Flow chart:

Maple code (for \( f(x) = \cos(x) - x = 0, \Phi(x) = \cos(x) \)):

```maple
simple_iter:=proc(x,tol)
    local xold,xnew,dif,phi;
    # transcription of input
    xold:=x;
    # ensures that loop starts
    dif:=tol+1;
    # iterative map
    phi:=z->cos(z);
    # iteration loop
    while dif>tol do
        # iteration
        xnew:=phi(xold);
        # error estimate
        dif:=abs(xnew-xold);
        # transcription
        xold:=xnew;
        end do;
    return xnew;
end proc:
```

Problem: loops without a counter may run forever.

b) Bisection algorithm
Consider a (continuous) function $f$ where $f(a) < 0$ and $f(b) > 0$ (or vice versa). Then $f(x) = 0$ for some $x$, $a < x < b$ by the intermediate value theorem. How to compute such a value?

Idea: Compute the midpoint $c = (a + b)/2$. If $f(b)$ and $f(c)$ have the same sign then $x$ is contained in (the smaller) interval $[a, c]$. If $f(a)$ and $f(c)$ have the same sign then $x$ is contained in $[c, b]$. Repeat the step with the smaller interval until its length is below the required threshold. In each step the length of the intervals halves, i.e., after $n$ steps the length of the interval is $|b - a|/2^n$. Thus the method always converges!

Structure of the code:

- Given: lower limit $a$, upper limit $b$, and threshold $\varepsilon$.
- Check whether $f(a)$ and $f(b)$ have opposite sign.
- Compute the midpoint $c = (a + b)/2$.
- If $f(a)$ and $f(c)$ have the same sign replace the lower limit $a$ by $c$ (to continue with $[c, b]$!), else replace the upper limit $b$ by $c$ (to continue with $[a, c]$!)
- If the length of the interval $|b - a|$ is larger than the threshold continue with step 3, else return the midpoint.

Flow chart:
Maple code (for $f(x) = \cos(x) - x = 0$):

```maple
simple_bisection:=proc(a,b,tol)
    local low,up,mid,f;
    f:=x->cos(x)-x;
    # transcription of input
    low:=a;
    up:=b;
    # check for sign change
    if f(up)*f(low)>0 then
        error "f(a) f(b)>0";
    end if;
    # recursive bisection
    while up-low>tol do
        # midpoint
        mid:=(up+low)/2;
        # bisection update
        if f(low)*f(mid)>0 then
            low:=mid;
        else
            up:=mid;
        end if;
    end do;
    # output
    return (up+low)/2;
end proc:
```

**Example 3.3:** Suppose you start a bisection algorithm with initial interval $[-2, 2]$ and the algorithm returns $1/2$. How many bisections have been performed?

The first bisection step computes the midpoint value 0. As our result is contained in $[0, 2]$ the algorithm continues with the interval $[0, 2]$ and the second bisection step computes for the midpoint the value 1. As our result is contained in $[0, 1]$ the algorithm returns the
midpoint value 1/2, i.e., two bisection steps have been performed.

c) Newton-Raphson method

Remark: on Taylor series expansions

\[
 f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \ldots
\]

\[
 f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{R(x, x_0)}{x - x_0}
\]

\[\varphi(x), \text{ linear in } x \] remainder ("small")

Graphically:

We want to compute \(x\) such that \(f(x) = 0\). Suppose we know some initial guess \(x_0\) (close to \(x\)). Replace \(f\) by the first order approximation \(\varphi\) and solve \(\varphi(x) = 0\), i.e.,

\[
 0 = f(x_0) + f'(x_0)(x - x_0) \quad \Rightarrow \quad x = x_0 - \frac{f(x_0)}{f'(x_0)} = x_1.
\]

Thus we obtain an improved value \(x_1\). Repeat the procedure until the sequence converges, i.e.,

\[
x_{k+1} = \Phi(x_k) = x_k - \frac{f(x_k)}{f'(x_k)}.
\]

Iterative solution method with some "intelligent" iterative map \(\Phi\).

\(\Phi(x) = x - f(x)/f'(x)\) is called Newton-Raphson map. Properties of \(\Phi\) (see §3a)

- "Fixed point condition": if \((x_0, x_1, x_2, \ldots)\) converges, \(x_n \to x_*\) then

\[
x_* = \Phi(x_*) = x_* - \frac{f(x_*)}{f'(x_*)} \quad \Rightarrow \quad f(x_*) = 0
\]

i.e., the limit is the desired solution.