1 1: ,

:

$$f(x) = 3x^2 - 2x - 2, \quad x \in [-1, 1]$$

1.1

:

$$\frac{\sqrt{5} - 1}{2} \approx 0.618$$

c d, , .
Python:

```
blueimport numpy blueas np

bluedef f(x):
    bluereturn 3*x**2 - 2*x - 2

bluedef golden_section_search(a, b, tol=1e-5):
    gr = (np.sqrt(5) - 1) / 2
    c = b - gr * (b - a)
    d = a + gr * (b - a)

    bluewhile blueabs(b - a) > tol:
        blueif f(c) < f(d):
            b = d
        blueelse:
            a = c

        c = b - gr * (b - a)
        d = a + gr * (b - a)

    bluereturn (b + a) / 2

green#green green                green green       green green
    green green
a, b = -1, 1
x_min = golden_section_search(a, b)
blueprint(fred"red               red␣red               red:red␣red{
    redx_minred},red␣redfred(redx_minred)red␣red=red␣red{redfred(redx_min
    red)red}red")
```

1.2

δ . , .

Python:

```
bluedef dichotomy_search(a, b, tol=1e-5, delta=1e-6):
    bluewhile blueabs(b - a) > tol:
        x1 = (a + b) / 2 - delta
        x2 = (a + b) / 2 + delta

        blueif f(x1) < f(x2):
            b = x2
        blueelse:
            a = x1

    bluereturn (a + b) / 2

green#green green               green green               green
    green
x_min_dichotomy = dichotomy_search(a, b)
blueprint(fred"red              red␣red              red␣
    red              red␣red                  red:red␣red{
    redx_min_dichotomyred},red␣redfred(redx_minred)red␣red=red␣red{redf
    red(redx_min_dichotomyred)red}red")
```

1.3

( ). .

Python:

```
bluedef coordinate_search(x0, tol=1e-5, alpha=0.1):
    x = x0
    step = alpha
    bluewhile blueabs(step) > tol:
        f_left = f(x - step)
        f_right = f(x + step)

        blueif f_left < f(x):
            x = x - step
        blueelif f_right < f(x):
            x = x + step
        blueelse:
            step *= 0.5

    bluereturn x

green#green green                   green green           green green
    green green
x0 = 0
x_min_coordinate = coordinate_search(x0)
blueprint(fred"red              red␣red              red␣
    red                         red␣red             red:red␣red{
    redx_min_coordinatered},red␣redfred(redx_minred)red␣red=red␣red{redf
    red(redx_min_coordinatered)red}red")
```

2 2:

:

$$f(x) = 3x^3 + 6x^2 + x + 1, \quad x \in [-3, 3]$$

, . .

Python:

```
bluedef f_cubic(x):
    bluereturn 3*x**3 + 6*x**2 + x + 1

bluedef parabola_search(a, b, tol=1e-5):
    x1, x2, x3 = a, (a + b) / 2, b

    bluewhile blueabs(b - a) > tol:
        f1, f2, f3 = f_cubic(x1), f_cubic(x2), f_cubic(x3)

        num = (x2**2 - x3**2)*f1 + (x3**2 - x1**2)*f2 + (x1**2 - x2
            **2)*f3
        denom = (x2 - x3)*f1 + (x3 - x1)*f2 + (x1 - x2)*f3

        x_min = 0.5 * num / denom

        blueif f_cubic(x_min) < f_cubic(x2):
            blueif x_min < x2:
                x3 = x2
            blueelse:
                x1 = x2
            x2 = x_min
        blueelse:
            blueif x_min < x2:
                x1 = x_min
            blueelse:
                x3 = x_min

    bluereturn x_min

green#green green           green green                  green
    green               green green
a, b = -3, 3
x_min_parabola = parabola_search(a, b)
blueprint(fred"red              red␣red              red␣
    red              red␣red              red:red␣red{redx_min_parabola
    red},red␣redfred(redx_minred)red␣red=red␣red{redf_cubicred(
    redx_min_parabolared)red}red")
```