
Python DeCal
Week 3

Announcements

- 1st Hw was due just now!
  - Wanna discuss it a bit? (poll)
- Office Hour Poll (only four of y'all have filled it in :c)
  - https://www.when2meet.com/?9754936-Zp46U
- Attendance!
  - https://forms.gle/mMFvrrxjJLtB8cGKA

Recap

- Data Type (int, float, string, lists, dictionary, tuples, booleans)
- Functions
  - Construct a function
  - Calling a function
- Variable scope

However...

- Our code can’t make any “judgements”
- It can’t change the flow of the code according to the input/variable

Boolean Operations

operators	descriptions
(), [], {}, ‘’	tuple, list, dictionnary, string
x.attr, x[], x[i:j], f()	attribute, index, slide, function call
+x, -x, ~x	unary negation, bitwise invert
**	exponent
*, /, %	multiplication, division, modulo
+, -	addition, substraction
<<, >>	bitwise shifts
&	bitwise and
^	bitwise xor
	bitwise or
<, <=, >=, >	comparison operators
==, !=, is, is not, in,	comparison operators (continue)
not in	comparison operators (continue)
not	boolean NOT
and	boolean AND
or	boolean OR

Boolean Operations

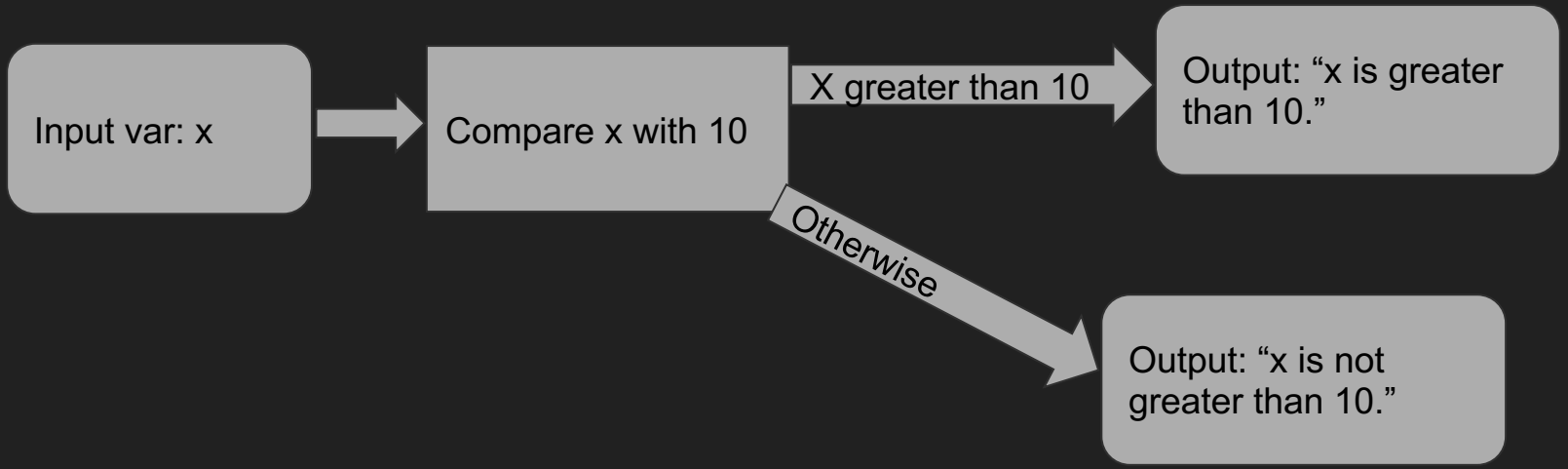
- Ture or false

1. `True and True`
2. `1==2`
3. `2!=2 & 1==1`
4. `(2==2)^(1==1)`
5. `(2!=2)^(1==1)`
6. `2!=2^1==1`
7. `(2!=2)or(1==1)`
8. `2!=2 or 1==1`

operators	descriptions
(), [], {}, ‘’	tuple, list, dictionnary, string
x.attr, x[], x[i:j], f()	attribute, index, slide, function call
+x, -x, ~x	unary negation, bitwise invert
**	exponent
*, /, %	multiplication, division, modulo
+, -	addition, substraction
<<, >>	bitwise shifts
&	bitwise and
^	bitwise xor
	bitwise or
<, <=, >=, >	comparison operators
==, !=, is, is not, in,	comparison operators (continue)
not in	comparison operators (continue)
not	boolean NOT
and	boolean AND
or	boolean OR

If Statements

- Write a piece of code that outputs if the variable is greater than 10:



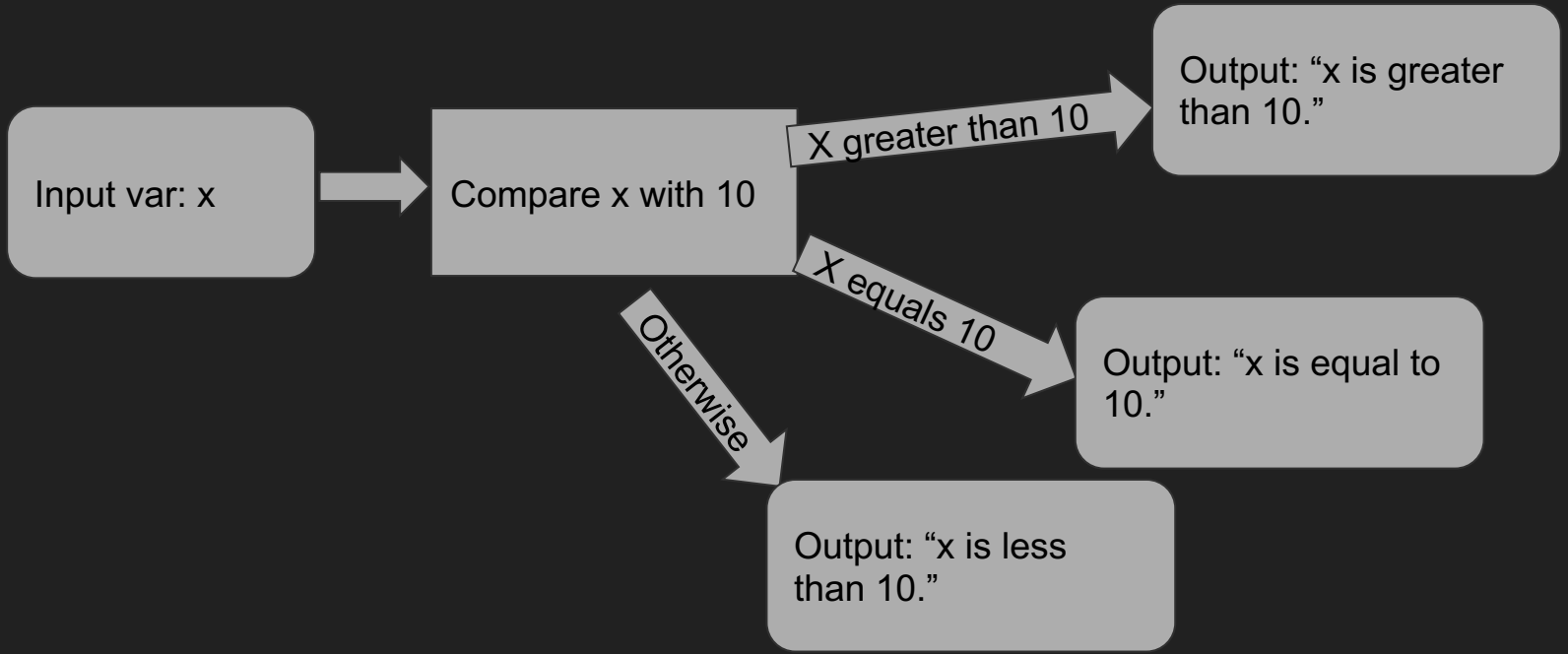
If Statements

- Write a piece of code that outputs if 10:

```
x = int(input(“please enter an ir
if x>10:
    print(“x is greater than 10.”
else:
    print(“x is not greater than
```

If Statements

- What if we want to separate also the equal case?



If Statements

- Write a piece of code that outputs if the variable is greater than 10:

```
x = int(input(“please enter an integer:”))
if x>10:
    print(“x is greater than 10.”)
elif x==10:
        print(“x is equal to 10”)
else:
        print(“x is smaller than 10.”)
```

If Statements

- We can also write nested if statements:
- We want to find if x can be divided by 2 and 3:

```
x = int(input(“please enter an integer:”))
if x%2 == 0:
    if x%3 ==0:
      print(“2 and 3 both divides x.”)
else:
       print(“2 and 3 can’t divide x simultaneously.”)
```

- Can you write it without the nested statements?

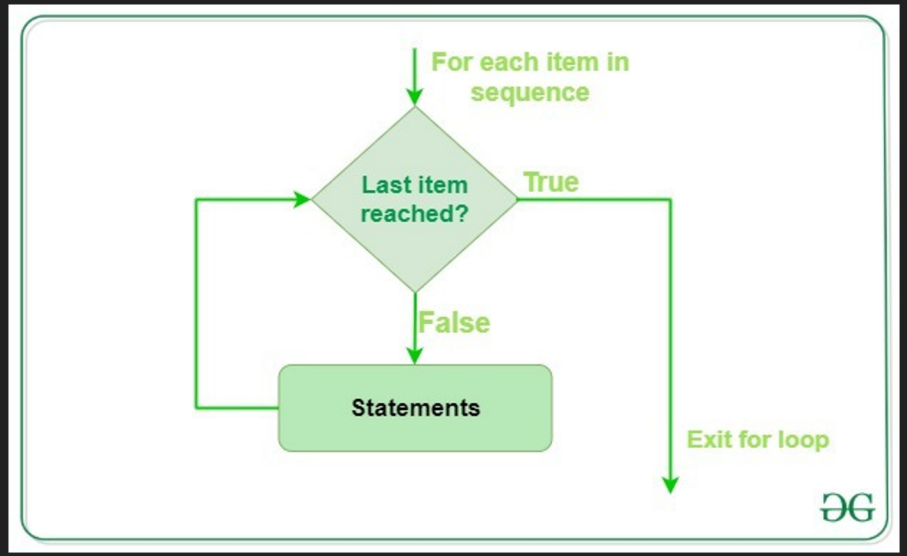
If Statements

- We can also write nested if statements:
- We want to find if x can be divided by 2 and 3:
- Challenge: now we want to include all the consequences--
  - Can be divided by 2 but not by 3
  - Can be divided by 3 but not by 2
  - Can’t be divided by 3 nor 2

Iterative tasks

- Now we want the code to run n times
- For example, you wanna spam your friends (don’t do that). You would want the code to run numerous times (like 100 times *precisely*).
- If statements does not do that :c

Loops-for loop



Loops-for loop

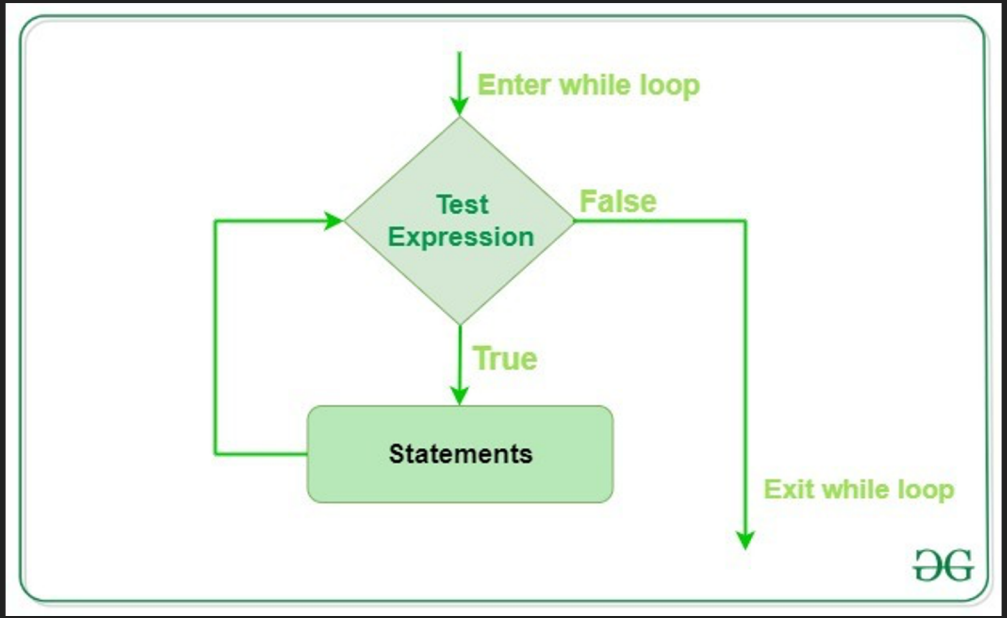
- For loop: iterate over items in a list
- Then we need either run commands on existing lists, or create new list for it to run over.
- For example, you want to print numbers from 0 to 100. It would be a long task for you if you wish to write them all out!

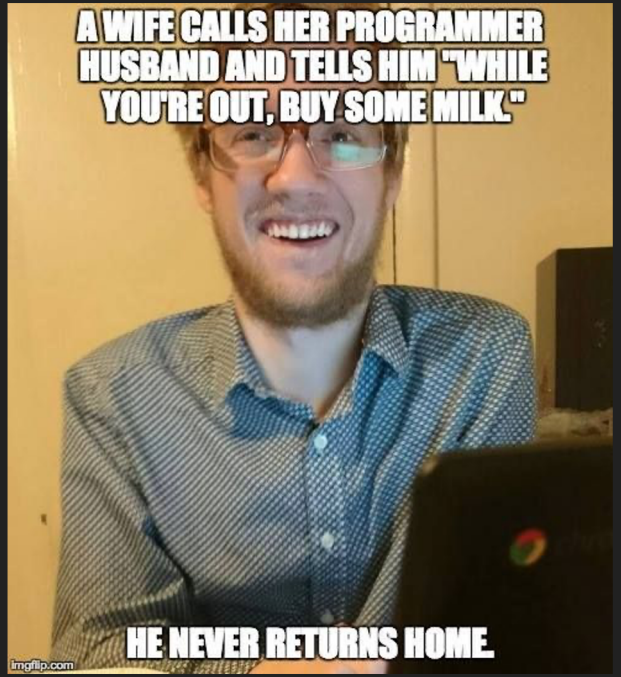
Loops-for loop

- Print numbers from 1 to 100.

```
for i in range(1,101):
    print(i)
```

Loops-while loop





Loops-while loop

- While loop: print numbers 1 to 5

```
i = 1
while i <6:
    print(i)
    i += 1
```

- What will be the output now?

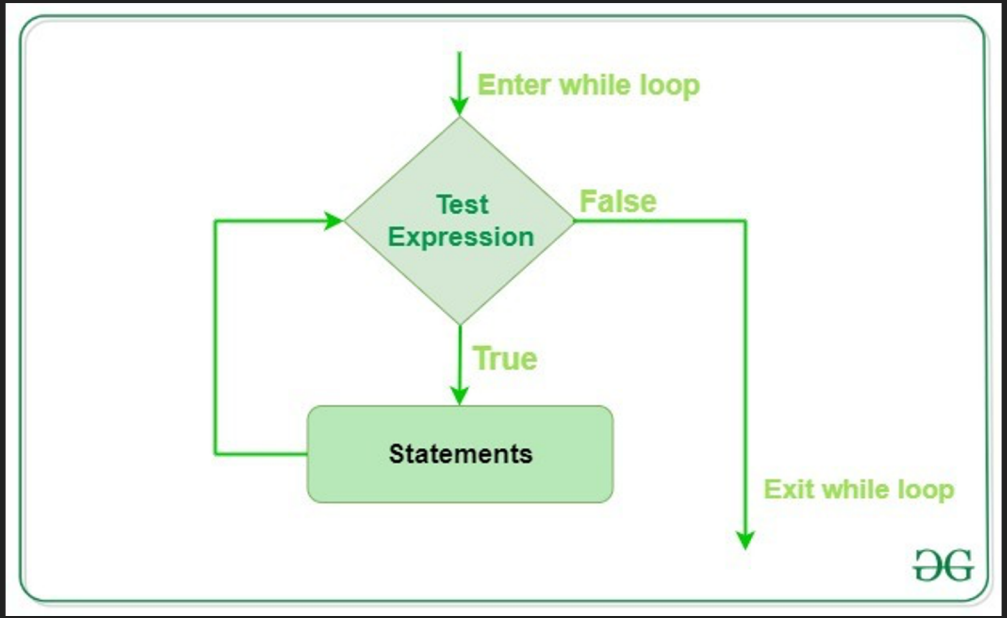
```
i = 1
while i <6:
    i += 1
    print(i)
```

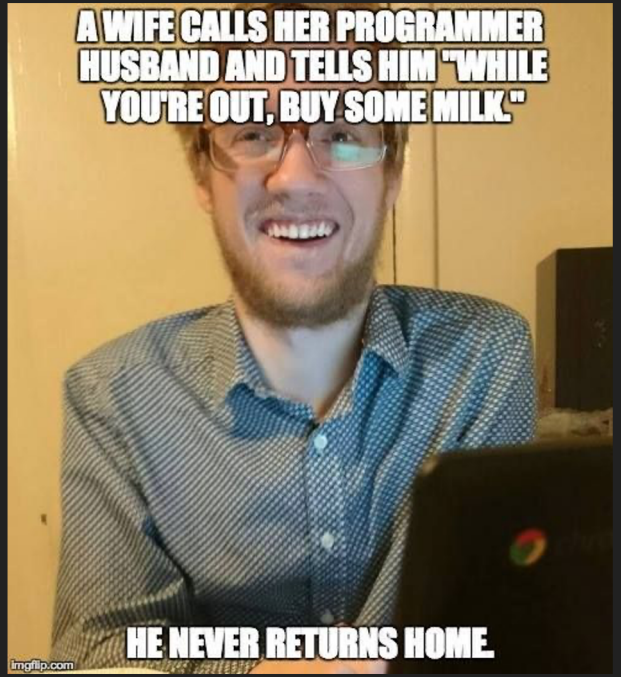
Loops-while loop

- What will be the output now?

```
i = 1
while i <6:
    print(i)
```

Loops-while loop





Loops-while loop

```
out = True
milk=0
while out = True:
    print(“Buy milk”)
    milk+=1
    if milk>0:
            break
print(“The programmer is back home”)
```

FRIDAY

Announcements

- Homework 2 is up! It will take a bit more time than last week so be prepared.
- Office Hours
- What are we doing today?

Attendance Form

- https://tinyurl.com/our-dearest-pluto

Break Out Rooms!

Question: What is the difference between a `while` and a `for` loop and why would you want to use each of them?

While

- Use when you want something to happen as long as condition is met

```
While (condition is True):
        Do some stuff
        Something will change each time
```

For

- Used when you want to iterate over something (a list, a tuple, etc...) but you don't necessarily need a condition to be met while it does this.

```
For (thing in a bunch of things):

        Do some stuff for each thing in a bunch of
things
```

If Dr. Seuss could code

Examples

`While`

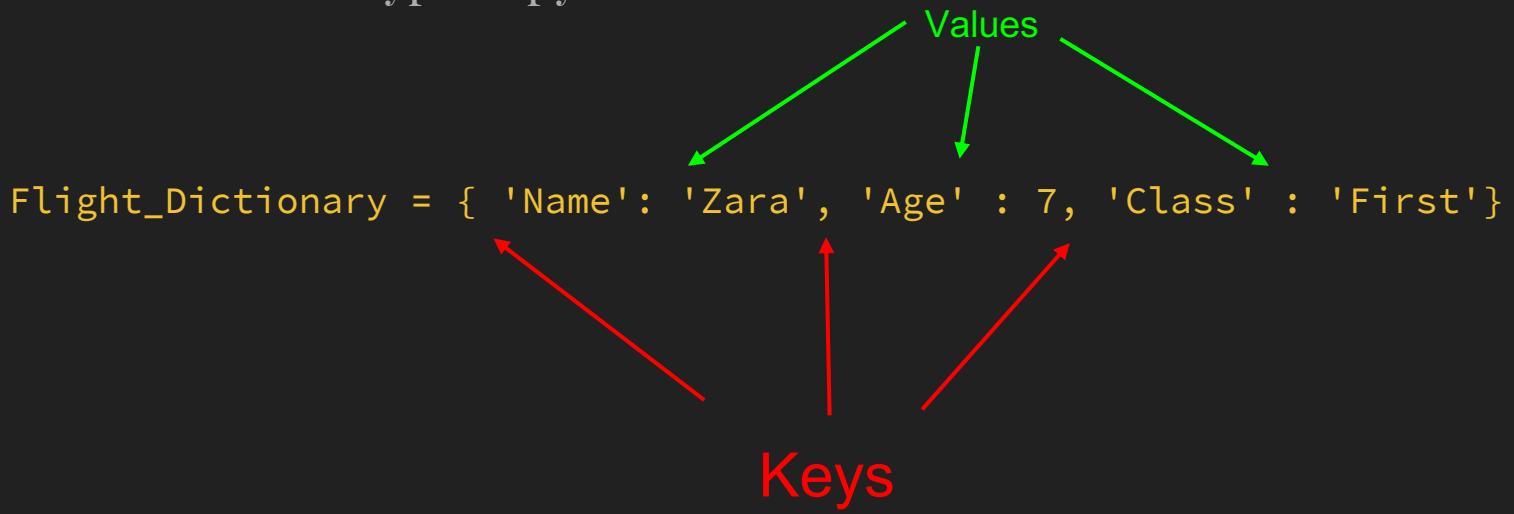
- Video Games
- Password Inquirer
- Simulations
- Steps don’t need to be Discrete integer values

`For`

- Data Sets (large lists or files)
- Grid Simulations
- Changing specific values in a list
- Discrete integer steps

Dictionaries

- The last built in data type of python!



Dictionaries

- Flight_Dictionary[key] = value
- Flight_Dictionary[‘Name’] = ‘Zara’
- Flight_Dictionary[‘Age’] = 7

```
>>> nums = {'I': 1.0, 'V': 5, 'X': 10}
>>> nums['X']
10
>>> nums['I'] = 1
>>> nums['L'] = 50
>>> nums
{'X': 10, 'L': 50, 'V': 5, 'I': 1}
>>> sum(nums.values())
66
>>> dict([(3, 9), (4, 16), (5, 25)])
{3: 9, 4: 16, 5: 25}
>>> nums.get('A', 0)
0
>>> nums.get('V', 0)
5
```

Dictionaries

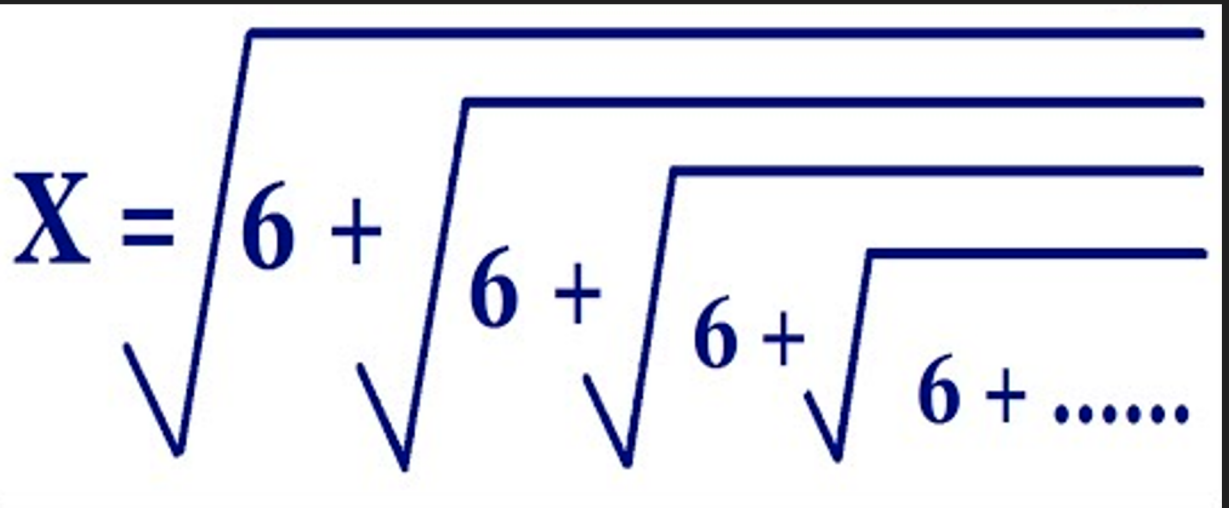
Example:

```
SN1987a = { ‘Apparent Magnitude’:2.9,
            ‘Distance’:51.4,
            ‘RA’:‘05h 35m 28.03s’,
            ‘Dec’:‘-69° 16' 11.79"’,
            ‘Type’:‘II’ }
```

Recursion

- Very complicated topic, should take CS61A if you are really curious about it
- Essence… call a function on itself repeatedly to make it do what you want

WE CAN CODE THIS….



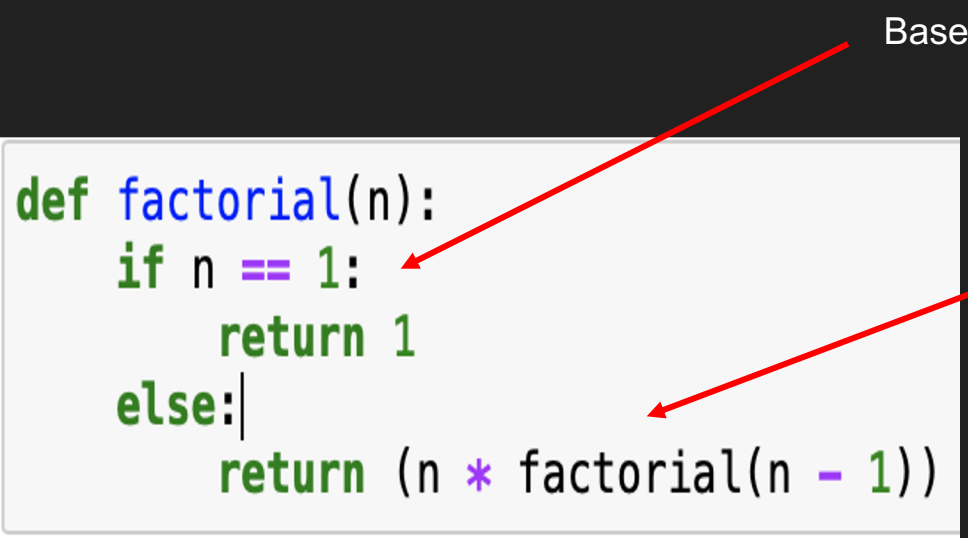
Recursion

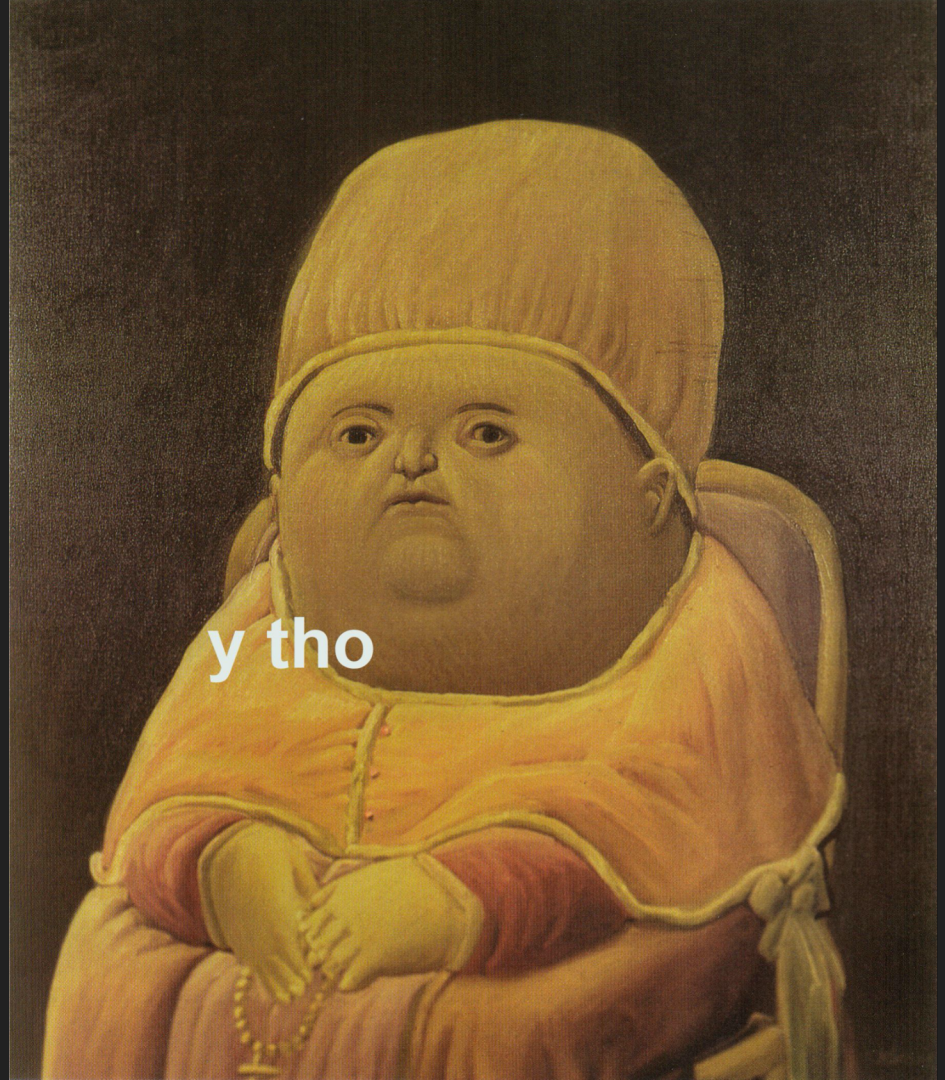
- You should always have two groups of cases. A base case and a recursive case

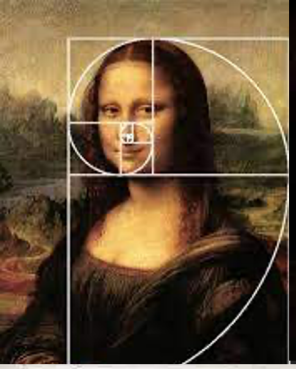
- Base case essentially is the end of the long chain, which allows the chain reaction to begin the other way to give you desired answer

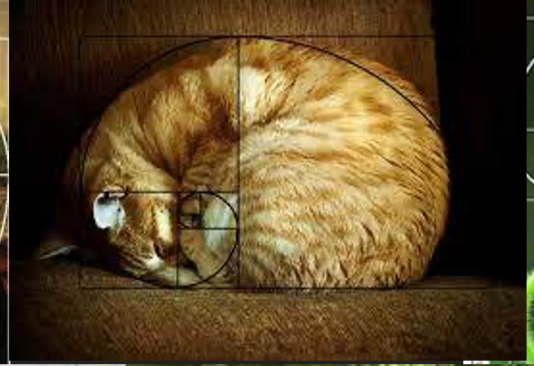
Recursion

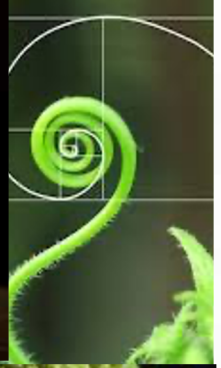
- How do we implement it ???
- Take a recursive leap of faith....

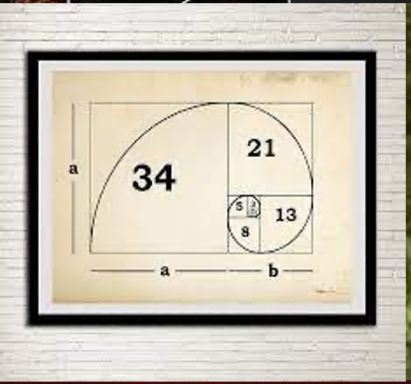


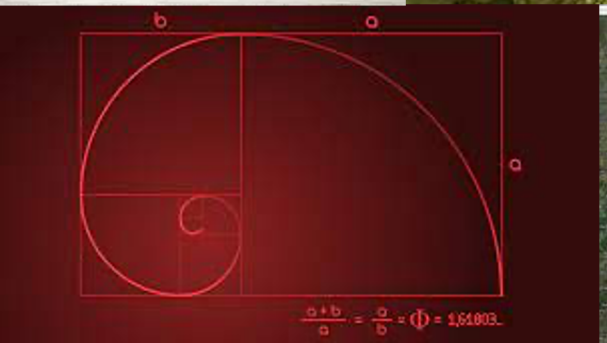

y tho

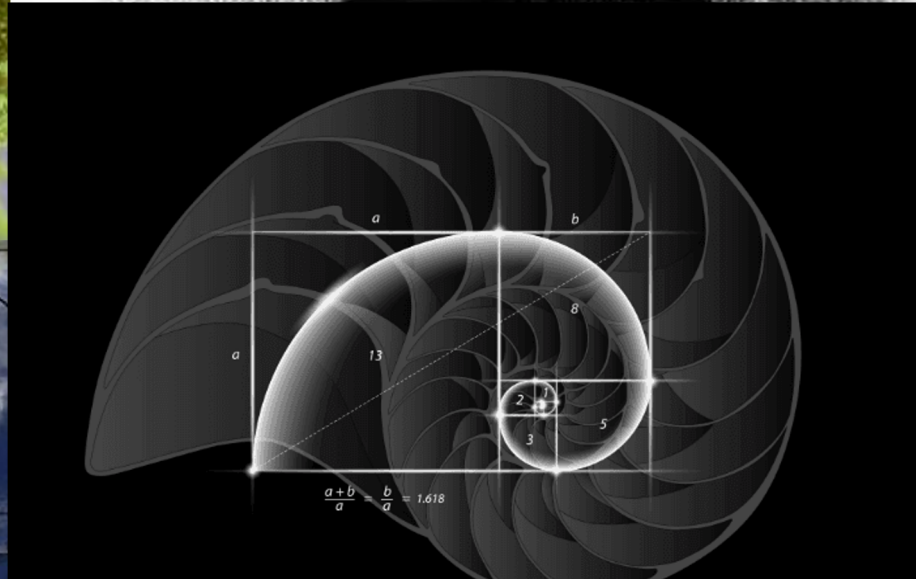







34
21
13
8
5
2
a
a
b


b
a
a
$\frac{a+b}{a} = \frac{a}{b} = \Phi = 1{,}61803\ldots$


a
b
a
13
8
2
1
3
5
$\frac{a+b}{a} = \frac{b}{a} = 1.618$

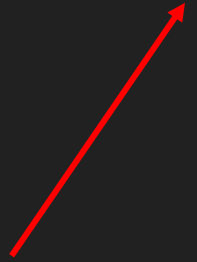
Example: Fibonacci

WHILE METHOD:

```
def while_fib(n):
        pred = 0
        curr = 1
        k = 1
        while k < n:
            pred = curr
            curr = pred + curr
            k = k + 1
        return curr
```

RECURSIVE METHOD:

```
def fib(n):
        if n <= 1:
            return n
        else:
            return (fib(n - 1) +
fib(n - 2))
```



IT'S PRETTY....

Demos

Questions