

Picture Language Part 2

...

So far...



You

Follow what I
say and do it
for me!

Smaller,
Simpler,
Steps



Cool and complex stuff

In the lecture earlier...

**Repeating
the same
steps**

**Smaller,
Simpler,
Steps**

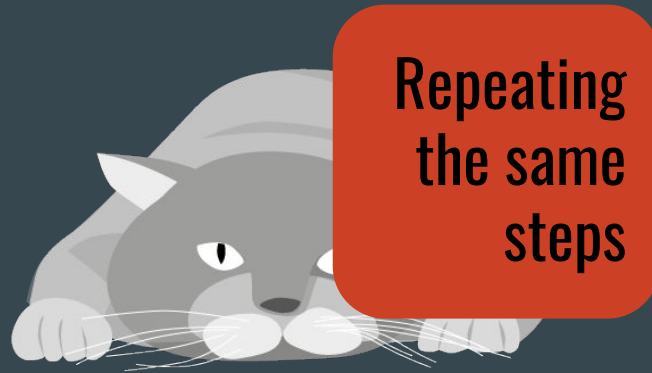
You

Follow what I
say and do it
for me!



Cool and complex stuff

In the lecture earlier...



Repeating
the same
steps

Smaller,
Simpler,
Steps

You

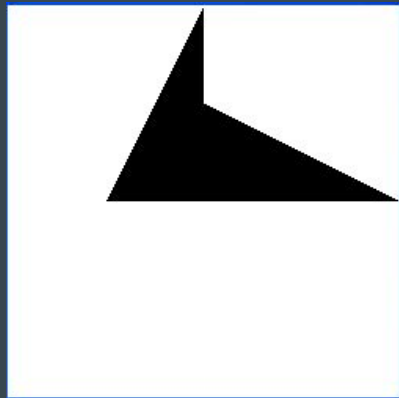
Loops



Cool and complex stuff

In this video...

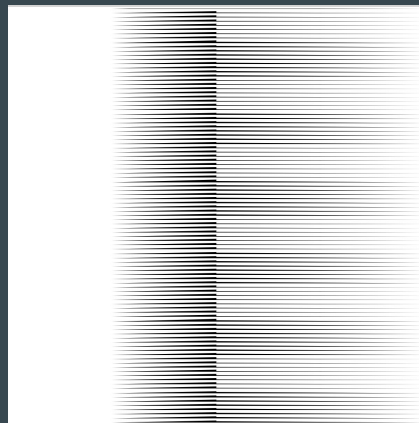
How do we use **loops** to repeat the same
runes to create complex patterns?



nova_bb



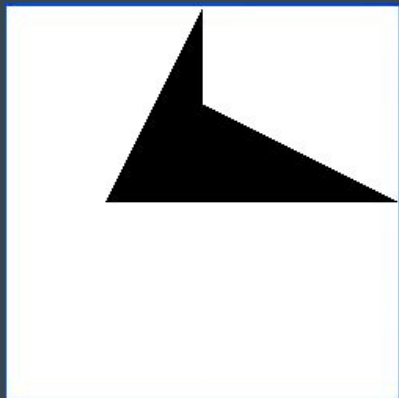
Repeat using
loops



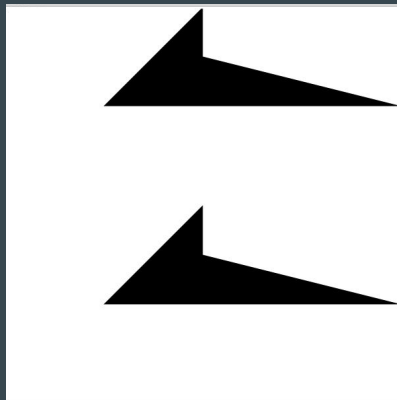
99 rows of nova_bb

Let's start with this first...

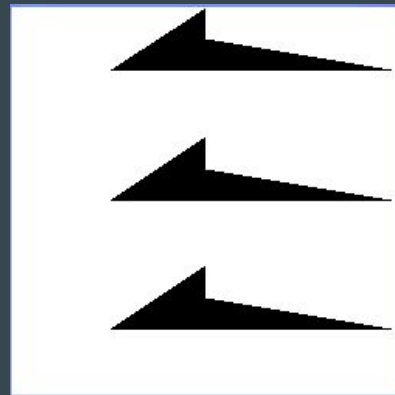
1 row



2 rows



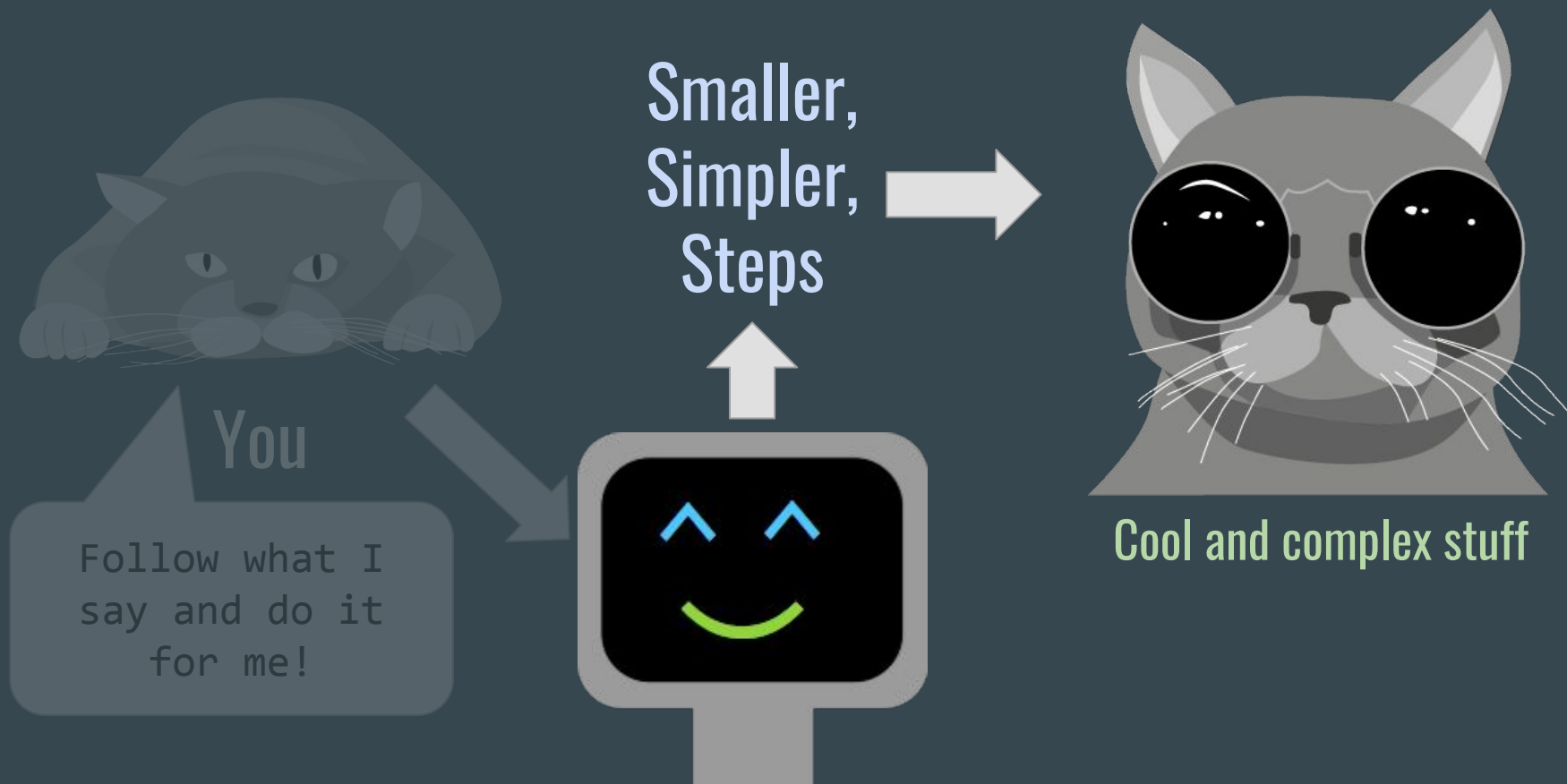
3 rows



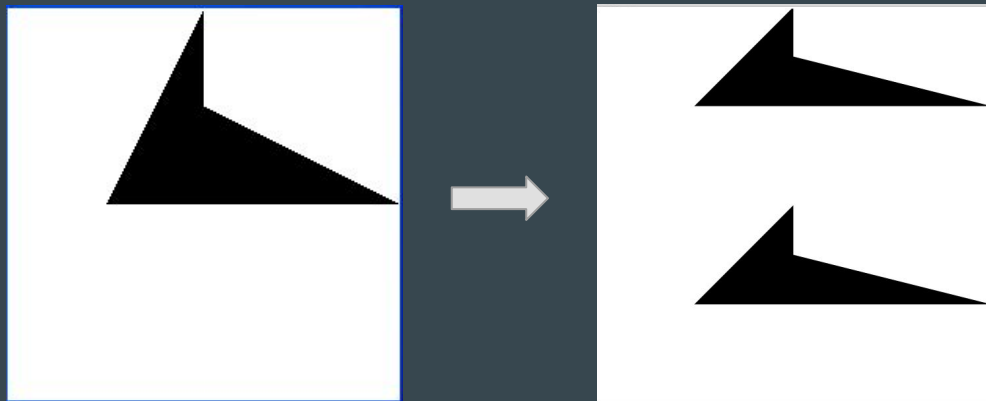
n rows?



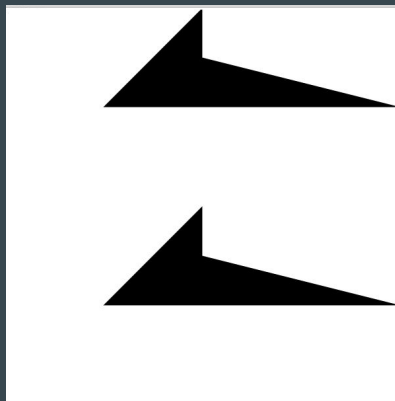
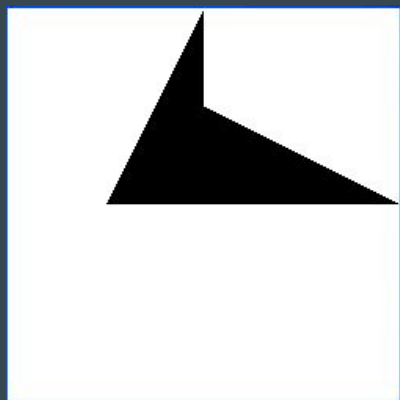
Let's use an example to make the complex simpler



Example: How do we create 2 equal rows with nova_bb?

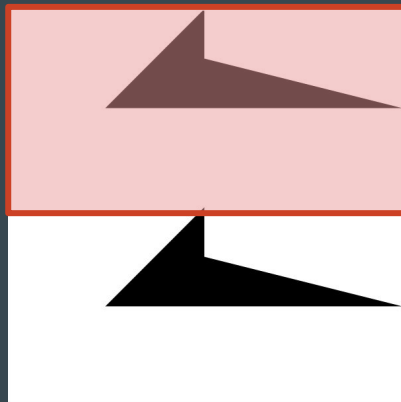
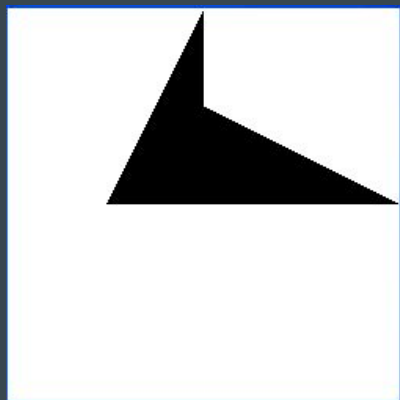


Example: How do we create 2 equal rows with nova_bb?



Use two identical
nova_bb

Example: How do we create 2 equal rows with nova_bb?



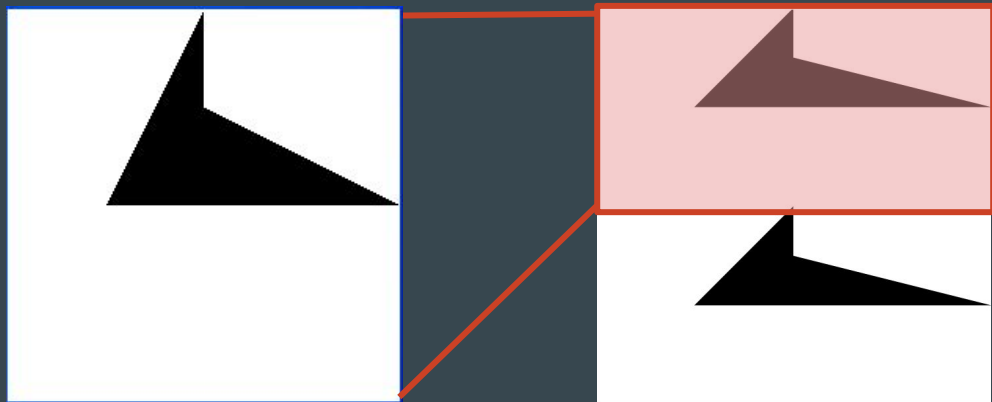
First nova_bb takes up
top half the space

Use two identical
nova_bb

2nd nova_bb takes up
the remaining bottom

We already know this!

```
stack_frac(1/2, nova_bb, nova_bb)
```



First `nova_bb` takes up
top half the space

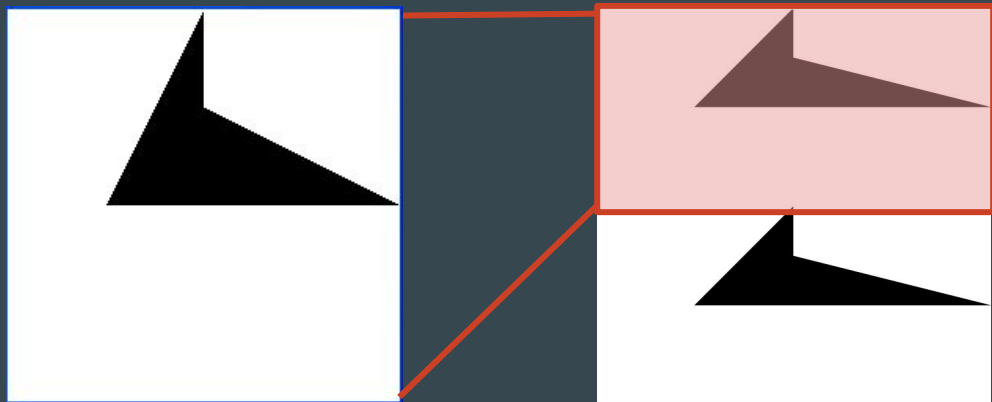
Use two identical
`nova_bb`

2nd `nova_bb` takes up
the remaining bottom

We already know this!

```
stack_frac(1/2, nova_bb, nova_bb)
```

Stack blocks



First `nova_bb` takes up
top half the space

Use two identical
`nova_bb`

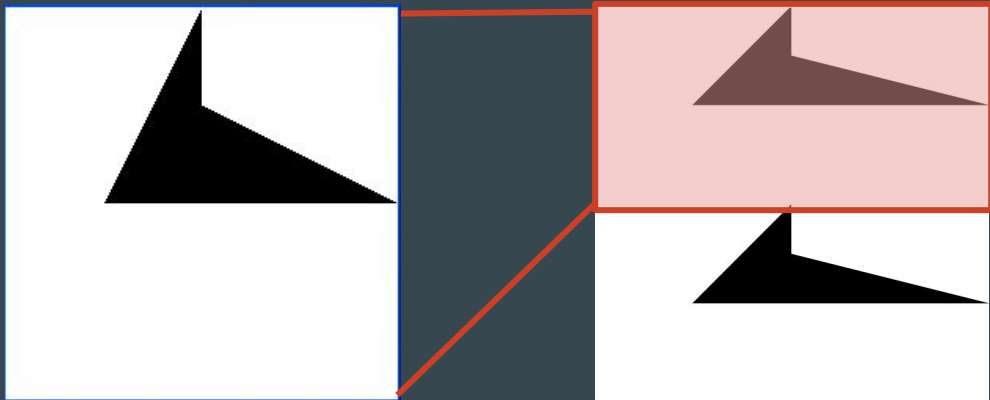
2nd `nova_bb` takes up
the remaining bottom

We already know this!

```
stack_frac(1/2, nova_bb, nova_bb)
```

Stack blocks

Top block
takes up $\frac{1}{2}$ of
the space



First `nova_bb` takes up
top half the space

Use two identical
`nova_bb`

2nd `nova_bb` takes up
the remaining bottom

We already know this!

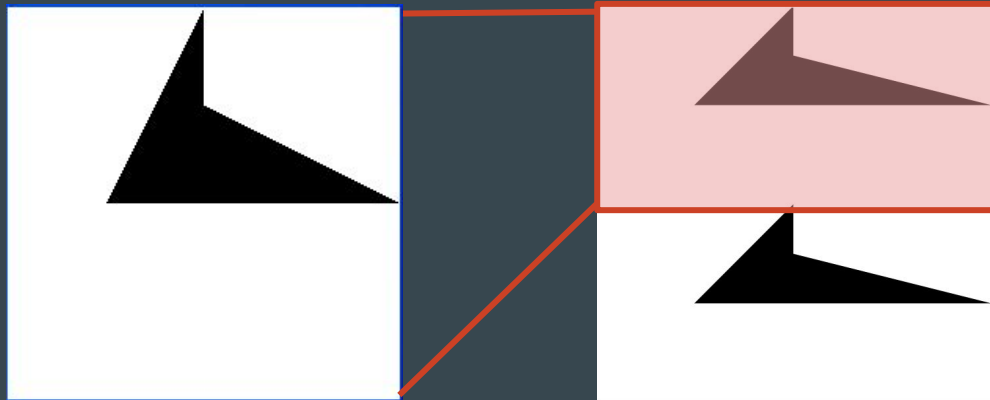
```
stack_frac(1/2, nova_bb, nova_bb)
```

Stack blocks

Top block
takes up 1/2 of
the space

Top block

Bottom block

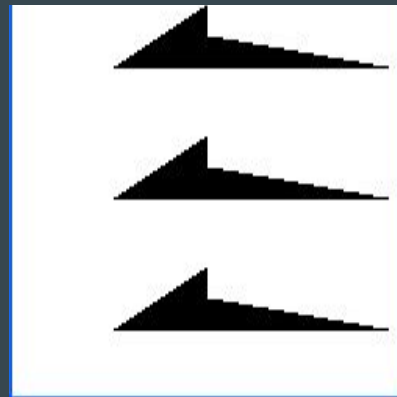


First `nova_bb` takes up
top half the space

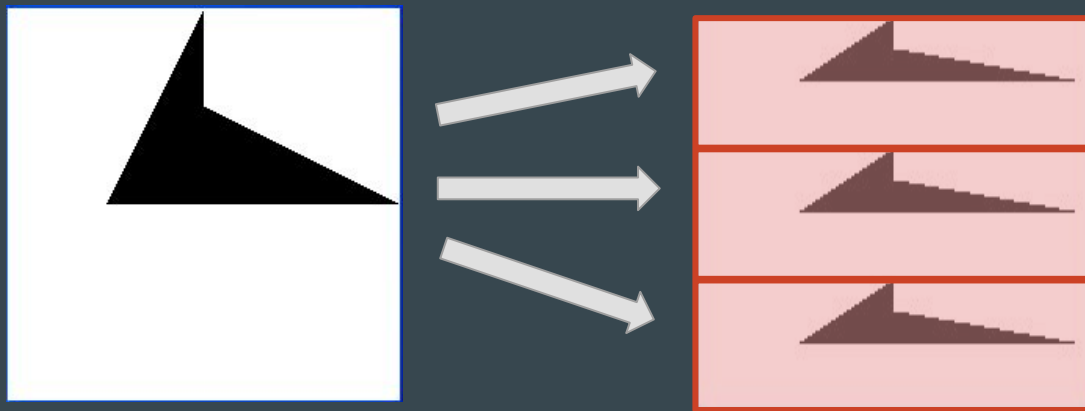
Use two identical
`nova_bb`

2nd `nova_bb` takes up
the remaining bottom

How do we make this into 3 rows then?



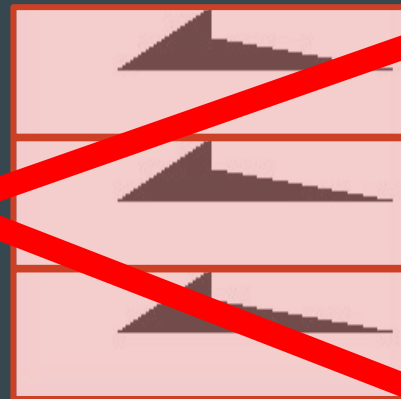
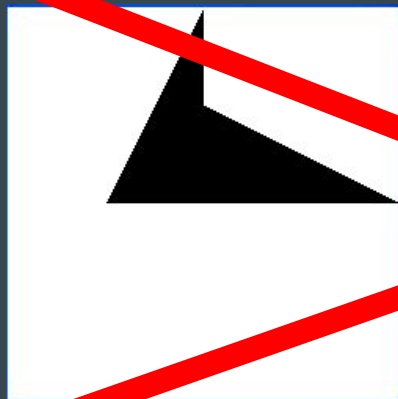
How do we make this into 3 rows then?



How do we make this into 3 rows then?

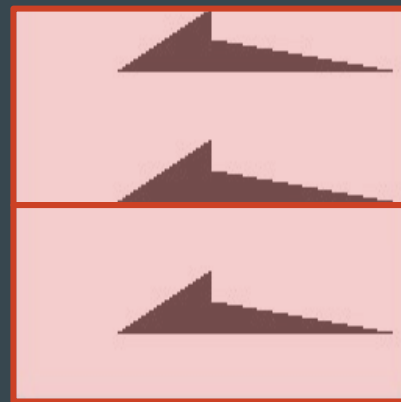
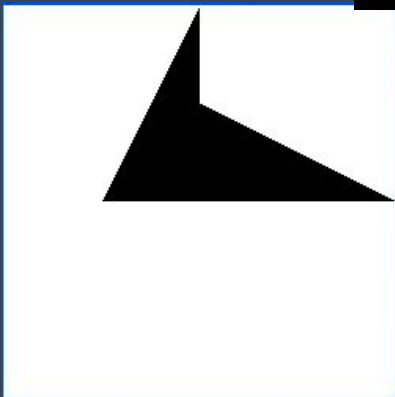


Hmm, but `stack_frac`
only allows me to
deal with 2 blocks...



How do we make this into 3 rows then?

`stack_frac(1/2, ??, ??)`

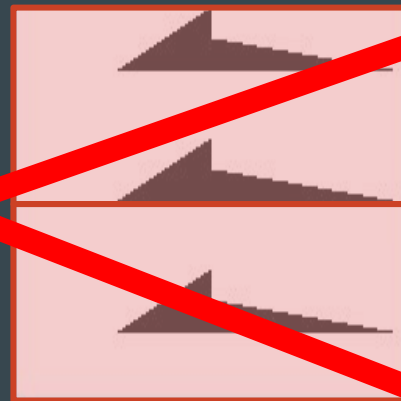
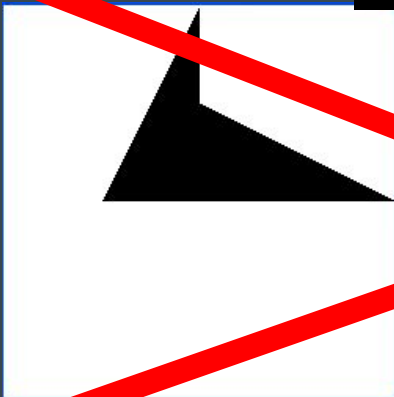


How do we make this into 3 rows then?



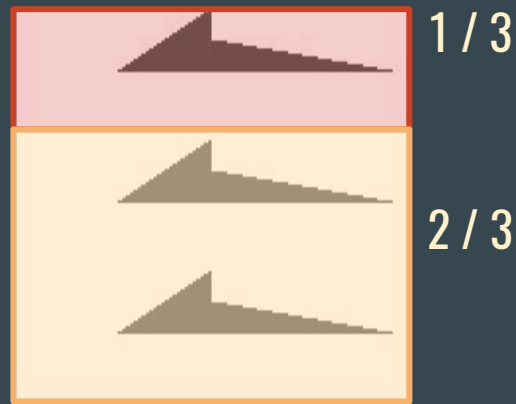
How am I going to
create two of such
patterns?

```
stack_frac(1/2, ??, ??)
```



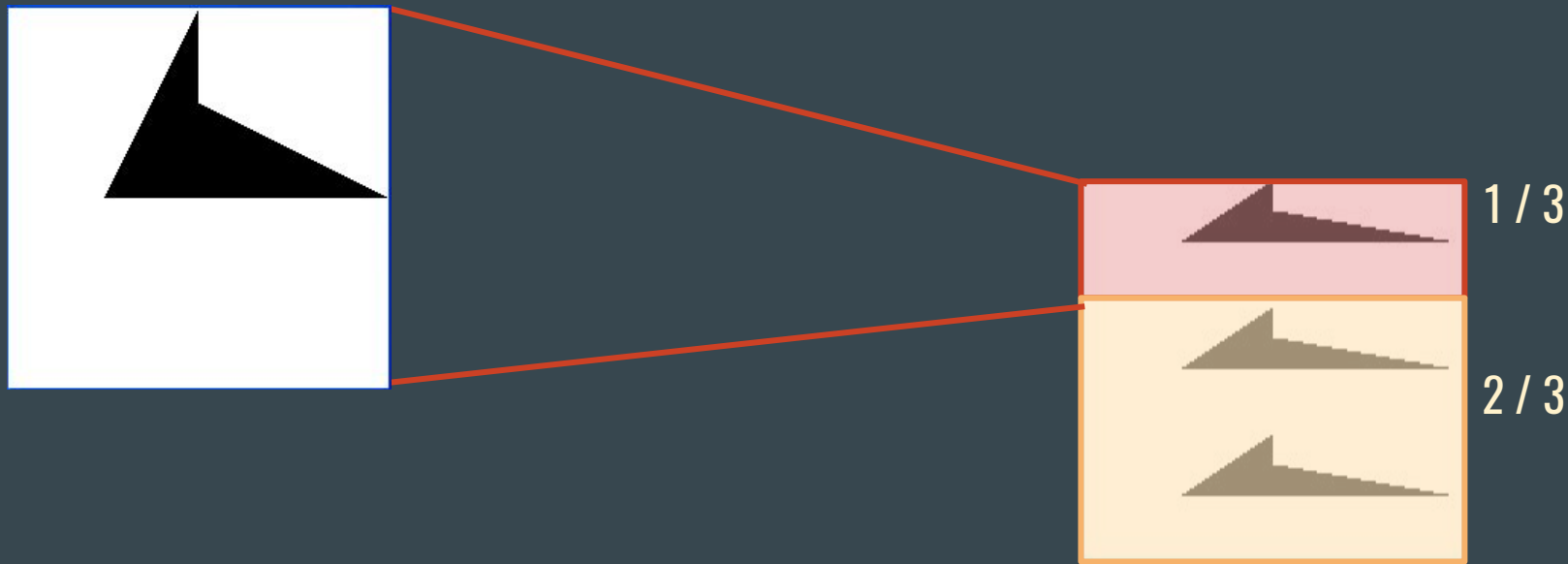
How do we make this into 3 rows then?

```
stack_frac(1/3, nova_bb, ??)
```



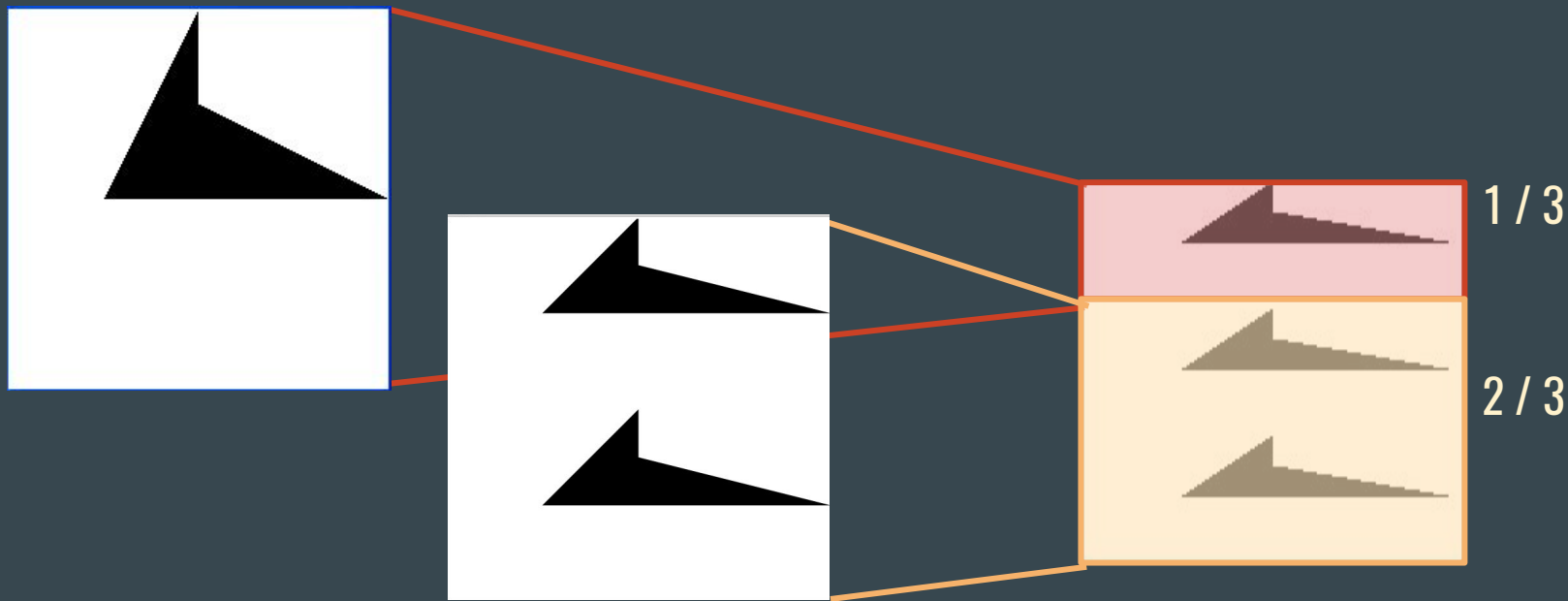
How do we make this into 3 rows then?

```
stack_frac(1/3, nova_bb, ??)
```



How do we make this into 3 rows then?

```
stack_frac(1/3, nova_bb, (stack_frac(1/2, nova_bb, nova_bb)))
```



How do we make this into 3 rows then?

```
stack_frac(1/3, nova_bb, (stack_frac(1/2, nova_bb, nova_bb)))
```



We got it, but it looks messy!



Remember Variables?

Remember
order
'5 cans of
tuna'

order = '5 cans of
tuna'



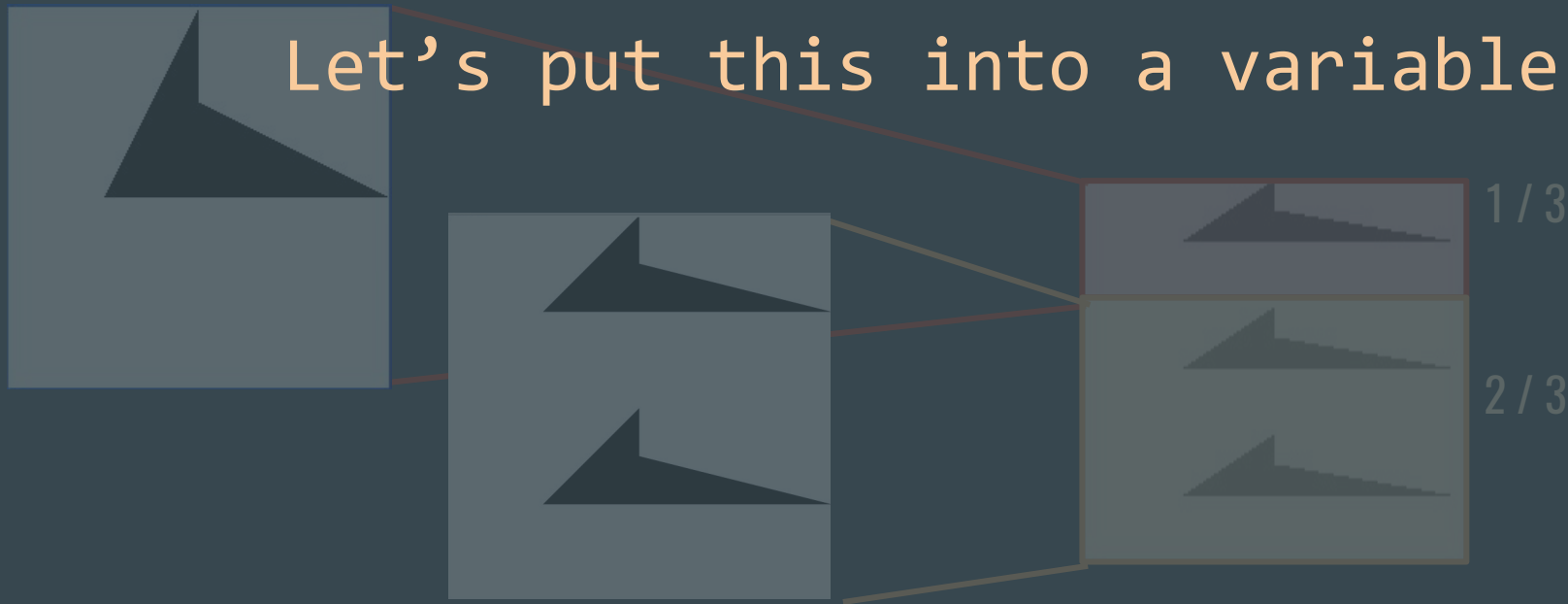
Interpreter
IDLE

```
Python 3.5.1 Shell
>>> order = '5 cans of tuna'
:
:
Ln: 21 Col: 0
```


Making it neater

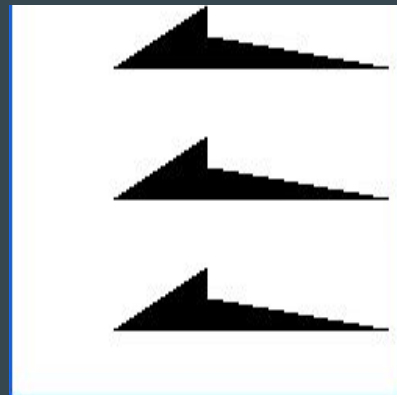
```
stack_frac(1/3, nova_bb, (stack_frac(1/2, nova_bb, nova_bb)))
```

Let's put this into a variable



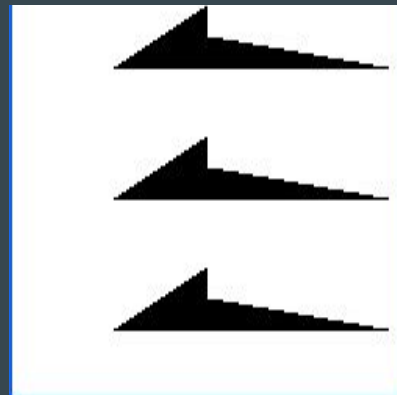
Making it neater

```
nova_2 = (stack_frac(1/2, nova_bb, nova_bb))  
stack_frac(1/3, nova_bb, (stack_frac(1/2, nova_bb, nova_bb)))
```



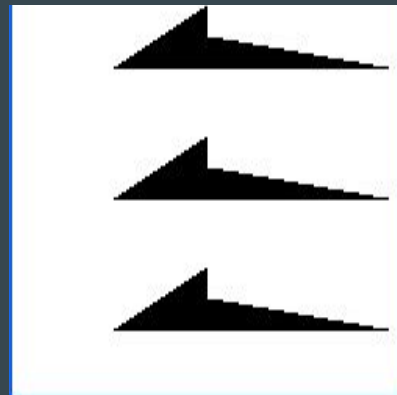
Making it neater

```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)  
stack_frac(1/3, nova_bb, nova_2)
```



Making it neater

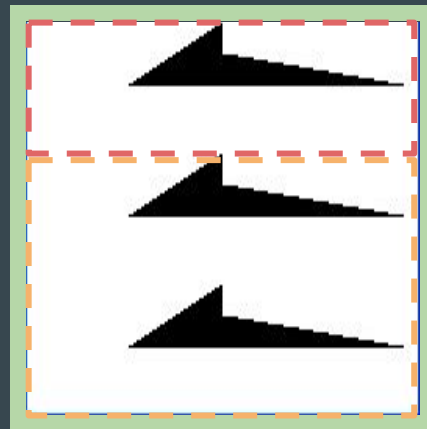
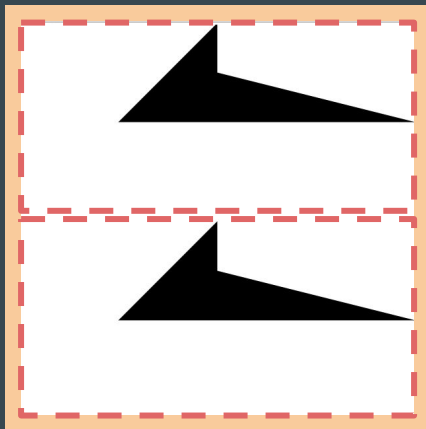
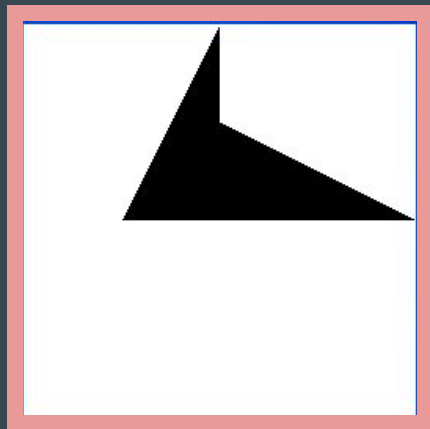
```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)  
nova_3 = stack_frac(1/3, nova_bb, nova_2)
```



What we have so far

```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)
```

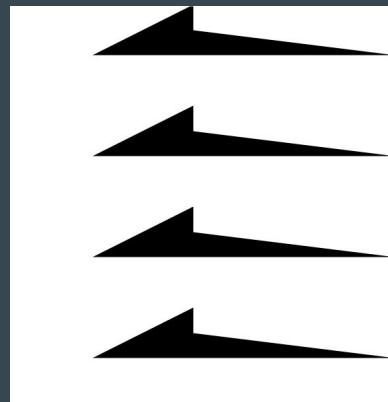
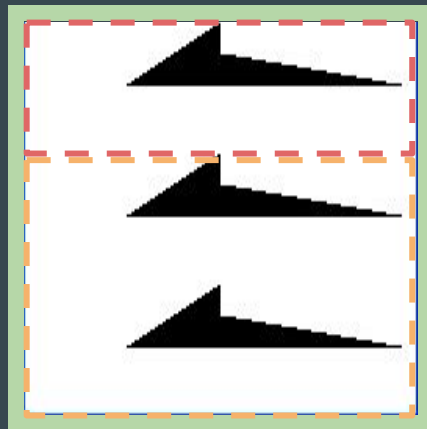
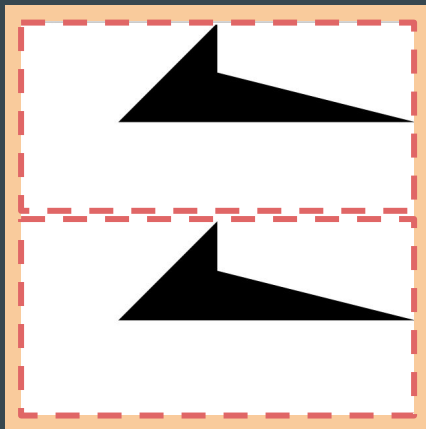
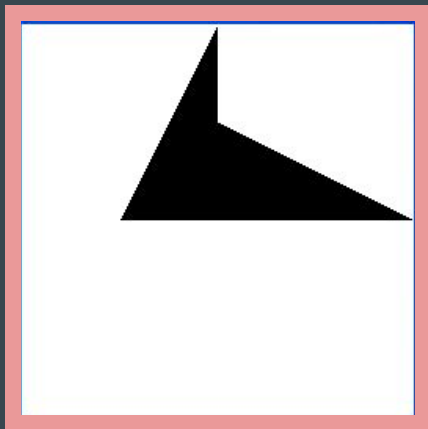
```
nova_3 = stack_frac(1/3, nova_bb, nova_2)
```



How about 4 rows?

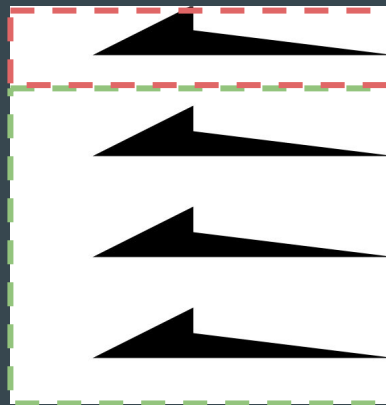
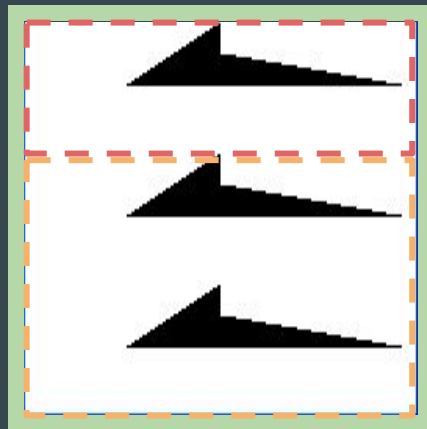
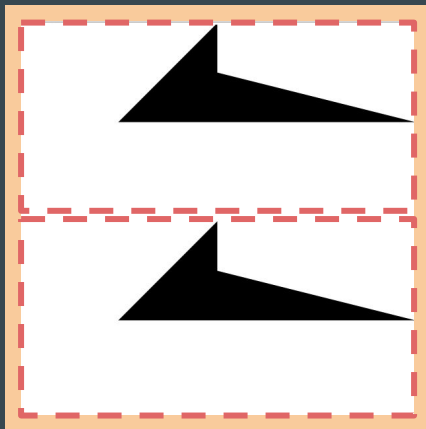
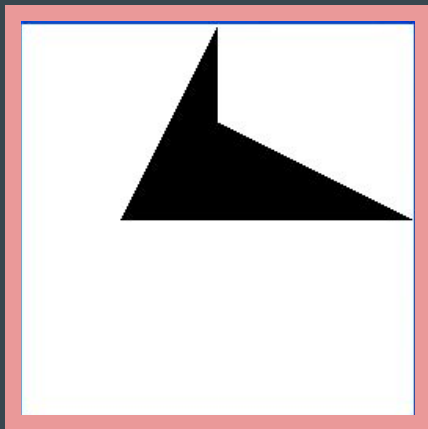
```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)
```

```
nova_3 = stack_frac(1/3, nova_bb, nova_2)
```



How about 4 rows?

```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)  
nova_3 = stack_frac(1/3, nova_bb, nova_2)  
        stack_frac(1/4, nova_bb, nova_3)
```

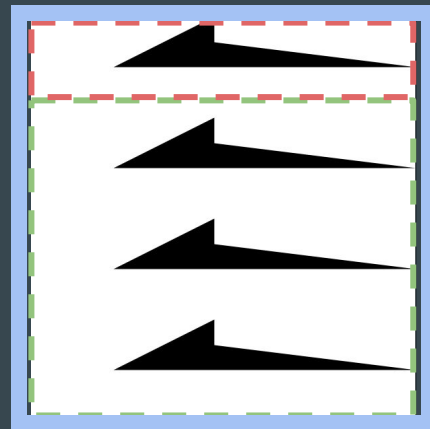
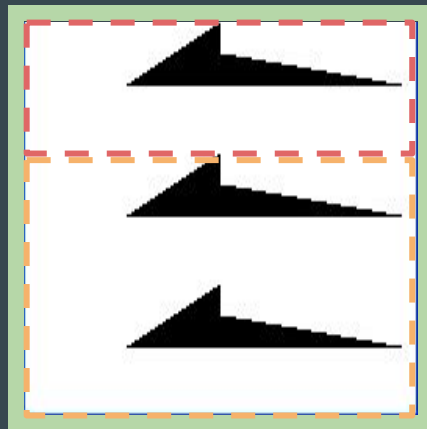
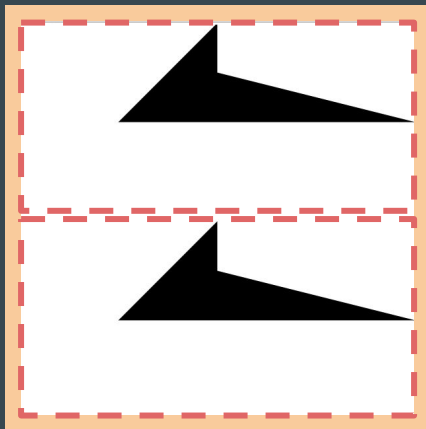
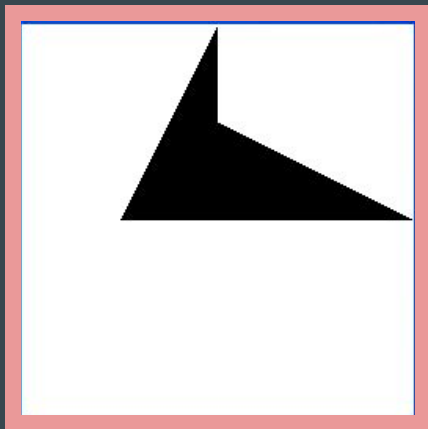


How about 4 rows?

```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)
```

```
nova_3 = stack_frac(1/3, nova_bb, nova_2)
```

```
nova_4 = stack_frac(1/4, nova_bb, nova_3)
```



And the 5th row....

```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)
nova_3 = stack_frac(1/3, nova_bb, nova_2)
nova_4 = stack_frac(1/4, nova_bb, nova_3)
nova_5 = stack_frac(1/5, nova_bb, nova_4)
```

And the 6th row....

```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)
nova_3 = stack_frac(1/3, nova_bb, nova_2)
nova_4 = stack_frac(1/4, nova_bb, nova_3)
nova_5 = stack_frac(1/5, nova_bb, nova_4)
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```

Noticed any patterns so far?

```
nova_2 = stack_frac(1/2, nova_bb, nova_bb)
nova_3 = stack_frac(1/3, nova_bb, nova_2)
nova_4 = stack_frac(1/4, nova_bb, nova_3)
nova_5 = stack_frac(1/5, nova_bb, nova_4)
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```

Noticed any patterns so far?

nova_2	=	stack_frac(1/2,	nova_bb,	nova_bb)
nova_3	=	stack_frac(1/3,	nova_bb,	nova_2)
nova_4	=	stack_frac(1/4,	nova_bb,	nova_3)
nova_5	=	stack_frac(1/5,	nova_bb,	nova_4)
nova_6	=	stack_frac(1/6,	nova_bb,	nova_5)

.

.

.

nova_n = stack_frac(1/ , ,)

Noticed any patterns so far?

nova_2	=	stack_frac(1/2,	nova_bb,	nova_bb)
nova_3	=	stack_frac(1/3,	nova_bb,	nova_2)
nova_4	=	stack_frac(1/4,	nova_bb,	nova_3)
nova_5	=	stack_frac(1/5,	nova_bb,	nova_4)
nova_6	=	stack_frac(1/6,	nova_bb,	nova_5)

.

.

.

nova_n = stack_frac(1/ , ,)

Noticed any patterns so far?

nova_2 = stack_frac(1/2, nova_bb, nova_bb)

nova_3 = stack_frac(1/3, nova_bb, nova_2)

nova_4 = stack_frac(1/4, nova_bb, nova_3)

nova_5 = stack_frac(1/5, nova_bb, nova_4)

nova_6 = stack_frac(1/6, nova_bb, nova_5)

.

.

.

nova_n = stack_frac(1/ , nova_bb,)

Noticed any patterns so far?



Noticed any patterns so far?

nova_2 = stack_frac(1/2, nova_bb, nova_bb)

nova_3 = stack_frac(1/3, nova_bb, nova_2)

nova_4 = stack_frac(1/4, nova_bb, nova_3)

nova_5 = stack_frac(1/5, nova_bb, nova_4)

nova_6 = stack_frac(1/6, nova_bb, nova_5)

.

.

.

nova_n = stack_frac(1/n, nova_bb,)

Noticed any patterns so far?

nova_2 = stack_frac(1/2, nova_bb, nova_bb)

nova_3 = stack_frac(1/3, nova_bb, nova_2)

nova_4 = stack_frac(1/4, nova_bb, nova_3)

nova_5 = stack_frac(1/5, nova_bb, nova_4)

nova_6 = stack_frac(1/6, nova_bb, nova_5)

.

.

.

nova_n = stack_frac(1/n, nova_bb,)

Noticed any patterns so far?

nova_2 = stack_frac(1/2, nova_bb, nova_bb)

nova_3 = stack_frac(1/3, nova_bb, nova_bb)

nova_4 = stack_frac(1/4, nova_bb, nova_bb)

nova_5 = stack_frac(1/5, nova_bb, nova_bb)

nova_6 = stack_frac(1/6, nova_bb, nova_bb)

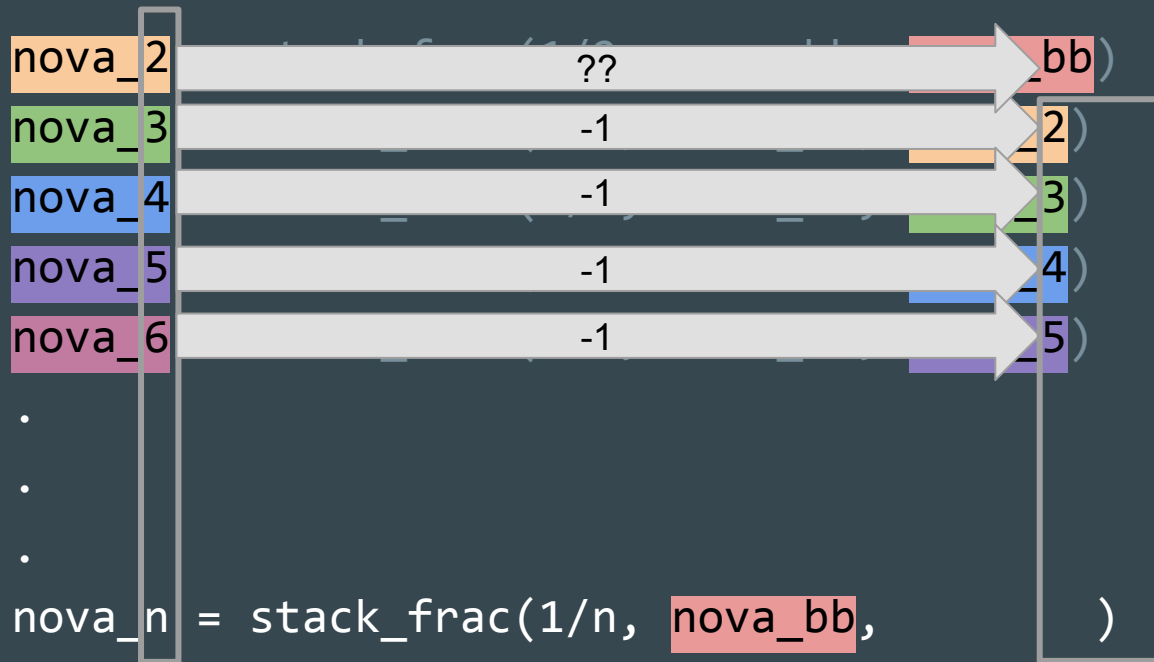
.

.

.

nova_n = stack_frac(1/n, nova_bb, nova_bb)

Noticed any patterns so far?



Noticed any patterns so far?

nova_1 = nova_bb

nova_2 = stack_frac(1/2, nova_bb, nova_bb)

nova_3 = stack_frac(1/3, nova_bb, nova_2)

nova_4 = stack_frac(1/4, nova_bb, nova_3)

nova_5 = stack_frac(1/5, nova_bb, nova_4)

nova_6 = stack_frac(1/6, nova_bb, nova_5)

.

.

.

nova_n = stack_frac(1/n, nova_bb,)

Noticed any patterns so far?

nova_1 = nova_bb

nova_2 = stack_frac(1/2, nova_bb, nova_1)

nova_3 = stack_frac(1/3, nova_bb, nova_2)

nova_4 = stack_frac(1/4, nova_bb, nova_3)

nova_5 = stack_frac(1/5, nova_bb, nova_4)

nova_6 = stack_frac(1/6, nova_bb, nova_5)

.

.

.

nova_n = stack_frac(1/n, nova_bb,)

Noticed any patterns so far?

nova_1 = nova_bb

nova_2 = stack_frac(1/2, nova_bb, nova_1)

nova_3 = stack_frac(1/3, nova_bb, nova_2)

nova_4 = stack_frac(1/4, nova_bb, nova_3)

nova_5 = stack_frac(1/5, nova_bb, nova_4)

nova_6 = stack_frac(1/6, nova_bb, nova_5)

.

.

.

nova_n = stack_frac(1/n, nova_bb,)

Noticed any patterns so far?

nova_1 = nova_bb

nova_2 = stack_frac(1/2, nova_bb, nova_1)

nova_3 = stack_frac(1/3, nova_bb, nova_2)

nova_4 = stack_frac(1/4, nova_bb, nova_3)

nova_5 = stack_frac(1/5, nova_bb, nova_4)

nova_6 = stack_frac(1/6, nova_bb, nova_5)

.

.

.

nova_n = stack_frac(1/n, nova_bb, nova_{n-1})

Noticed any patterns so far?

nova_1 = nova_bb

nova_2 = stack_frac(1/2, nova_bb, nova_1)

nova_3 = stack_frac(1/3, nova_bb, nova_2)

nova_4 = stack_frac(1/4, nova_bb, nova_3)

nova_5 = stack_frac(1/5, nova_bb, nova_4)

nova_6 = stack_frac(1/6, nova_bb, nova_5)

.

.

.

nova_n = stack_frac(1/n, nova_bb, nova_{n-1})

Noticed any patterns so far?

nova_1 = nova_bb

nova_2 $\xrightarrow{-1}$ 1)

nova_3 $\xrightarrow{-1}$ 2)

nova_4 $\xrightarrow{-1}$ 3)

nova_5 $\xrightarrow{-1}$ 4)

nova_6 $\xrightarrow{-1}$ 5)

.

.

.

nova_n = stack_frac(1/n, nova_bb,)

Noticed any patterns so far?

nova_1 = nova_bb

nova_2 $\xrightarrow{-1}$ 1)

nova_3 $\xrightarrow{-1}$ 2)

nova_4 $\xrightarrow{-1}$ 3)

nova_5 $\xrightarrow{-1}$ 4)

nova_6 $\xrightarrow{-1}$ 5)

·
·
·

nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)

This is how we stack n rows

```
nova_1 = nova_bb
```

```
nova_2 = stack_frac(1/2, nova_bb, nova_1)
```

```
nova_3 = stack_frac(1/3, nova_bb, nova_2)
```

```
nova_4 = stack_frac(1/4, nova_bb, nova_3)
```

```
nova_5 = stack_frac(1/5, nova_bb, nova_4)
```

```
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```

.

.

.

```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```



You

Follow what I
say and do it
for me!

Smaller,
Simpler,
Steps



Cool and complex stuff



You

Follow what I
say and do it
for me!

Variables

Smaller,
Simpler,
Steps



Cool and complex stuff



Updating variables

```
age = 14
```

```
age = age + 1
```

Updating variables

```
age = 14
```

```
age = age + 1
```

= is not the same as ==

Updating Variables to simplify

```
nova_1 = nova_bb
```

```
nova_2 = stack_frac(1/2, nova_bb, nova_1)
```

```
nova_3 = stack_frac(1/3, nova_bb, nova_2)
```

```
nova_4 = stack_frac(1/4, nova_bb, nova_3)
```

```
nova_5 = stack_frac(1/5, nova_bb, nova_4)
```

```
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```


.

.

.

```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```


Updating Variables to simplify



The diagram illustrates the flow of variable updates. A red box labeled 'nova_bb' has two arrows pointing to it: one from the 'nova_bb' argument in the first line of the code and another from the 'nova_1' argument in the second line. This indicates that 'nova_bb' is updated based on 'nova_1' and then used to calculate 'nova_2'.

```
nova_1 = nova_bb
nova_2 = stack_frac(1/2, nova_bb, nova_1)
nova_3 = stack_frac(1/3, nova_bb, nova_2)
nova_4 = stack_frac(1/4, nova_bb, nova_3)
nova_5 = stack_frac(1/5, nova_bb, nova_4)
nova_6 = stack_frac(1/6, nova_bb, nova_5)
.
.
.
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify

The diagram illustrates a sequence of variable updates. Each line of code is highlighted with a different background color. Colored arrows point from the variable being updated to the variable it depends on. For example, a red arrow points from `nova_1` to `nova_bb` in the second line, and a yellow arrow points from `nova_2` to `nova_1` in the third line.

```
nova_1 = nova_bb  
nova_2 = stack_frac(1/2, nova_bb, nova_1)  
nova_3 = stack_frac(1/3, nova_bb, nova_2)  
nova_4 = stack_frac(1/4, nova_bb, nova_3)  
nova_5 = stack_frac(1/5, nova_bb, nova_4)  
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```

.

.

.

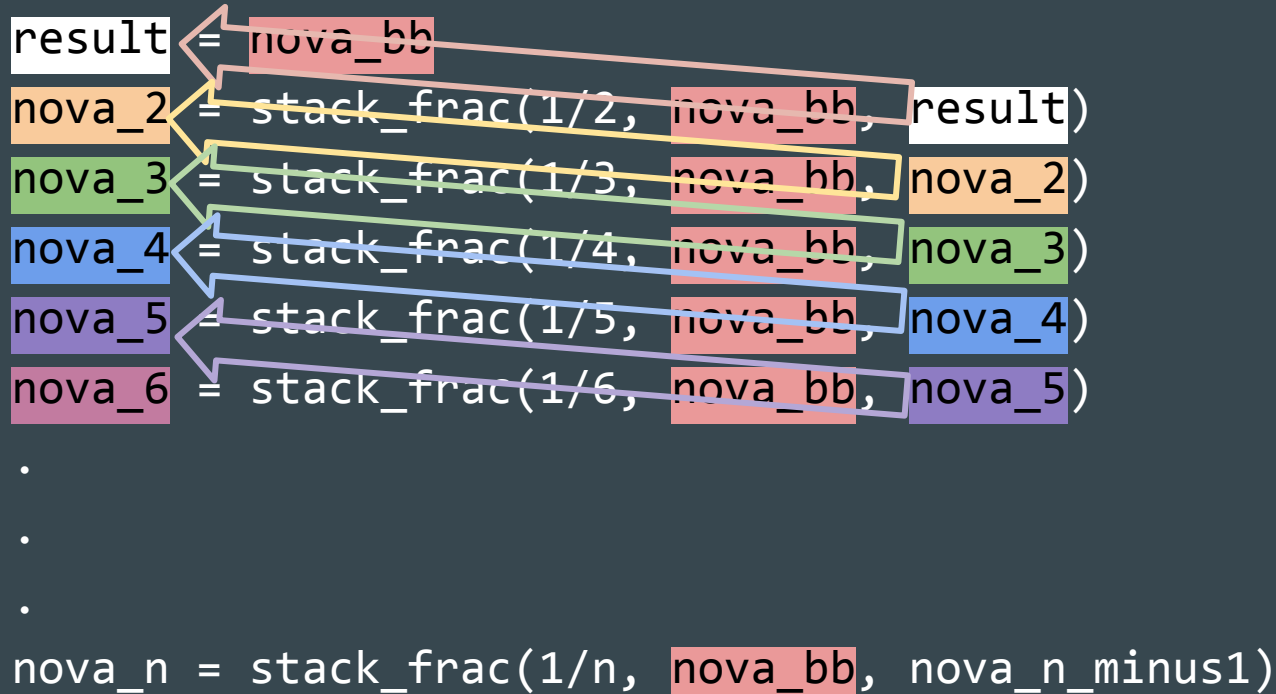
```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify

The diagram illustrates a sequence of variable updates. Colored boxes highlight the variables being updated: nova_bb (red), nova_2 (orange), nova_3 (green), nova_4 (blue), nova_5 (purple), and nova_6 (pink). Arrows show the flow of data: nova_bb is used in all stack_frac calls. Each nova_n is calculated using nova_bb and the previous nova_{n-1} value. The final line shows a general update for nova_n.

```
result = nova_bb  
nova_2 = stack_frac(1/2, nova_bb, nova_1)  
nova_3 = stack_frac(1/3, nova_bb, nova_2)  
nova_4 = stack_frac(1/4, nova_bb, nova_3)  
nova_5 = stack_frac(1/5, nova_bb, nova_4)  
nova_6 = stack_frac(1/6, nova_bb, nova_5)  
.  
.  
.  
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify



The diagram illustrates a sequence of variable updates. Each line of code is highlighted with a different background color. Colored arrows point from the variable names in the function calls to their previous definitions, showing the flow of data and dependencies. The variables are: `result` (white), `nova_bb` (red), `nova_2` (orange), `nova_3` (green), `nova_4` (blue), `nova_5` (purple), and `nova_6` (pink). The arrows show that `result` depends on `nova_bb`, `nova_2` depends on `nova_bb` and `result`, `nova_3` depends on `nova_bb` and `nova_2`, `nova_4` depends on `nova_bb` and `nova_3`, `nova_5` depends on `nova_bb` and `nova_4`, and `nova_6` depends on `nova_bb` and `nova_5`.

```
result = nova_bb
nova_2 = stack_frac(1/2, nova_bb, result)
nova_3 = stack_frac(1/3, nova_bb, nova_2)
nova_4 = stack_frac(1/4, nova_bb, nova_3)
nova_5 = stack_frac(1/5, nova_bb, nova_4)
nova_6 = stack_frac(1/6, nova_bb, nova_5)
.
.
.
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify

```
result = nova_bb  
result = stack_frac(1/2, nova_bb, result)  
nova_3 = stack_frac(1/3, nova_bb, nova_2)  
nova_4 = stack_frac(1/4, nova_bb, nova_3)  
nova_5 = stack_frac(1/5, nova_bb, nova_4)  
nova_6 = stack_frac(1/6, nova_bb, nova_5)  
.  
.  
.  
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

The diagram illustrates the flow of variable updates. Colored boxes highlight the variables involved in each line: `result` (white), `nova_bb` (red), `nova_2` (orange), `nova_3` (green), `nova_4` (blue), `nova_5` (purple), and `nova_6` (pink). Arrows show the sequence of updates: `result` is updated from `nova_bb` in the second line. Then, `nova_3` is updated from `nova_bb` and `nova_2` in the third line. `nova_4` is updated from `nova_bb` and `nova_3` in the fourth line. `nova_5` is updated from `nova_bb` and `nova_4` in the fifth line. Finally, `nova_6` is updated from `nova_bb` and `nova_5` in the sixth line. The general case at the bottom shows `nova_n` being updated from `nova_bb` and `nova_n_minus1`.

Updating Variables to simplify

```
result = nova_bb
```

```
result = stack_frac(1/2, nova_bb, result)
```

```
nova_3 = stack_frac(1/3, nova_bb, result)
```

```
nova_4 = stack_frac(1/4, nova_bb, nova_3)
```

```
nova_5 = stack_frac(1/5, nova_bb, nova_4)
```

```
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```

.

.

.

```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify

```
result = nova_bb
```

```
result = stack_frac(1/2, nova_bb, result)
```

```
nova_3 ← stack_frac(1/3, nova_bb, result)
```

```
nova_4 ← stack_frac(1/4, nova_bb, nova_3)
```

```
nova_5 ← stack_frac(1/5, nova_bb, nova_4)
```

```
nova_6 ← stack_frac(1/6, nova_bb, nova_5)
```

.

.

.

```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify

```
result = nova_bb
```

```
result = stack_frac(1/2, nova_bb, result)
```

```
result = stack_frac(1/3, nova_bb, result)
```

```
nova_4 = stack_frac(1/4, nova_bb, nova_3)
```

```
nova_5 = stack_frac(1/5, nova_bb, nova_4)
```

```
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```

.

.

.

```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```


Updating Variables to simplify

```
result = nova_bb
```

```
result = stack_frac(1/2, nova_bb, result)
```

```
result = stack_frac(1/3, nova_bb, result)
```

```
nova_4 = stack_frac(1/4, nova_bb, result)
```

```
nova_5 = stack_frac(1/5, nova_bb, nova_4)
```

```
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```

.

.

.

```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify

```
result = nova_bb
```

```
result = stack_frac(1/2, nova_bb, result)
```

```
result = stack_frac(1/3, nova_bb, result)
```

```
nova_4 = stack_frac(1/4, nova_bb, result)
```

```
nova_5 = stack_frac(1/5, nova_bb, nova_4)
```

```
nova_6 = stack_frac(1/6, nova_bb, nova_5)
```

.

.

.

```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify

```
result = nova_bb
```

```
result = stack_frac(1/2, nova_bb, result)
```

```
result = stack_frac(1/3, nova_bb, result)
```

```
result = stack_frac(1/4, nova_bb, result)
```

```
result = stack_frac(1/5, nova_bb, result)
```

```
result = stack_frac(1/6, nova_bb, result)
```

```
.
```

```
.
```

```
.
```

```
nova_n = stack_frac(1/n, nova_bb, nova_n_minus1)
```

Updating Variables to simplify

```
result = nova_bb
```

```
result = stack_frac(1/2, nova_bb, result)
```

```
result = stack_frac(1/3, nova_bb, result)
```

```
result = stack_frac(1/4, nova_bb, result)
```

```
result = stack_frac(1/5, nova_bb, result)
```

```
result = stack_frac(1/6, nova_bb, result)
```

```
.
```

```
.
```

```
.
```

```
result = stack_frac(1/n, nova_bb, result)
```

Updating Variables to simplify

```
result = nova_bb  
result = stack_frac(1/2, nova_bb, result)  
result = stack_frac(1/3, nova_bb, result)  
result = stack_frac(1/4, nova_bb, result)  
result = stack_frac(1/5, nova_bb, result)  
result = stack_frac(1/6, nova_bb, result)  
.  
.  
.  
result = stack_frac(1/n, nova_bb, result)
```



You

Follow what I
say and do it
for me!

Loops



Smaller,
Simpler,
Steps



Cool and complex stuff

Loops recap

```
print(0)
```

```
print(1)
```

```
print(2)
```

```
print(3)
```

```
print(4)
```

Loops recap

```
for number in range(5):  
    print(number)
```


Loops recap

```
for number in range(5):  
    print(number)
```

number is 0

Loops recap

```
for number in range(5):  
    print(number)
```

number is 0 → print(0) → 0

Loops recap

```
for number in range(5):  
    print(number)
```

number is 0 → print(0) → 0

→ number is 1

Loops recap

```
for number in range(5):  
    print(number)
```

number is 0 → print(0) → 0

→ number is 1 → print(1) → 1

Loops recap

```
for number in range(5):  
    print(number)
```

number is 0 → print(0) → 0

→ number is 1 → print(1) → 1

→ number is 2

Loops recap

```
for number in range(5):  
    print(number)
```

number is 0 → print(0) → 0

→ number is 1 → print(1) → 1

→ number is 2 → print(2) → 2

Loops recap

```
for number in range(5):  
    print(number)
```

number is 0 → print(0) → 0

→ number is 1 → print(1) → 1

→ number is 2 → print(2) → 2

→ number is 3 → print(3) → 3

→ number is 4 → print(4) → 4

→ number is 5

Loops recap

```
for number in range(5):  
    print(number)
```

```
for number in range(5):  
    print(number+2)
```

number is 0 → print(0) → 0
→ number is 1 → print(1) → 1
→ number is 2 → print(2) → 2
→ number is 3 → print(3) → 3
→ number is 4 → print(4) → 4
→ number is 5

Loops recap

```
for number in range(5):  
    print(number)
```

number is 0 → print(0) → 0
→ number is 1 → print(1) → 1
→ number is 2 → print(2) → 2
→ number is 3 → print(3) → 3
→ number is 4 → print(4) → 4
→ number is 5

```
for number in range(5):  
    print(number+2)
```

number is 0 → print(0+2) → 2

Loops recap

```
for number in range(5):  
    print(number)
```

number is 0 → print(0) → 0
→ number is 1 → print(1) → 1
→ number is 2 → print(2) → 2
→ number is 3 → print(3) → 3
→ number is 4 → print(4) → 4
→ number is 5

```
for number in range(5):  
    print(number+2)
```

number is 0 → print(0+2) → 2
→ number is 1 → print(1+2) → 3
→ number is 2 → print(2+2) → 4
→ number is 3 → print(3+2) → 5
→ number is 4 → print(4+2) → 6
→ number is 5

Using Loops to simplify

```
result = nova_bb  
result = stack_frac(1/2, nova_bb, result)  
result = stack_frac(1/3, nova_bb, result)  
result = stack_frac(1/4, nova_bb, result)  
result = stack_frac(1/5, nova_bb, result)  
result = stack_frac(1/6, nova_bb, result)  
.  
.  
.  
result = stack_frac(1/n, nova_bb, result)
```

Using Loops to simplify

```
result = nova_bb  
result = stack_frac(1/2, nova_bb, result)  
result = stack_frac(1/3, nova_bb, result)  
result = stack_frac(1/4, nova_bb, result)  
result = stack_frac(1/5, nova_bb, result)  
result = stack_frac(1/6, nova_bb, result)  
.  
.  
.  
result = stack_frac(1/n, nova_bb, result)
```

Using Loops to simplify

```
result = nova_bb  
result = stack_frac(1/2, nova_bb, result)  
result = stack_frac(1/3, nova_bb, result)  
result = stack_frac(1/4, nova_bb, result)  
result = stack_frac(1/5, nova_bb, result)  
result = stack_frac(1/6, nova_bb, result)  
.  
.  
.  
result = stack_frac(1/n, nova_bb, result)
```

Using Loops to simplify

```
result = nova_bb
```

```
number is 0 → result = stack_frac(1/2, nova_bb, result)
```

```
number is 1 → result = stack_frac(1/3, nova_bb, result)
```

```
number is 2 → result = stack_frac(1/4, nova_bb, result)
```

```
number is 3 → result = stack_frac(1/5, nova_bb, result)
```

```
number is 4 → result = stack_frac(1/6, nova_bb, result)
```

```
.
```

```
.
```

```
.
```

```
result = stack_frac(1/n, nova_bb, result)
```

Using Loops to simplify

```
result = nova_bb
```

```
number is 0 → +2 → 2, nova_bb, result)
number is 1 → +2 → 3, nova_bb, result)
number is 2 → +2 → 4, nova_bb, result)
number is 3 → +2 → 5, nova_bb, result)
number is 4 → +2 → 6, nova_bb, result)
.
.
.
number is n-2 → +2 → n, nova_bb, result)
```

Using Loops to simplify

```
result = nova_bb
```

```
number is 0 → result = stack_frac(1/(number+2), nova_bb, result)
```

```
number is 1 → result = stack_frac(1/(number+2), nova_bb, result)
```

```
number is 2 → result = stack_frac(1/(number+2), nova_bb, result)
```

```
number is 3 → result = stack_frac(1/(number+2), nova_bb, result)
```

```
number is 4 → result = stack_frac(1/(number+2), nova_bb, result)
```

```
•
```

```
•
```

```
•
```

```
•
```

```
•
```

```
•
```

```
result = stack_frac(1/(number+2), nova_bb, result)
```


Using Loops to simplify

```
result = nova_bb
```

```
result = stack_frac(1/(number+2), nova_bb, result)
```

Using Loops to simplify

```
result = nova_bb  
for number in range(n):  
    result = stack_frac(1/(number+2), nova_bb, result)
```

Using Loops to simplify

```
result = nova_bb  
for number in range(n):  
    result = stack_frac(1/(number+2), nova_bb, result)  
show(result)
```

Using Loops to simplify

```
result = nova_bb
for number in range(n):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

Example

```
result = nova_bb  
for number in range(2):  
    result = stack_frac(1/(number+2), nova_bb, result)  
show(result)
```

Example

```
result = nova_bb  
for number in range(2):  
    result = stack_frac(1/(number+2), nova_bb, result)  
show(result)
```

Example

```
result = nova_bb
```

```
for number in range(2):
```

```
    result = stack_frac(1/(number+2), nova_bb, result)
```

```
show(result)
```

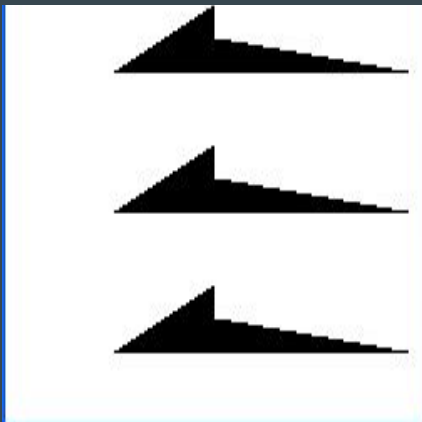
number is 0 → result = stack_frac(1/2, nova_bb, result)

number is 1 → result = stack_frac(1/3, nova_bb, result)

number is 2

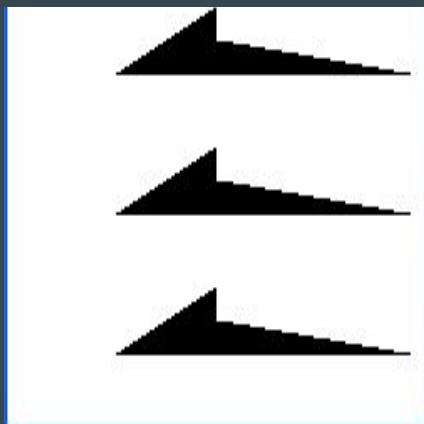
Example

```
result = nova_bb  
for number in range(2):  
    result = stack_frac(1/(number+2), nova_bb, result)  
show(result)
```



Refining further

```
result = nova_bb  
for number in range(2):  
    result = stack_frac(1/(number+2), nova_bb, result)  
show(result)
```



3 rows

Refining further

```
result = nova_bb  
for number in range(n):  
    result = stack_frac(1/(number+2), nova_bb, result)  
show(result)
```



n+1 rows

So that

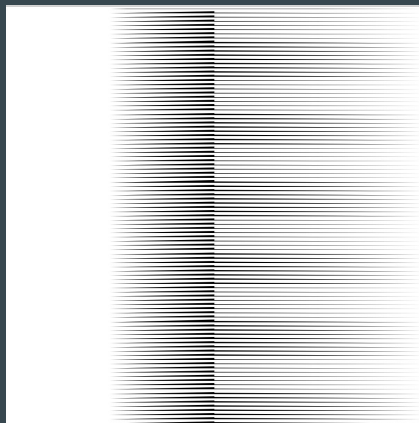
```
result = nova_bb  
for number in range(n-1):  
    result = stack_frac(1/(number+2), nova_bb, result)  
show(result)
```



n rows

So that when $n = 99$,

```
result = nova_bb  
for number in range(99-1):  
    result = stack_frac(1/(number+2), nova_bb, result)  
show(result)
```



99 rows

See how much we have simplified our code

99 rows of nova_bb - Before

```
nova_1 = nova_bb
nova_2 = stack_frac(1/2, nova_bb, nova_1)
nova_3 = stack_frac(1/3, nova_bb, nova_2)
nova_4 = stack_frac(1/4, nova_bb, nova_3)
nova_5 = stack_frac(1/5, nova_bb, nova_4)
nova_6 = stack_frac(1/6, nova_bb, nova_5)
nova_7 = stack_frac(1/7, nova_bb, nova_6)
nova_8 = stack_frac(1/8, nova_bb, nova_7)
nova_9 = stack_frac(1/9, nova_bb, nova_8)
.
.
.
nova_99 = stack_frac(1/99, nova_bb, nova_98)
```

99 rows of nova_bb - After

```
result = nova_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

Recap



You

Follow what I
say and do it
for me!

Smaller,
Simpler,
Steps



Cool and complex stuff

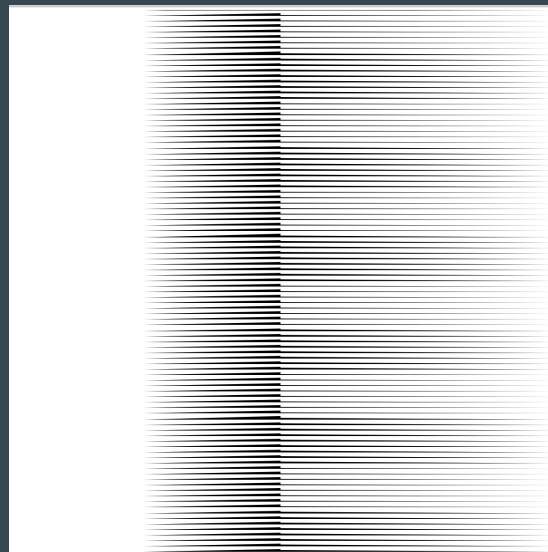
Recap



You

Follow what I
say and do it
for me!

Smaller,
Simpler,
Steps



Cool and complex stuff

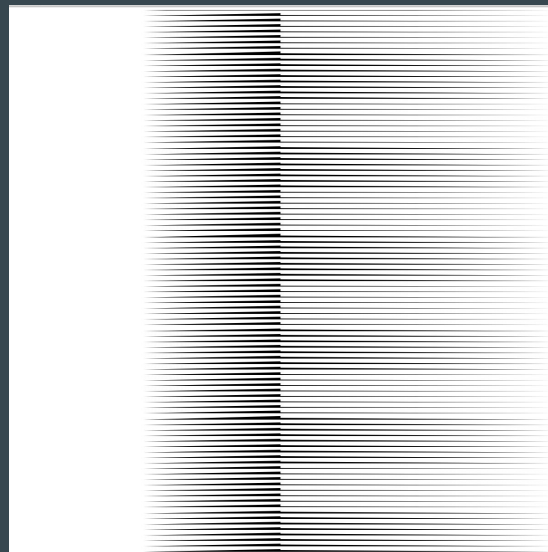
Instead of...



You

Step 1...
Step 2...
Step 3...

Repeat
similar
steps



Cool and complex stuff

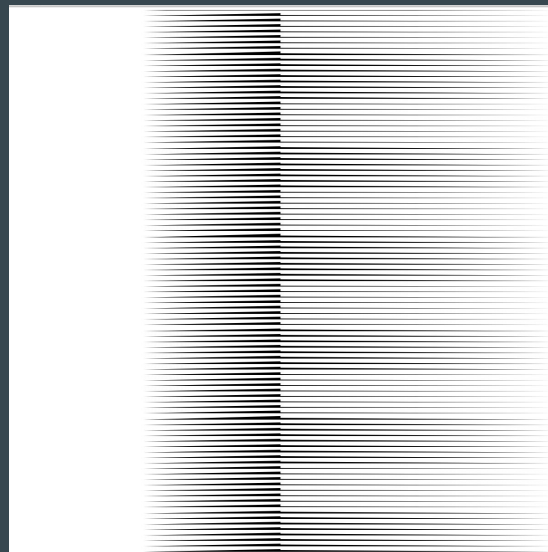
We did...



You

Update variable.

Repeat
similar
steps



Cool and complex stuff

We did...

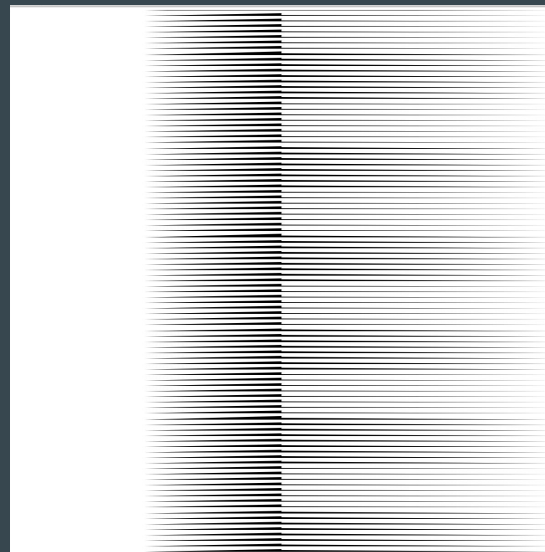


You

Loop this.
Update variable.



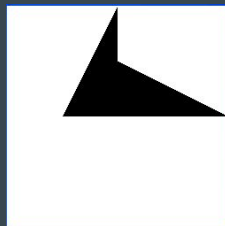
Repeat
similar
steps



Cool and complex stuff

In the last video...

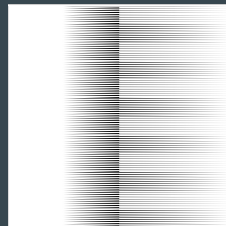
How do we use **loops** to repeat the same steps to create complex patterns?



nova_bb



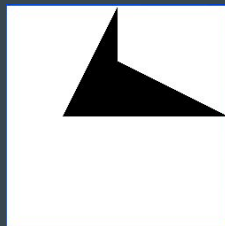
Repeat using
loops



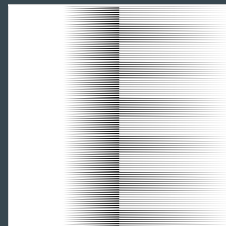
99 rows of nova_bb

In this video...

Can we reuse our code to produce do similar things to different runes?



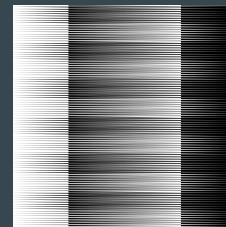
nova_bb



99 rows of nova_bb



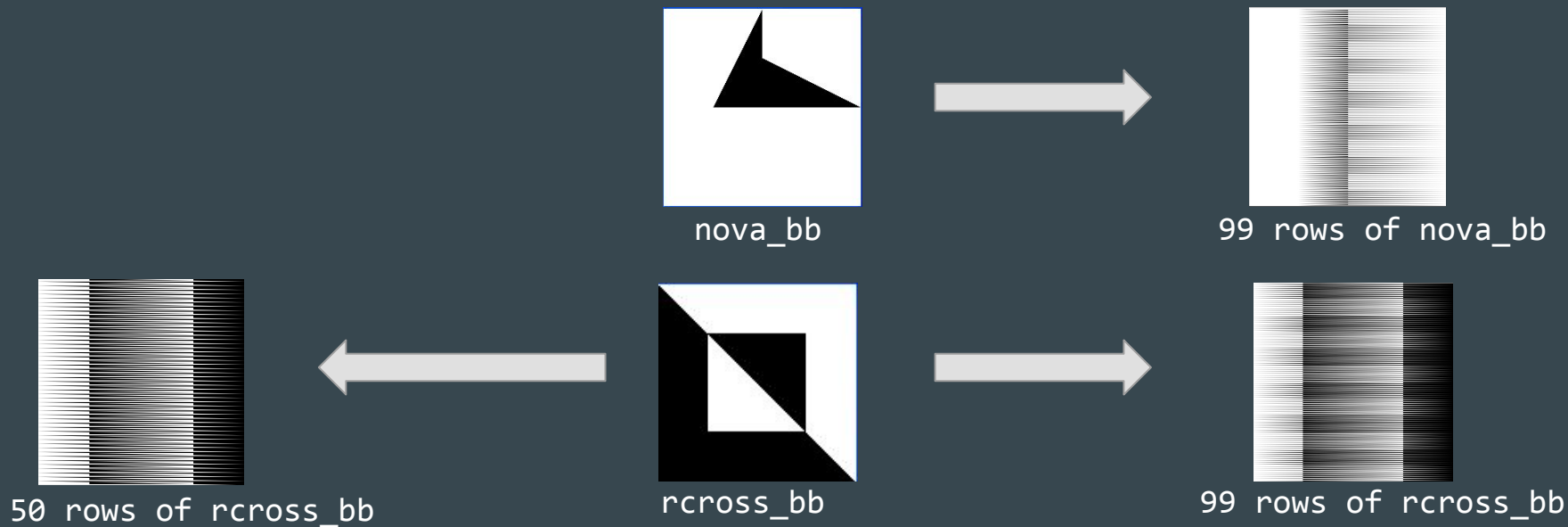
rcross_bb



99 rows of rcross_bb

In this video...

Can we reuse our code to produce do similar things to the same runes?



Can we simplify this further to use in more situations?

99 rows of nova_bb

```
result = nova_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

99 rows of circle_bb

```
result = circle_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

95 rows of nova_bb

```
result = nova_bb
for number in range(95-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

60 rows of circle_bb

```
result = circle_bb
for number in range(60-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

Recap - functions with parameters

```
def area_of_rectangle(length, breadth):  
    return length * breadth
```

Recap - functions with parameters

```
def area_of_rectangle(length, breadth):  
    return length * breadth
```



```
area_of_rectangle(length, breadth):  
    return length * breadth
```

area_of_rectangle(3, 5)



Recap - functions with parameters

```
def area_of_rectangle(length, breadth):  
    return length * breadth
```

```
area_of_rectangle(3, breadth):  
    return 3 * breadth
```

area_of_rectangle(3, 5)



Recap - functions with parameters

```
def area_of_rectangle(length, breadth):  
    return length * breadth
```

```
area_of_rectangle(3, 5):  
    return 3 * 5
```

area_of_rectangle(3, 5)



What changes?

99 rows of nova_bb

```
result = nova_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

99 rows of circle_bb

```
result = circle_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

95 rows of nova_bb

```
result = nova_bb
for number in range(95-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

60 rows of circle_bb

```
result = circle_bb
for number in range(60-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

What changes?

Number of rows

99 rows of nova_bb

```
result = nova_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

99 rows of circle_bb

```
result = circle_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

95 rows of nova_bb

```
result = nova_bb
for number in range(95-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

60 rows of circle_bb

```
result = circle_bb
for number in range(60-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

What changes?

Number of rows

Type of rune

99 rows of nova_bb

```
result = nova_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

99 rows of circle_bb

```
result = circle_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

95 rows of nova_bb

```
result = nova_bb
for number in range(95-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

60 rows of circle_bb

```
result = circle_bb
for number in range(60-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

parameters:

Number of rows

Type of rune

```
result = nova_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

```
result = nova_bb
for number in range(95-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

```
result = circle_bb
for number in range(99-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

```
result = circle_bb
for number in range(60-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

parameters:

n

Type of rune

```
result = nova_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

```
result = nova_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

```
result = circle_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

```
result = circle_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

parameters:

n

Type of rune

```
result = nova_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```



Similar code

```
result = nova_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

```
result = circle_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```



Similar code

```
result = circle_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```


parameters:

n

Type of rune

```
result = nova_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), nova_bb, result)
show(result)
```

```
result = circle_bb
for number in range(n-1):
    result = stack_frac(1/(number+2), circle_bb, result)
show(result)
```

parameters:

n

rune

```
result = rune
for number in range(n-1):
    result = stack_frac(1/(number+2), rune, result)
show(result)
```

```
result = rune
for number in range(n-1):
    result = stack_frac(1/(number+2), rune, result)
show(result)
```

parameters:

n

rune

Similar code

```
result = rune
for number in range(n-1):
    result = stack_frac(1/(number+2), rune, result)
show(result)
```



```
result = rune
for number in range(n-1):
    result = stack_frac(1/(number+2), rune, result)
show(result)
```

parameters:

n

rune

```
result = rune
```


```
for number in range(n-1):
```

```
result = stack_frac(1/(number+2), rune, result)
```

```
show(result)
```

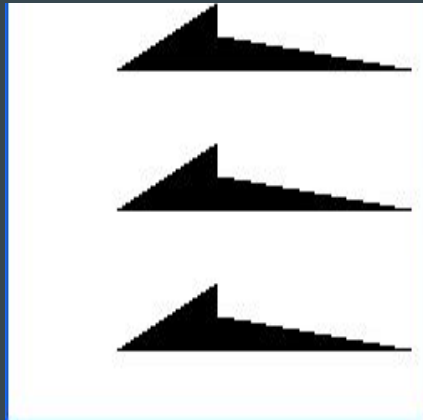
```
def stacker(n, rune):  
    result = rune  
  
    for number in range(n-1):  
        result = stack_frac(1/(number+2), rune, result)  
    show(result)
```

```
def stacker(n, rune):  
    result = rune  
  
    for number in range(n-1):  
        result = stack_frac(1/(number+2), rune, result)  
    show(result)
```



stacker(3, nova_bb)


```
def stacker(n, rune):  
    result = rune  
    for number in range(n-1):  
        result = stack_frac(1/(number+2), rune, result)  
    show(result)
```



`stacker(3, nova_bb)`

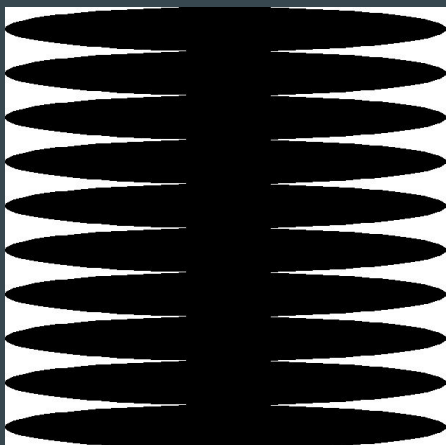


```
def stacker(n, rune):  
    result = rune  
  
    for number in range(n-1):  
        result = stack_frac(1/(number+2), rune, result)  
    show(result)
```



stacker(10, circle_bb)


```
def stacker(n, rune):  
    result = rune  
    for number in range(n-1):  
        result = stack_frac(1/(number+2), rune, result)  
    show(result)
```



`stacker(10, circle_bb)`

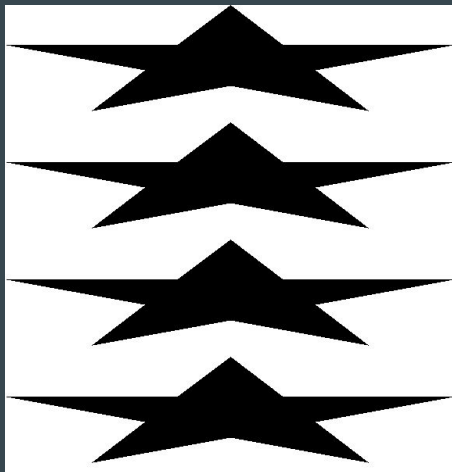


```
def stacker(n, rune):  
    result = rune  
  
    for number in range(n-1):  
        result = stack_frac(1/(number+2), rune, result)  
    show(result)
```

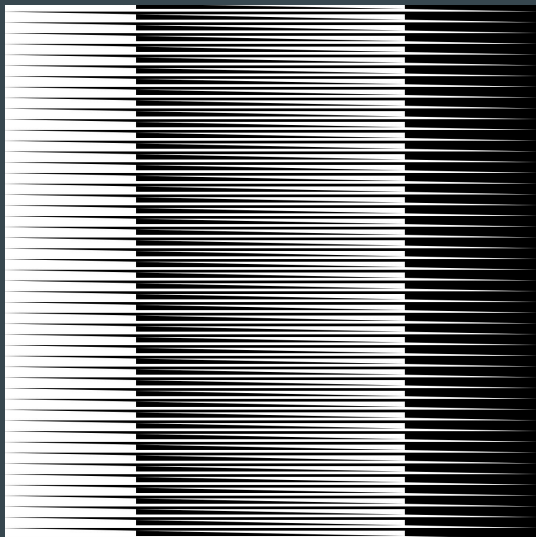
`stacker(4, pentagram_bb)`



```
def stacker(n, rune):  
    result = rune  
    for number in range(n-1):  
        result = stack_frac(1/(number+2), rune, result)  
    show(result)
```



```
def stacker(n, rune):  
    result = rune  
    for number in range(n-1):  
        result = stack_frac(1/(number+2), rune, result)  
    show(result)
```



50 rows of rcross_bb

stacker(??)



Summary



You

Follow what I
say and do it
for me!

Smaller,
Simpler,
Steps



Cool and complex stuff

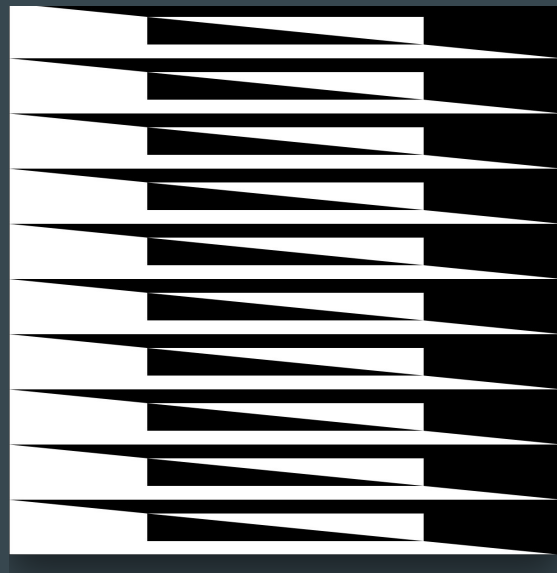
Last video



You

Loop this.
Update variable.

Repeat
similar
steps



Cool and complex stuff

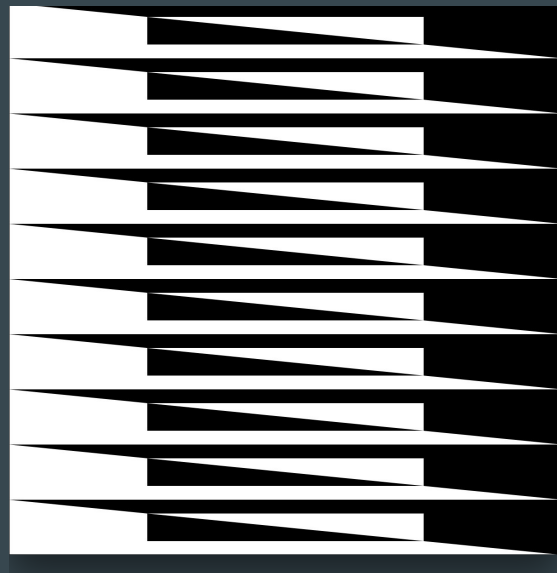
This video



You

Functions with
parameters
Loop this.
Update variable.

Repeat
similar
steps



Cool and complex stuff