UCI Division of Continuing Education

Module 1: Functions and Organization
Topic 2.2: Guidelines for Functions

Function Naming

- Give functions a good name
  - Behavior can be understood at a glance
  - Parameter naming counts too

```
func ProcessArray (a []int)
      float {}
func ComputeRMS (samples
      []float) float {}
```

- RMS = Root Mean Square
- `samples` is a slice of samples of a time-varying signal

UCI Division of Continuing Education

Functional Cohesion

- Function should perform **only one "operation"**
- An "operation" depends on the context
- Example: Geometry application
- Good functions:
  - `PointDist(), DrawCircle(), TriangleArea()`
- Merging behaviors makes code complicated
  - `DrawCircle() + TriangleArea()`

UCI Division of Continuing Education

Few Parameters

- Debugging requires tracing function input data
- More difficult with a large number of parameters
- Function may have bad functional cohesion
  - `DrawCircle()` and `TriangleArea()` require different arguments

Reducing Parameter Number

- May need to group related arguments into structures
- `TriangleArea()`, bad solution
  - 3 points needed to define triangle
  - Each point has 3 floats (in 3D)
  - Total, 9 arguments
- `TriangleArea()`, good solution

```
type Point struct{x, y, z float}
```

  - Total, 3 arguments