LECTURE 4: CASE STUDY - INTERFACE DESIGN

Hung-Yu Wei

turtle module

- Create image using turtle graphics
  - *import turtle*
- Create a turtle object
  - *t= turtle.Turtle()*
- Move forward
  - *t.fd(distance)*
- Turn left/right
  - *t.lt(degree)*
  - *t.rt(degree)*

Repetition: *for* loop

- Control your program flow
  - ***Repetition***
  - *Selection (will teach this in the future)*
- Repetition with loop
  - *Avoid copy the same codes for several times*
  - *for loop*
- Syntax: Repeat n times with for loop

  for i in range*(n)*

  … actions to repeat

- Syntax: range(x)
  - *Create x integers from **0** to **x-1***

```
1  for i in range(4):
2      print('Hello!')
```

```
Hello!
Hello!
Hello!
Hello!
```

Drawing with turtle

```
1  import turtle
2  bob=turtle.Turtle()
3  print(bob)
4
5  bob.fd(100)
6  bob.lt(90)
7  bob.fd(100)
8
9  turtle.mainloop()
```

With Loop

```
1  import turtle
2  bob=turtle.Turtle()
3  print(bob)
4
5  for i in range(4):
6      bob.fd(100)
7      bob.lt(90)
8
9  turtle.mainloop()
```

Encapsulation

- Encapsulate with a function
- Example
  - *Draw a square*

```
def square(t):
    for i in range(4):
        t.fd(100)
        t.lt(90)

square(bob)
```

Keyword arguments for a function

- You might forget the meaning/order of arguments
- Tip: Include the names of parameters in argument list

Keyword arguments

```
polygon(bob, n=7, length=70)
```

[Concept] What's interface?

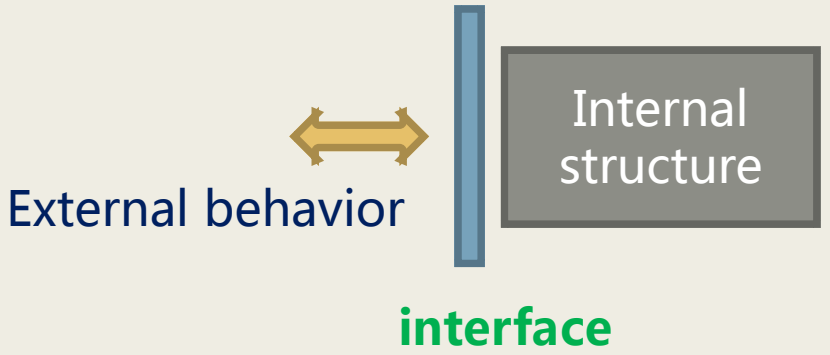
- **Interface** of a function
  - *Define how it is used? What does the function do?*
  - *What are the parameters?*
    - Input
  - *What is the return value?*
    - Output
- Documentation is important

```
def circle(t, r):
    circumference = 2 * math.pi * r
    n = 50
    length = circumference / n
    polygon(t, n, length)
```

[Concept]Refactoring

- **Refactoring** to improve the design of existing code
  - *improves its internal structure*
  - *does **NOT** alter the external behavior*

- Interface design



Overview of turtle examples in this chapter

- **Square** → **Polygon** (generalization) Slide 10
- **Polygon** → **Circle** (circle is approximated by polygon) Slide 11
- **Arc** Slide 12
  - *Observation: some codes in Arc are similar to **Polygon***
  - *Refactoring (I)*
    - Take the similar codes → create a new function **polyline**
    - Use **polyline** in the updated version of **Arc** and **Polygon** Slide 13
  - *Refactoring (II)*
    - Use **Arc** in the updated version of **Circle** Slide 14

Example: Square → Polygon

- Generalize for different side ***length***

```
def square(t, length):
    for i in range(4):
        t.fd(length)
        t.lt(90)

square(bob, 100)
```

```
def polygon(t, n, length):
    angle = 360 / n
    for i in range(n):
        t.fd(length)
        t.lt(angle)

polygon(bob, 7, 70)
```

- Generalize from square to ***polygon***

Example: Polygon → Circle

- Generalization

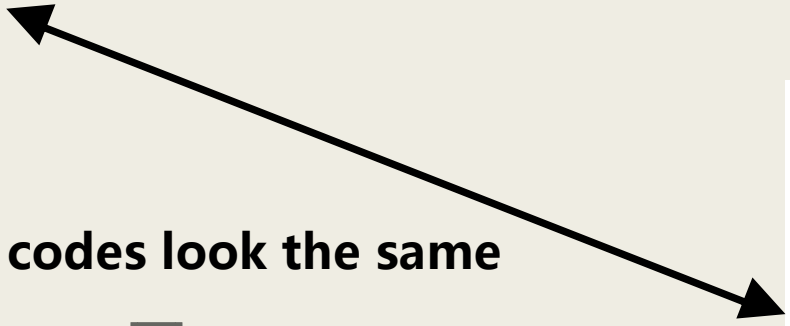
```
def circle(t, r):
    circumference = 2 * math.pi * r
    n = 50
    length = circumference / n
    polygon(t, n, length)
```

```
def circle(t, r):
    circumference = 2 * math.pi * r
    n = int(circumference / 3) + 3
    length = circumference / n
    polygon(t, n, length)
```

Example: arc

```
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 = angle / n

    for i in range(n):
        t.fd(step_length)
        t.lt(step_angle)
```



```
def polygon(t, n, length):
    angle = 360 / n
    for i in range(n):
        t.fd(length)
        t.lt(angle)
```

These codes look the same

Let's refactor the codes

Example: arc -- refactoring polyline

- **Refactoring**: create a new function for similar (same) codes

```
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 = angle / n

    for i in range(n):
        t.fd(step_length)
        t.lt(step_angle)
```

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

```
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)
```

Example: circle (using arc)

- **Refactoring**: circle

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

- Notice: they have the same interface

Old codes

```
def circle(t, r):
    circumference = 2 * math.pi * r
    n = int(circumference / 3) + 3
    length = circumference / n
    polygon(t, n, length)
```

Reading

- Chapter 4 in textbook “Think Python”