Interface Design

The Interface of a Function

A ***interface*** of a function is a summary of how it is used:

- What are the parameters?
- What does the function do?
- What is the return value?

Interface Design

- A function's interface should be as simple as possible, but not simpler (Einstein).
- The interface is like a contract between the designer of the function and its users.

Turtle Graphics Functions

Let's design a collection of functions that let us draw graphical shapes. We want our functions to be as general as possible.

- square()
- polygon()
- circle()
- arc()
- polyline()

square() -- Second Try

```
def square(t, l):
"""Uses the turtle t to draw a square with
sides of length l
"""
   for i in range(4):
      fd(t, l)
      lt(t)
```

- Design: Useful for drawing squares. What about other shapes?

square() -- First Try

```
def square(t):
"""Uses the turtle t to draw a square"""
   for i in range(4):
      fd(t, 100)
      lt(t)
```

- Documentation: The ***docstring*** ("""...""") explains the interface.
- Design: Limited usefulness – it can't be used to draw different size squares.

pentagon()

```
def pentagon(t, l):
"""Uses turtle t to draw a pentagon with
sides of length l
"""
   for i in range(5):
      fd(t, l)
      lt(t, 72)
```

- Design: Useful for drawing pentagons.

triangle()

```
def triangle(t, l):
"""Uses the turtle t to draw an equilateral
triangle with sides of length l
"""
   for i in range(3):
      fd(t, l)
      lt(t, 120)
```

- Design: Useful for drawing equilateral triangles.

Approximating a Circle

- A square is a (bad) approximation of a circle.
- A pentagon is an (only slightly) better approximation.
- A 120-sided polygon is a pretty good approximation.

```
polygon(bob, 120,3)
```

- Each side of the polygon is 3 pixels in length
- Circumference = 120 * 3 = 360 pixels
- Circumference = 2 * pi * r
- r = Circumference / (2 * pi) = (120 * length) / (2 * pi)

polygon() -- Generalization

```
def polygon(t, n, l):
"""Uses the turtle t to draw a regular n-
sided polygon with sides of length l
"""
   angle = 360.0 / n  # Amount to turn
   for i in range(n):
      fd(t, l)
      lt(t, angle)
```

• Design: This is appropriately general. It can draw triangles, squares, pentagons, etc.

circle(): interface design

```
def circle(t, r):
""" Draws a circle of radius r"""
   length = 3
   sides = 120
   length = circumference / sides
   polygon(t, sides, length)
```

• Limitation: Sides is fixed, so for big circles the length of the sides may be too big.

• Design: Should we add a 3rd parameter and give the user more control: circle(t,r,n)??

circle()

```
def circle(t, r):
""" Draws a circle of radius r"""
   circumference = 2 * 3.14159 * r
   sides = 120
   length = circumference / sides
   polygon(t, sides, length)
```

• Design: A circle is a special kind of polygon, one with lots of (120) sides.

arc() : Design

- An ***arc*** can be a ***generalization*** of circle.
- A circle is an arc of 360 degrees.
- circumference = 2 * 3.14159 * r
- arc_length = Circumference * angle / 360
- sides = int(arc_length / 3) + 1
- step_length = arc_length / sides
- step_angle = float(angle / sides)

```
def arc(t, r, angle):
    arc_length = 2 * 3.14159 * r * angle / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n

    for i in range(n):
        fd(t, step_length)
```


arc() : 1st Try

- An ***arc*** can be a ***generalization*** of circle.
- A circle is an arc of 360 degrees.

```
def arc(t, r, angle):
    arc_length = 2 * 3.14159 * r * angle / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n

    for i in range(n):      # Looks like polygon
        fd(t, step_length)
        lt(t, step_angle)
```


circle() : 2nd Try

```
def circle(t, r):
""" Draws a circle of radius r"""
   circumference = 2 * 3.14159 * r
   length = 3
   sides = int(circumference / 3) + 1
   polygon(t, sides, length)
```

- Length is fixed at 3, number of sides varies.
- **circle(t,r)** is cleaner than **circle(t,r,n)**

arc() : Design

- An ***arc*** can be a ***generalization*** of circle.
- A circle is an arc of 360 degrees.
- circumference = 2 * 3.14159 * r
- arc_length = Circumference * angle / 360
- n = int(arc_length / 3) + 1
- step_length = arc_length / n
- step_angle = float(angle / n)

polyline() : Design

```
def polyline (t, n, length, angle):
"""Draws a line consisting of n segments of length,
length, with a turn of angle after each segment
"""
    for i in range(n):
        fd(t, length)
        lt(t, angle)
```

- polyline(bob, 100, 3, 0) # straight line
- polyline(bob, 100, 3, 360) # straight line
- polyline(bob, 4, 100, 360/4) # square with side 100
- polyline(bob, 5, 100, 360/5) # pentagon
- polyline(bob, 8, 100, 360/8) # octagon

arc() : Design

- Can't use polygon without adding an ***angle*** parameter, but that would ruin it.
- A polygon is nicely defined by number of sides and length of side.

```
def arc(t, r, angle):
    arc_length = 2 * 3.14159 * r * angle / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n

    for i in range(n):      # Looks like polygon
        fd(t, step_length)
        lt(t, step_angle)
```


arc() : Refactored

- arc(t,r,angle) is a special case polyline()
- arc_length is a portion of a circumference: angle/360
- n is the total number of line segments of length 3
- step_length is approximately 3
- step_angle is the angle / n

```
def arc(t, r, angle):
    arc_length = 2 * math.pi * r * angle / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n
    polyline(t, n, step_length, step_angle)
```


polygon() : Refactored

- Polygon() is now trivial to define as a special case of polyline.

```
def polygon (t, n, length):
   polyline(t, n, length, 360.0/n)
```

Shapes Library

```
def circle(t, r):
    arc(t,r,360)
```

```
def polygon(t, n, length):
  polyline(t,n,length,360.0/n)
```

```
def arc(t, r, angle):
    arc_length = 2 * math.pi * r * angle / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n
    polyline(t, n, step_length, step_angle)
```

```
def polyline (t, n, length, angle):
    for i in range(n):
        fd(t, length)
        lt(t, angle)
```


Interface Design

- Parameters are used to ***generalize*** the interface.
- Interfaces should be ***simple*** and ***standard.***
- A square() function that requires an angle parameter is not simple enough.
- A square() function that didn't let the user control the length is not standard.
- ***Refactoring*** – rewriting functions – is a standard part of designing and developing code.

circle() : Refactored

- A circle() is trivial to define as a 360 degree arc.

```
def circle(t, r):
    arc(t,r,360)
```


An Interface

- Interface is like a contract.
- For example, circle() requires 2 arguments, a Turtle and a number representing the radius.
- These requirements are ***preconditions.***
- If the user meets these preconditions, the function promises to meet certain ***postconditions*** – e.g., drawing a circle with the given radius.