Functions

We have been using functions in our math

- cosine, log, etc

Functions

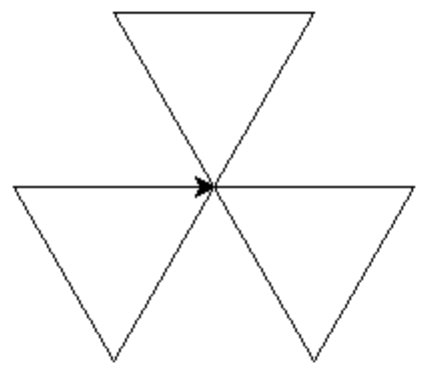
Abstraction

Turtle Graphic

- How to draw this?
- Drawing one triangle is

```
for i in range(3):
    forward(100)
    right(120)
```

- We can
  - Improvement?
  - First we can use a for-loop to repeat
  - Also, what is the meaning of these now?
    - It means draw triangle



```
for i in range(3):
    forward(100)
    right(120)
rt (120)
for i in range(3):
    forward(100)
    right(120)
rt (120)
for i in range(3):
    forward(100)
    right(120)
rt (120)
```

Turtle Graphic

- Is it more meaning if I do

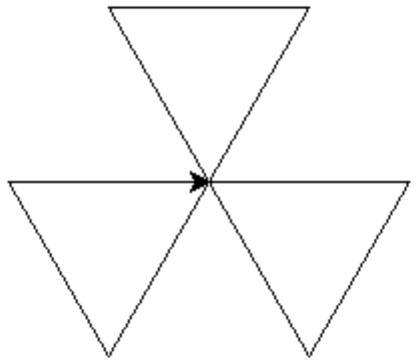
```
def draw_triangle():
    for i in range(3):
        forward(100)
        right(120)
```

```
draw_triangle()
rt(120)
draw_triangle()
rt(120)
draw_triangle()
rt(120)
```

- or even just

```
for i in range(0,3):
    draw_triangle()
    rt(120)
```

- Which version is the best and why?



```
for i in range(3):
    forward(100)
    right(120)
rt (120)
for i in range(3):
    forward(100)
    right(120)
rt (120)
for i in range(3):
    forward(100)
    right(120)
rt (120)
```

Function Abstraction

- What does this do?

```
def draw_triangle():
    for i in range(3):
        forward(100)
        right(120)
```

- It simply means,
  - Now I define a function called “draw_triangle()”
  - Whenever I call the function, I will execute its body

Function Abstraction

Abstraction in Daily Life: Cooking Rice

Abstraction in Daily Life: Cooking Rice

- BBC

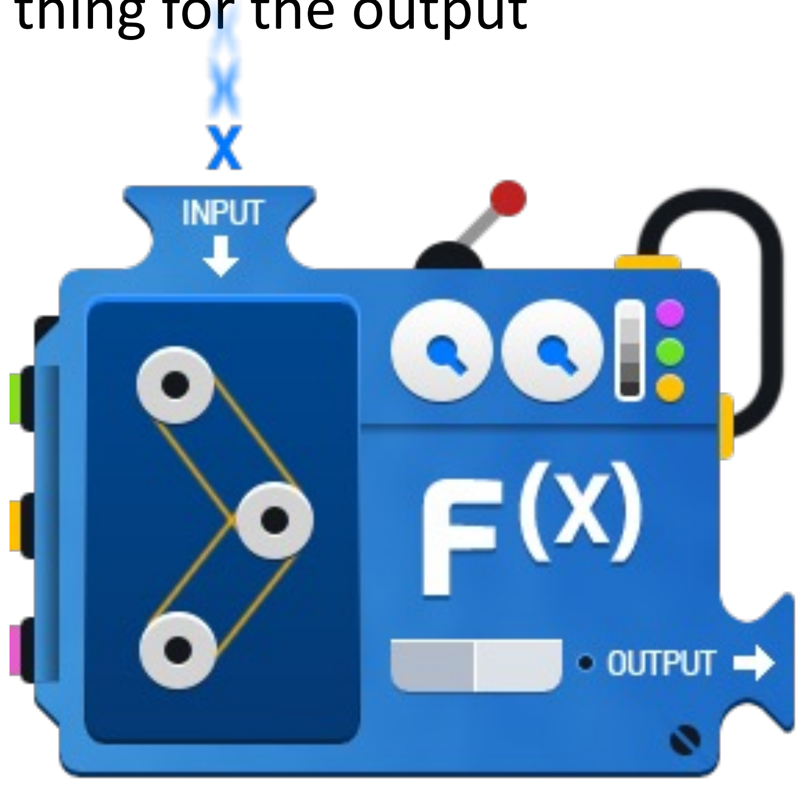
1. Measure the rice into a cup and level the top
2. Rinse the rice thoroughly in cold water
3. Pour the rice into a pan over a low heat
4. Add double the amount of water
5. Bring to a boil
6. Put a lid on and turn the heat down to as low as possible
7. Cook for 10 mins and do not take the lid off
8. Fluff the rice with a fork

- Abstraction:

https://www.bbcgoodfood.com/videos/techniques/how-cook-rice

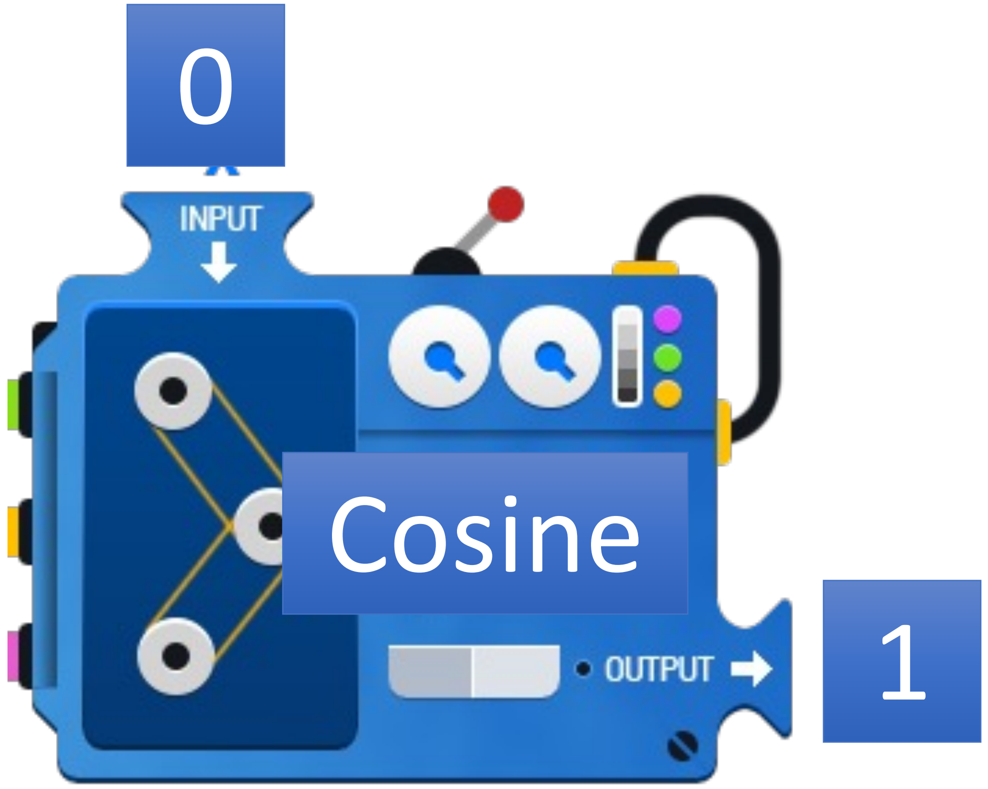
Functions

- A function is like a black box
  - You put in something for the input
  - And it will produce a new thing for the output



For example

- “Cosine” is a function
  - Input 0
  - Output 1

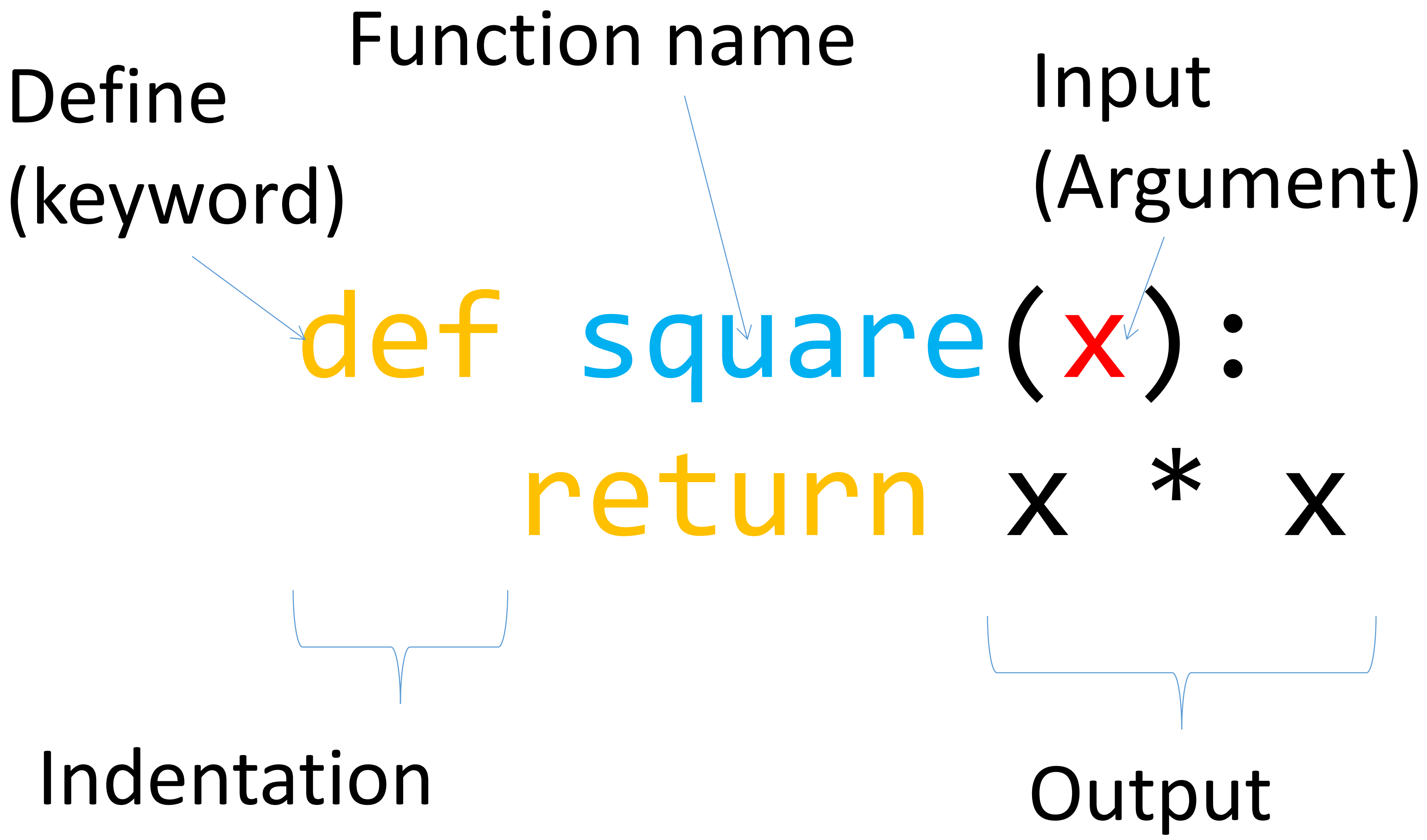


Let’s Write Our Own Function!

```
def square(x):
    return x * x
```

```
>>> square(3)
9
>>> square(square(2))
16
>>> square(3,4)
???
```

Let’s Write Our Own Function!



```
def square(x):
    return x * x
```

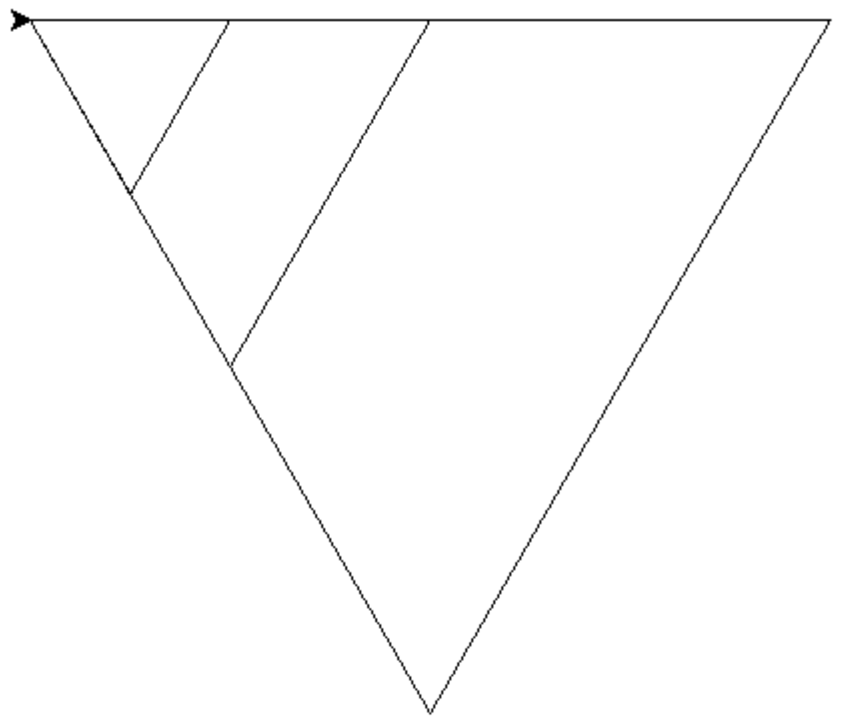
What if I want to draw a triangle of different length?

- Again, not elegant!
- The only difference is the length of the triangle
- We can make it into a parameter

```
def draw_triangle100():
    for i in range(3):
        forward(100)
        right(120)

def draw_triangle200():
    for i in range(3):
        forward(100)
        right(120)

def draw_triangle300():
    for i in range(3):
        forward(300)
        right(120)
```

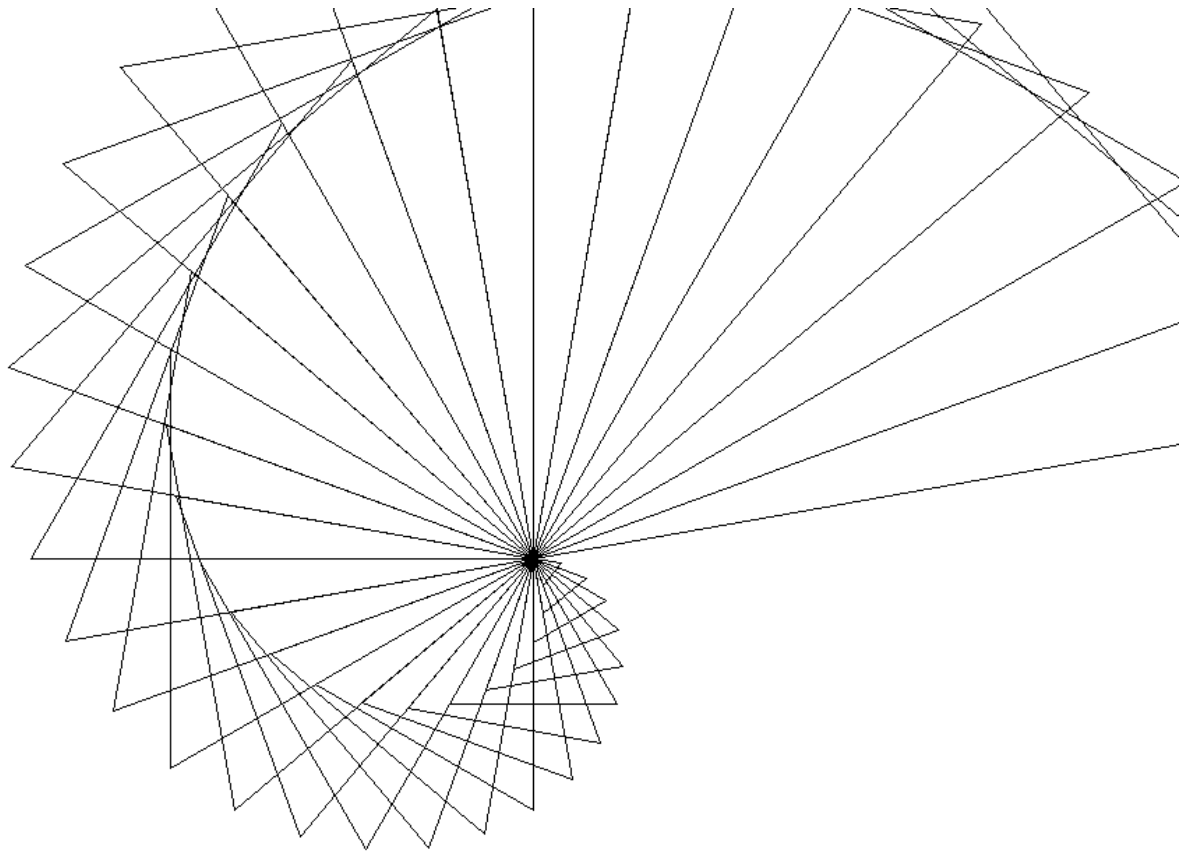
Function Parameters

```
def draw_triangle(side_length):
    for i in range(3):
        forward(side_length)
        right(120)

draw_triangle(100)
draw_triangle(200)
draw_triangle(400)
```



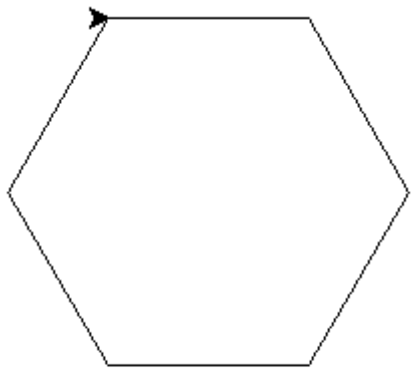
Play Around

```
def draw_triangle(side_length):
    for i in range(3):
        forward(side_length)
        right(120)

for i in range(30):
    draw_triangle(i*20)
    rt(10)
```

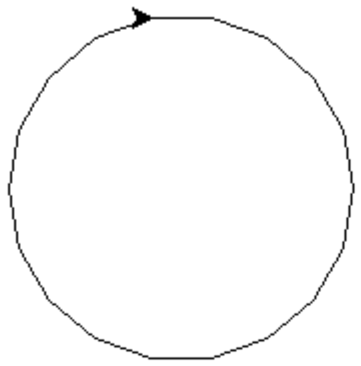


Exercise

- Write a function `square(side_length)` that can draw a square of any size
- Write a function `polygon(n,side_length)` that can draw a regular polygon with n sides
  - E.g. triangle (3), square(3), hexagon (6)



`polygon(6,100)`

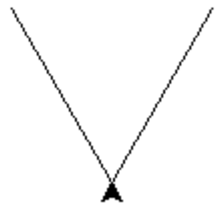


`polygon(18,30)`

Recursion

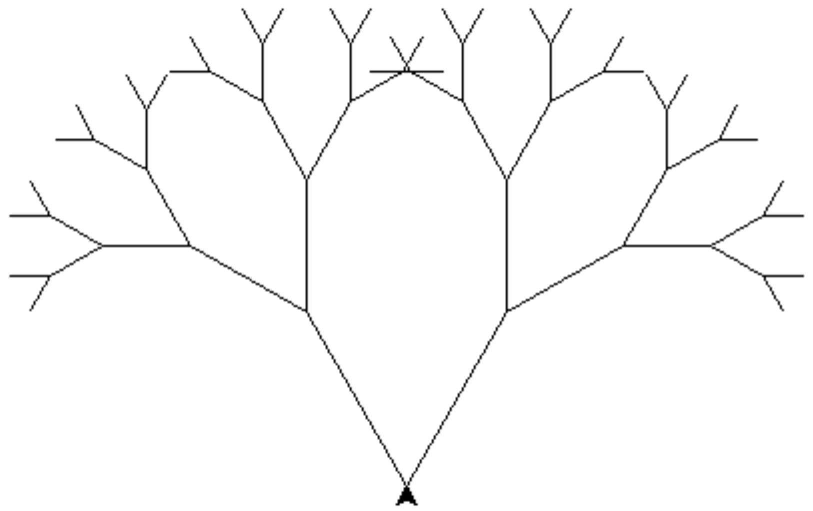
Let’s draw a V

```
def draw_v(l):
    lt(30)
    forward(l)
    backward(l)
    rt(60)
    forward(l)
    backward(l)
    lt(30)

lt(90)
draw_v(100)
```



```
def draw_v(l,n):
    if n == 0:
        return
    lt(30)
    forward(l)
    draw_v(l/1.5,n-1)
    backward(l)
    rt(60)
    forward(l)
    draw_v(l/1.5,n-1)
    backward(l)
    lt(30)

draw_v(100,5)
```

```
def draw_v(l,n):
    if n == 0:
        return
    lt(30)
    forward(l)
    draw_v(l/1.5,n-1)
    backward(l)
    rt(60)
    forward(l)
    draw_v(l/1.5,n-1)
    backward(l)
    lt(30)

draw_v(100,5)
```



Exercises

- Try to draw the following two