

Iteration 4: a = -1.833, b = 0.056,t2 = -1.111, t1 = -0.666

Iteration 5: a = -1.833, b = -0.666,t2 = -1.387, t1 = -1.111

Iteration 6: a = -1.387, b = -0.666,t2 = -1.111, t1 = -0.941

Iteration 7: a = -1.111, b = -0.666,t2 = -0.941, t1 = -0.836

Iteration 8: a = -1.111, b = -0.836,t2 = -1.006, t1 = -0.941

Iteration 9: a = -1.111, b = -0.941,t2 = -1.046, t1 = -1.006

Iteration 10: a = -1.046, b = -0.941,t2 = -1.006, t1 = -0.981

Iteration 11: a = -1.046, b = -0.981,t2 = -1.022, t1 = -1.006

Iteration 12: a = -1.022, b = -0.981,t2 = -1.006, t1 = -0.997

Iteration 13: a = -1.006, b = -0.981,t2 = -0.997, t1 = -0.991

Iteration 14: a = -1.006, b = -0.991,t2 = -1.0, t1 = -0.997

Iteration 15: a = -1.006, b = -0.997,t2 = -1.003, t1 = -1.0

Iteration 16: a = -1.003, b = -0.997,t2 = -1.0, t1 = -0.999

Final result: t = -1.00077

5．用抛物线插值法求解$min f(x) = 8x^3 - 2x^2 - 7x + 3$，已知初始单谷区间[a,b] = [0, 2]，按精度$\epsilon = 0.001$计算

```
def f(x):
    return 8*x**3 - 2*x**2 - 7*x + 3

a = 0
b = 2
epsilon = 0.001

t0 = (a + b) / 2
tbar = ( (t0**2-b**2)* f(a) + (b**2 - a**2)*f(t0) + (a**2-t0**2)*f(b) )/(
    2 * ((t0-b)* f(a) + (b - a)*f(t0) + (a-t0)*f(b)))

iteration = 1

while abs(t0 - tbar) > epsilon:

    if f(tbar) <= f(t0) and tbar < t0:
        b = t0
        t0 = tbar
    elif f(tbar) <= f(t0) and tbar > t0:
        a = t0
        t0 = tbar
    elif f(tbar) > f(t0) and tbar < t0:
        a = tbar
    elif f(tbar) > f(t0) and tbar > t0:
        b = tbar

    tbar = ( (t0**2-b**2)* f(a) + (b**2 - a**2)*f(t0) + (a**2-t0**2)*f(b) )/(
    2 * ((t0-b)* f(a) + (b - a)*f(t0) + (a-t0)*f(b)) )
    print(f"Iteration {iteration}: [a, b] = [{round(a,3)}, {round(b,3)}], t* = {round(tbar,3)},

    iteration += 1

print(f"Final result: t* = {round(tbar,5)}, function value ={round(f(tbar),5)}]")
```

打印结果为:

Iteration 1: [a, b] = [0, 1.0], t* = 0.549, f(t*) = -0.122

Iteration 2: [a, b] = [0.523, 1.0], t* = 0.613, f(t*) = -0.2

Iteration 3: [a, b] = [0.549, 1.0], t* = 0.621, f(t*) = -0.202

Iteration 4: [a, b] = [0.613, 1.0], t* = 0.627, f(t*) = -0.203

Iteration 5: [a, b] = [0.621, 1.0], t* = 0.629, f(t*) = -0.203

Iteration 6: [a, b] = [0.627, 1.0], t* = 0.629, f(t*) = -0.203

Final result: x* = 0.62947, function value f(x*)=-0.20342