



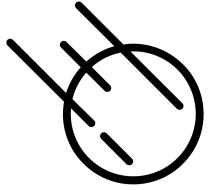
Putting it All Together

- Multi-paradigm programming means picking and choosing from our various paradigms,
 - Imperative
 - Declarative with pattern matching
 - Functional
 - OOP
 - First-class patterns
- To create the most readable and maintainable programs.



Case Study: QuickSort

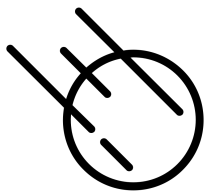
- We start with the imperative and the functional versions of the quicksort
 - Examining both the strengths and weaknesses of each approach
- We then pick and choose from each of these implementations and create a multi-paradigm version of the quicksort.
- Finally, we'll create some extensions such as a flexible sorting predicate based on higher-order programming.



Imperative Programming

```
-- imperative version of the quicksort
function qsort with a do
  if len(a) <= 1 do
    return a
  else do
    let pivot = a@0.
    let rest = a@(range(1, len(a))).
    let less = [].
    let more = [].
    for e in rest do
      if e <= pivot do
        less @append(e).
      else
        more @append(e).
      end
    end
    return qsort(less) + [pivot] + qsort(more).
  end
end

assert(qsort([3,7,1,6,9,5,2,10,8,4]) == [1,2,3,4,5,6,7,8,9,10]).
```



Functional Programming

```
-- functional version of the quicksort
function qsort
  with [] do
    []
  with [a] do
    [a]
  with [pivot|rest] do
    function filter
      with ([],_) do
        ([],[])
      with ([e|rest],pivot) do
        let (a,b) = filter (rest,pivot).
        return ([e]+a,b) if e <= pivot else (a,[e]+b).
      end
    let (less,more) = filter (rest,pivot).
    qsort less + [pivot] + qsort more.
  end
end

assert (qsort [3,7,1,6,9,5,2,10,8,4] == [1,2,3,4,5,6,7,8,9,10]).
```



Multi-Paradigm Programming

```
-- multi-paradigm version of the quicksort
```

```
function qsort
```

```
  with [] do
```

```
    []
```

```
  with [a] do
```

```
    [a]
```

```
  with [pivot|rest] do
```

```
    let less = [].
```

```
    let more = [].
```

```
    for e in rest do
```

```
      if e <= pivot do
```

```
        less @append e.
```

```
      else do
```

```
        more @append e.
```

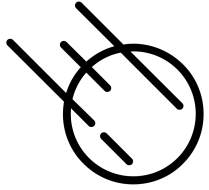
```
      end
```

```
    end
```

```
    qsort less + [pivot] + qsort more.
```

```
end
```

```
assert (qsort [3,7,1,6,9,5,2,10,8,4] == [1,2,3,4,5,6,7,8,9,10]).
```



Higher-Order Programming

In017/qhigh.ast

```
-- higher-order programming version of the quicksort
function qsort
  with ([],%function) do
    []
  with ([a],%function) do
    [a]
  with ([pivot|rest],order:%function) do
    let less = [].
    let more = [].
    for e in rest do
      if order (e,pivot) do ←
        less @append e.
      else do
        more @append e.
      end
    end
    qsort (less,order) + [pivot] + qsort (more,order).
  end
end

assert (qsort ([2,5,1,3,4],lambda with (a,b) do a<=b) == [1,2,3,4,5]).
```