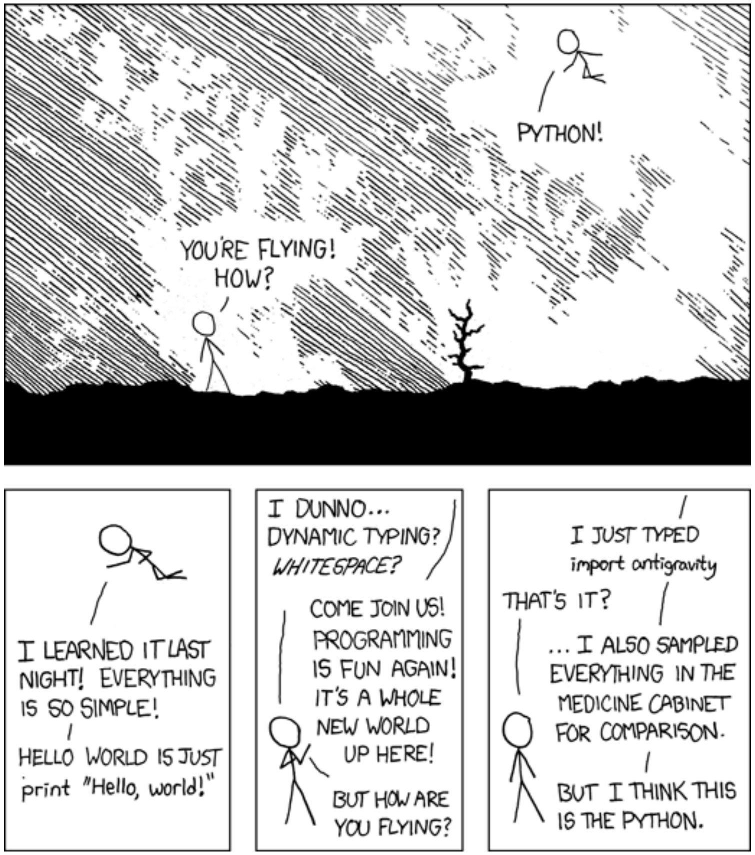
WTF is **programming**?

A (brief) introduction to python.

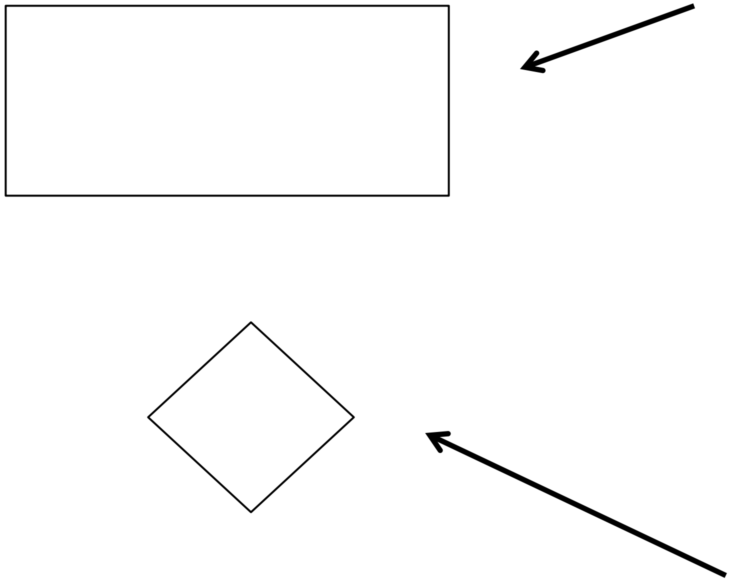
Robert with a computer.
See? I'm totally legit.


YOU'RE FLYING! HOW?
PYTHON!
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

With your host: Robert Williamson
a.k.a. that guy with the wings at Halloween

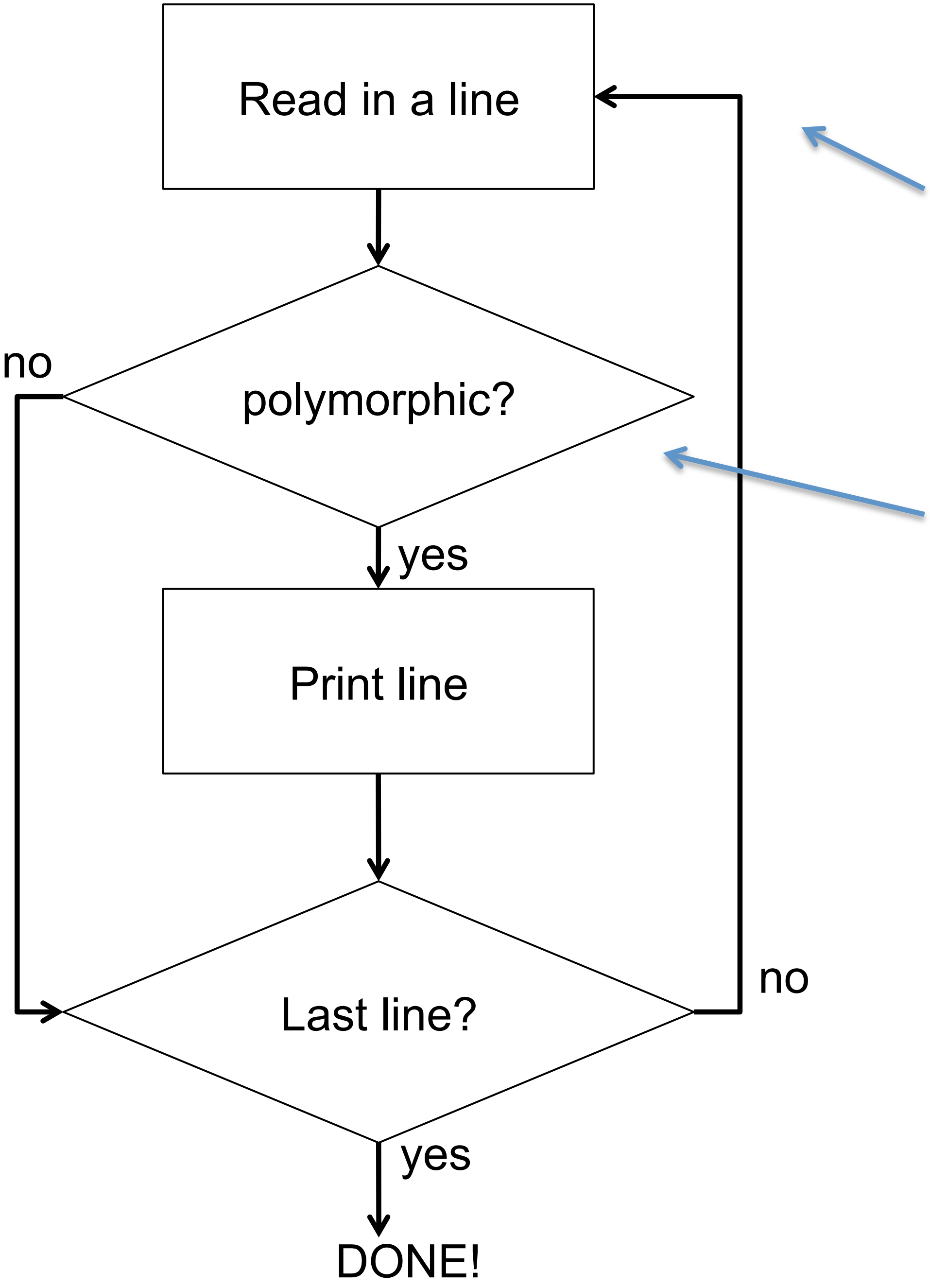
(scary!)

Thinking like a programmer: flow charts



Tasks, anything that your program needs to do

Decisions, anywhere your program needs to check on something



Read in a line

polymorphic?

no

yes

Print line

Last line?

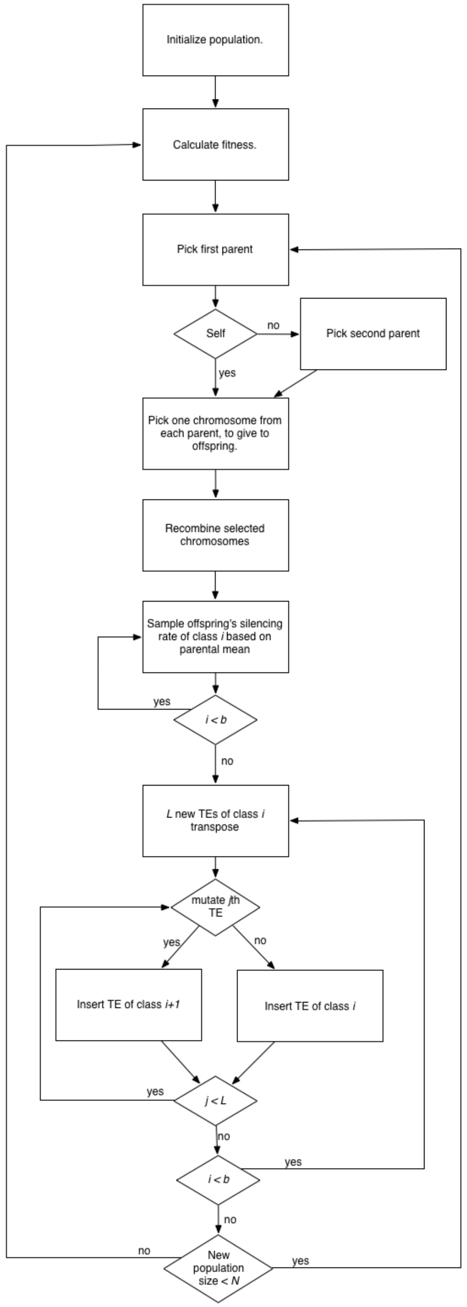
no

yes

DONE!

This means that we need a loop

These are if statements


Initialize population.
Calculate fitness.
Pick first parent
Self
no
Pick second parent
yes
Pick one chromosome from each parent, to give to offspring.
Recombine selected chromosomes
Sample offspring's silencing rate of class i based on parental mean
yes
i < b
no
L new TEs of class i transpose
mutate jth TE
yes
no
Insert TE of class i+1
Insert TE of class i
yes
j < L
no
yes
i < b
no
no
New population size < N
yes

Comparison Operators

Operator	**Effect**
A == B	returns True if A is equal to B
A != B	returns True if A is not equal to B
A < B	returns True if A is less than B
A > B	returns True is A is greater than B
A <= B	returns True if A is less than or equal to B
A >= B	returns True if A is greater than or equal to B
A or B	returns True if either A or B is True
A and B	returns True if both A and B are True
not A	returns True if A is False

Remember, the number 0, the empty list [], and the None value all evaluate as False. So if you try 0 and A, since 0 is equal to False this is equivalent to saying False and A.

Anything in blue, is Python syntax, and is required.

Anything in green, can be changed. We decide what to put there.

Anything in *itallics*, is a variable name. We could change this to any name.

Anything in **bold**, is a function that we defined. We could change these names too.

How to define a function:

```
def functionName(arg1, arg2, … argN):
    my code
```

How to define a function:

```
def functionName(arg1, arg2, … argN):
    my code
```

Any word with a parenthesis after it is a function

How to define a function:

```
def functionName(arg1, arg2, … argN):
    my code
```

Any word with a parenthesis after it is a function

```
def funnyPrint():
    print(“Look, a joke.”)
```

How to define a function:

```
def functionName(arg1, arg2, … argN):
    my code
```

Any word with a parenthesis after it is a function

```
def funnyPrint():
    print(“Look, a joke.”)
```

```
def multiply(x, y):
    z = x*y
    print(z)
```

How to return values from a function:

```
def functionName(arg1, arg2, … argN):
    my code
    return(result)
```

```
def getJoke():
    joke = “Look, a joke.”
    return(joke)
```

```
def multiply(x, y):
    z = x*y
    return(z)
```

```
def multiplyAndDivide(x, y):
    z = x*y
    w = x/y
    return(z, w)
```

Basic if statements:

```
if test:
    my code
```

```
if generation > 100:
    print(“Done”)
```

```
If genotype == homozygous:
    print(“homozygote found”)
```

Adding else statements:

```
if test:
    my code
else:
    my alternate code
```

```
if generation > 100:
    print(“Done”)
else:
    print(“continue to generation: “+str(generation))
```

for loops:

```
for x in list:
    my code
```

```
for number in myList:
    print(number)
```

```
for individual in population:
    calcFitness(individual)
```

Chained if-else statements:

```
if test:
    my code
elif other test:
    my other code
else:
    my alternate code
```

```
If genotype == homozygous:
    print(“homozygote found”)
elif genotype == heterozygous:
    print(“heterozygote found”)
else:
    print(“weird data found”)
```

while loops:

```
while test:
    my code
```

```
while generation < 100:
    print(generation)
    makeNextGeneration(generation)
    generation = generation + 1
```

How to open files and read them:

```
myFile = open(filename, “r”)
myFile.readline()
```

```
myFile = open(“meat.txt”, “r”)
myLine = myFile.readline()
print(myLine)
myLine = myFile.readline()
print(myLine)
```

```
myFile = open(“meat.txt”, “r”)
for line in myFile:
    print(line)
```

How to open files and write them:

```
myFile = open(filename, “w”)
myFile.write(text)
```

```
myFile = open(“output.txt”, “w”)
myFile.write( “Hello File!\n”)
```