

Picture Language

...

In 2D

Saving Your Code

```
Python 3.5.1 Shell - /Users/brian/Desktop/Untitled.py (3.5.1)
>>> pi = 3.14
>>> radius = 7
>>> radius
7
>>> area = pi * radius * radius
>>> area
153.86
>>> circumference = 2 * pi * radius
>>> circumference
43.96
>>>
```

Ln: 166 Col: 4

```
circle.py - /Users/briankoh/Documents/circle.py (3.5.1)
pi = 3.14

radius = 7
print(radius)

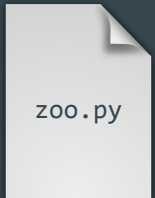
area = pi * radius * radius
print(area)

circumference = 2 * pi * radius
print(circumference)
```

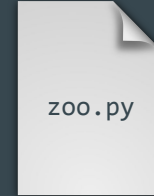
circle.py

Ln: 11 Col: 0

Sharing code



Sharing code



I made `print_elephant` and
`print_giraffe`!

Blackboxing and importing

[illegible]

```
animals_say.py - /Users/briankoh/Documents/animals_say.py (...)
```

```
from animals import *  
from zoo import print_elephant  
from zoo import print_giraffe  
  
print_elephant()  
print_giraffe()  
|
```

Ln: 7 Col: 0

Composite Functions

Combining functions, one at a time.

Composite Functions

```
def add_two(num):  
    return num + 2
```

```
def count_thrice():  
    return 1 + 1 + 1
```

Composite Functions

```
add_two(count_thrice())
```

```
def add_two(num):  
    return num + 2
```

```
def count_thrice():  
    return 1 + 1 + 1
```


Composite Functions

```
add_two(count_thrice())
```

```
def add_two(num):  
    return num + 2
```

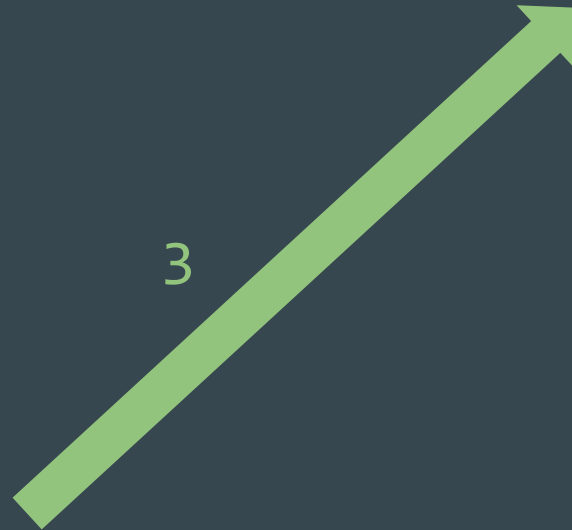
```
def count_thrice():  
    return 1 + 1 + 1
```

Composite Functions

```
def add_two(num):  
    return num + 2
```

```
def count_thrice():  
    return 1 + 1 + 1
```

add_two(count_thrice())



Composite Functions

```
def add_two(num):  
    return num + 2
```

```
def count_thrice():  
    return 1 + 1 + 1
```

add_two(count_thrice())



add_two(3)

Composite Functions

`add_two(count_thrice())`

```
def add_two(num):  
    return num + 2
```



`add_two(3)`

```
def count_thrice():  
    return 1 + 1 + 1
```

Composite Functions

`add_two(count_thrice())`

```
def add_two(num):  
    return num + 2
```



`return 3 + 2`

```
def count_thrice():  
    return 1 + 1 + 1
```

Composite Functions

`add_two(count_thrice())`

```
def add_two(num):  
    return num + 2
```

```
def count_thrice():  
    return 1 + 1 + 1
```

`return 3 + 2`



5

Composite Functions

`add_two(count_thrice())`

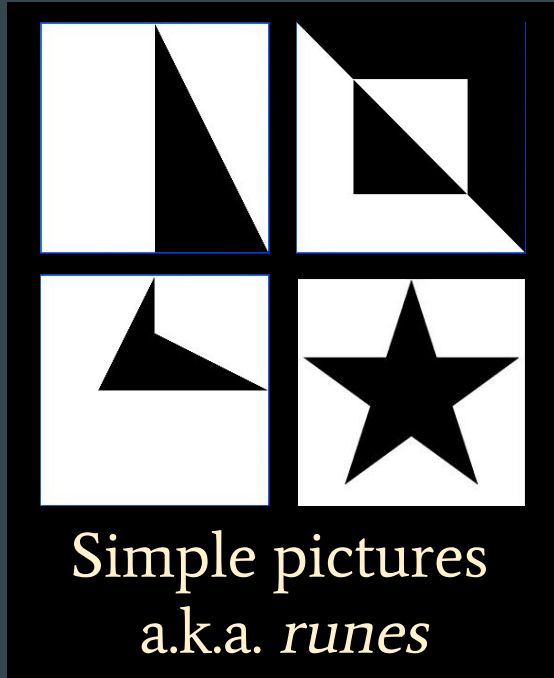
```
def add_two(num):  
    return num + 2
```

```
def count_thrice():  
    return 1 + 1 + 1
```



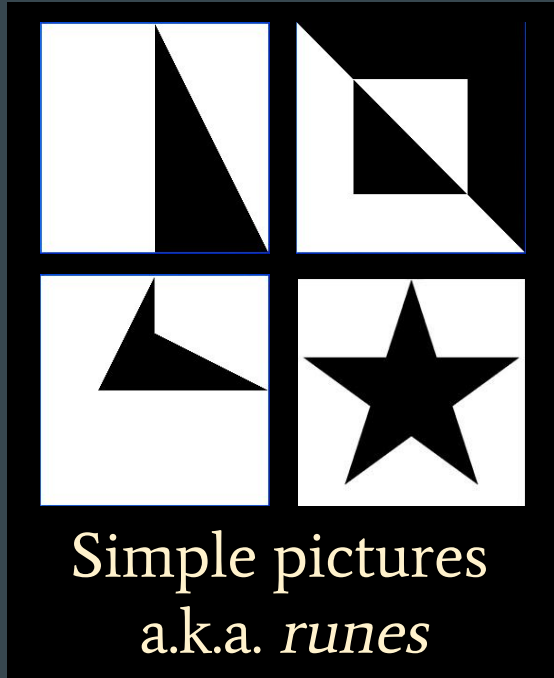
5

What is the Picture Language about?

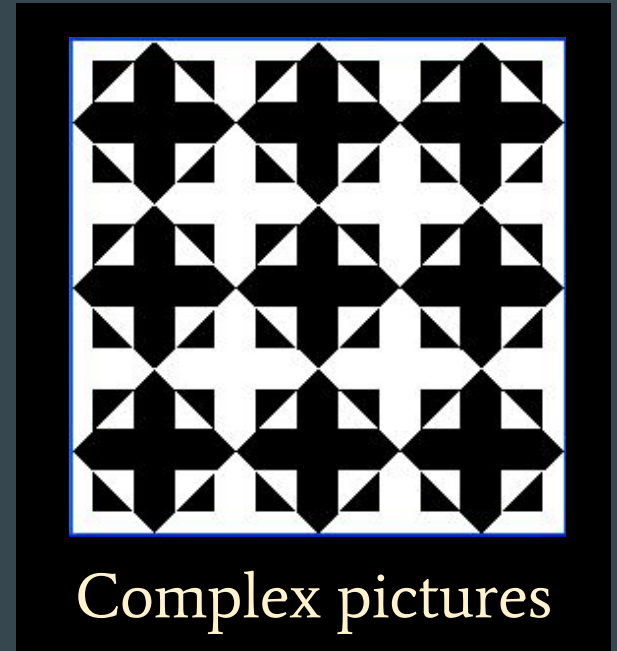
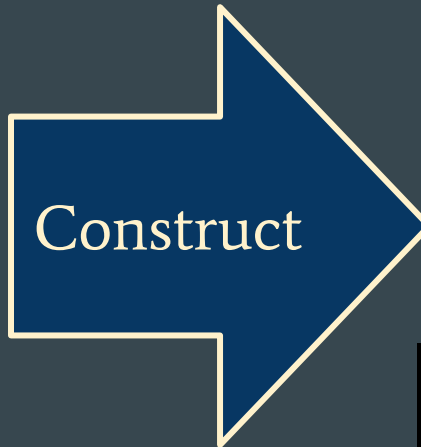


stack
quarter_turn_right
Simple functions

What is the Picture Language about?



stack
quarter_turn_right
Simple functions



stacker
make_cross
Complex functions

Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) Combining runes

Picture Language 1

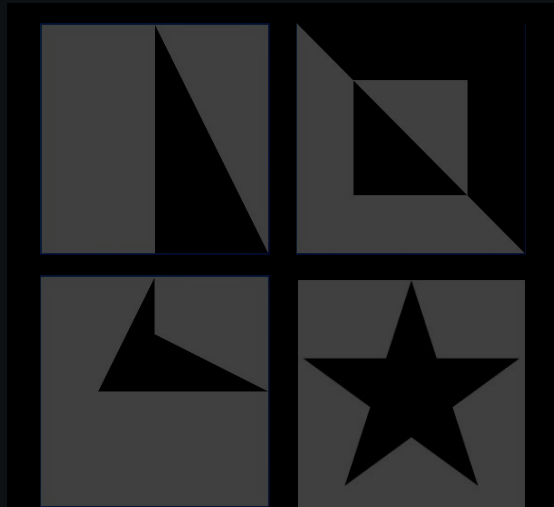
1) Importing libraries

2) Showing runes

3) Basic functions & composing them

4) Combining runes

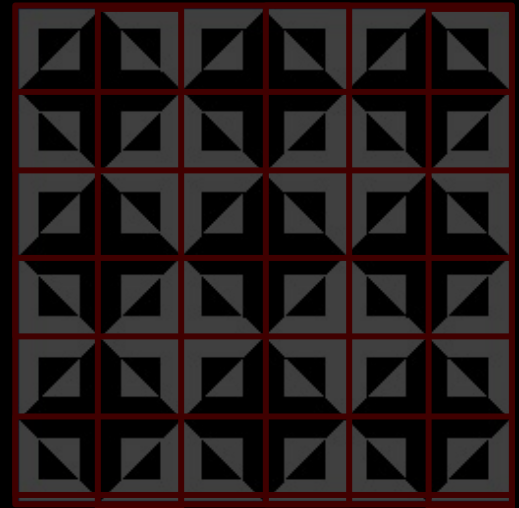
Overview



Simple pictures
a.k.a. *runes*



stack
quarter_turn_right
Simple functions

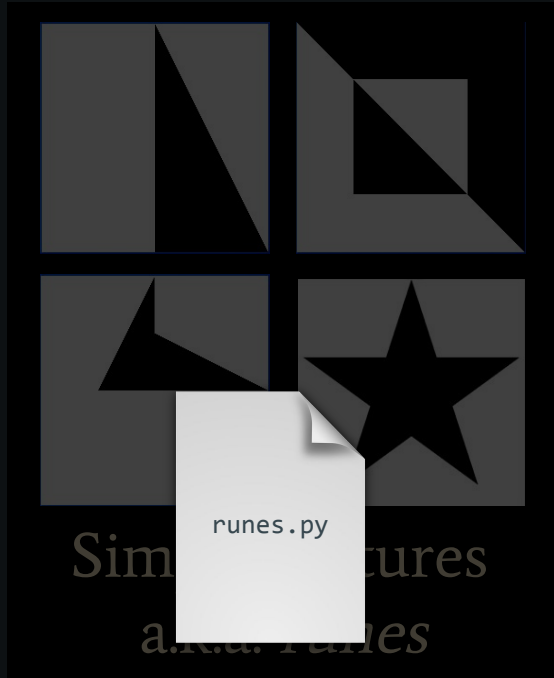


Complex pictures

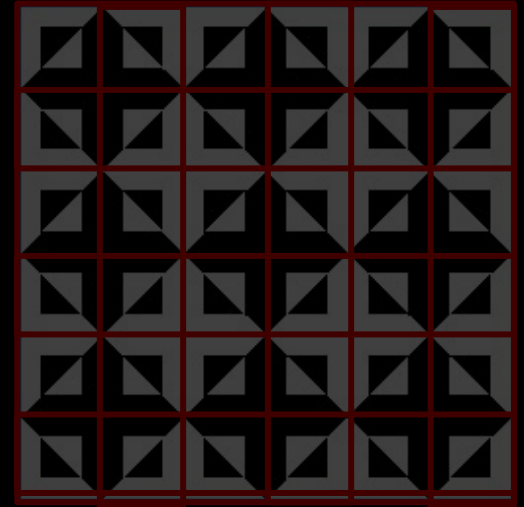


stacker
make_cross
Complex functions

The building blocks will be given to you in `runes.py`



`stack`
`quarter_turn_right`
Simple functions

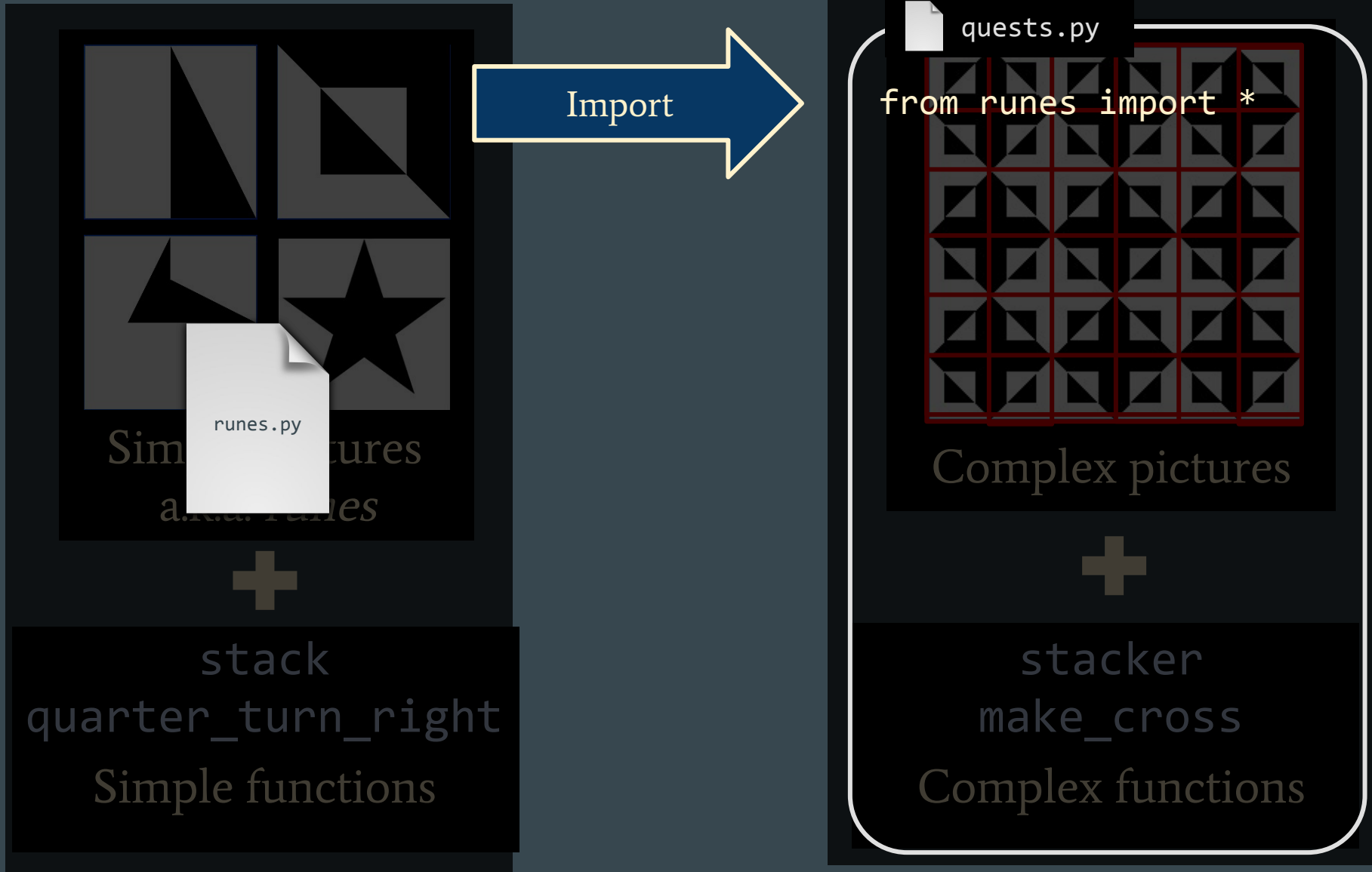


Complex pictures



`stacker`
`make_cross`
Complex functions

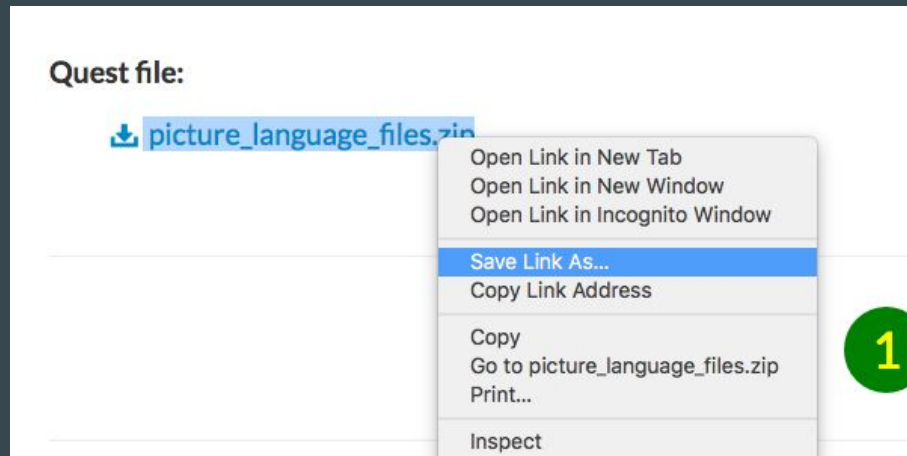
In quests.py, we import the code from runes.py



Downloading the files

You should already be in the “Picture Language 1” Basic Training

1. Right click on “picture_language_files.zip”
2. Click “Save Link As...” and save the file on your Desktop.



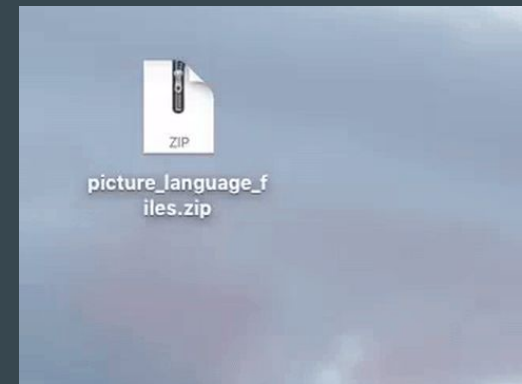
Unzipping the File

Windows

1. Go to your Desktop and look for “picture_language_files.zip”
2. If you are using an NUS computer,
 - a. Right-click on the file
 - b. Click ‘7-Zip’
 - c. Click ‘Extract to
“picture_language_files\”
3. If you are not using an NUS computer, ask your teacher for help if you cannot unzip the file.
4. Open the unzipped folder

Mac OS X

1. Go to your Desktop and look for “picture_language_files.zip”
2. Double-click on the file to unzip it
3. Open the unzipped folder



Blackboxing and importing

[illegible]

```
animals_say.py - /Users/briankoh/Documents/animals_say.py (...)
```

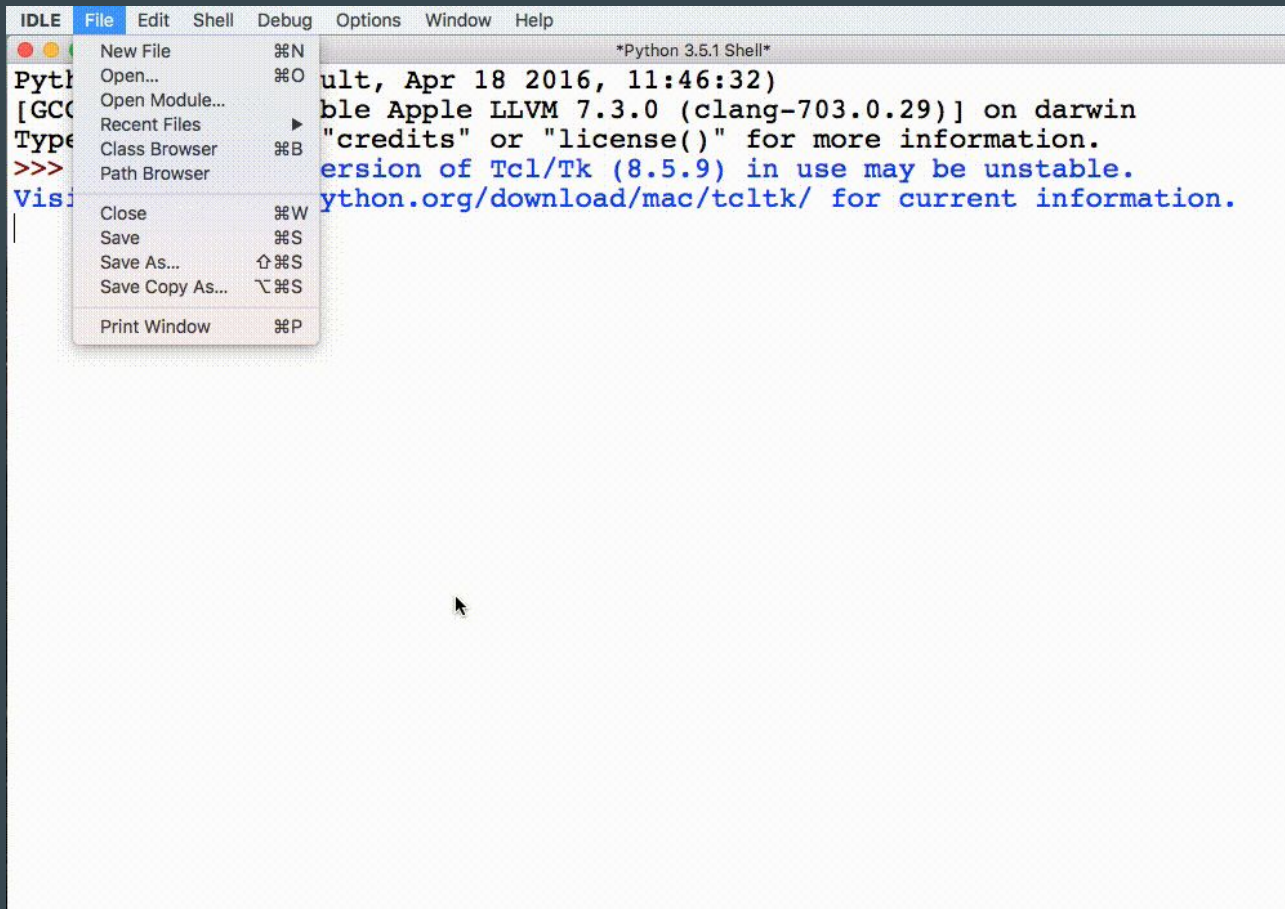
```
from animals import *  
from zoo import print_elephant  
from zoo import print_giraffe  
  
print_elephant()  
print_giraffe()  
|
```

Ln: 7 Col: 0

Importing Runes

from runes import *

and save it in the folder *picture_language_files* you just unzipped



Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) Combining runes

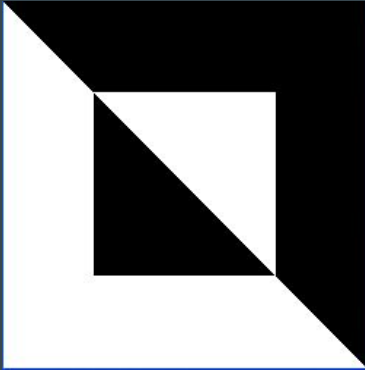
Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) Combining runes

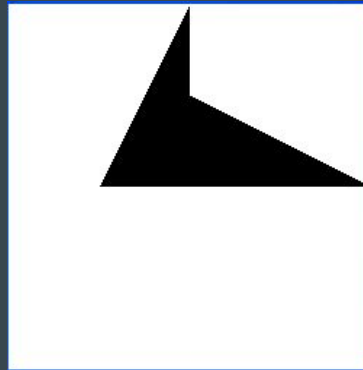
How are we going to get IDLE to display the pictures?

```
show( rune )
```

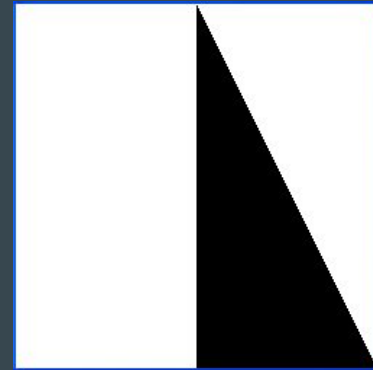
Some basic runes



rcross_bb



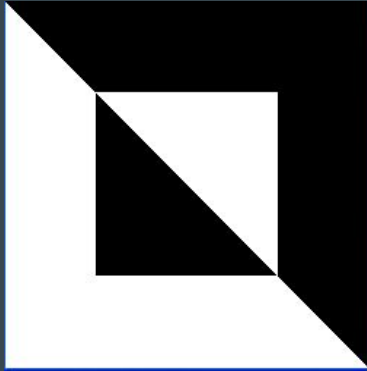
nova_bb



sail_bb

What if I want to show `rcross_bb`

```
show( rune )
```



`rcross_bb`



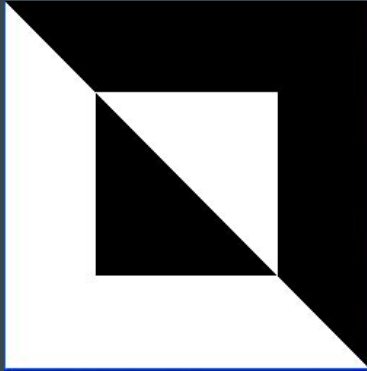
`nova_bb`



`sail_bb`

What if I want to show `rcross_bb`

```
show(rcross_bb)
```



`rcross_bb`



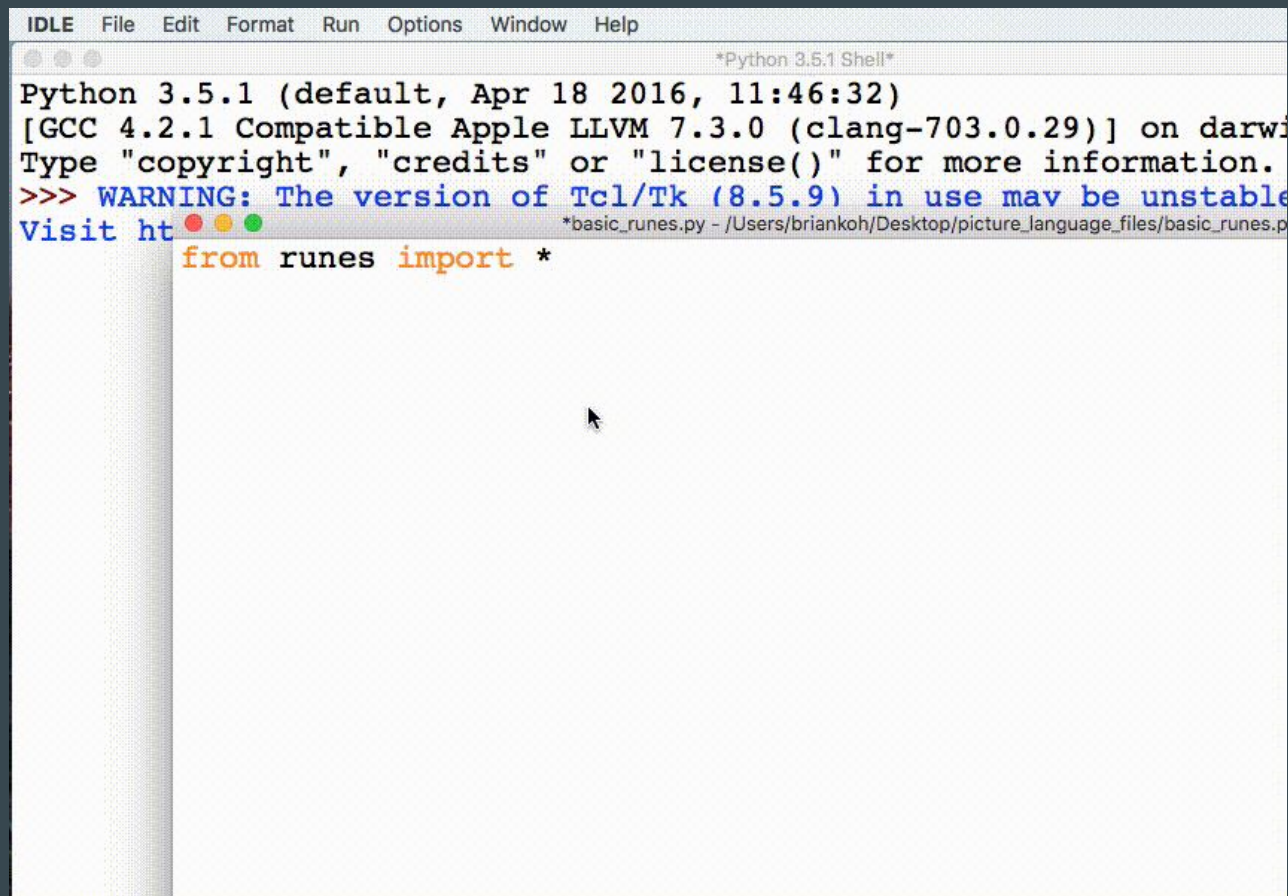
`nova_bb`



`sail_bb`

Showing Pictures

To show `rcross_bb`, type `show(rcross_bb)` and run the file by clicking “Run” > “Run Module”. The picture should appear in a separate window. You might be asked to save your file first.

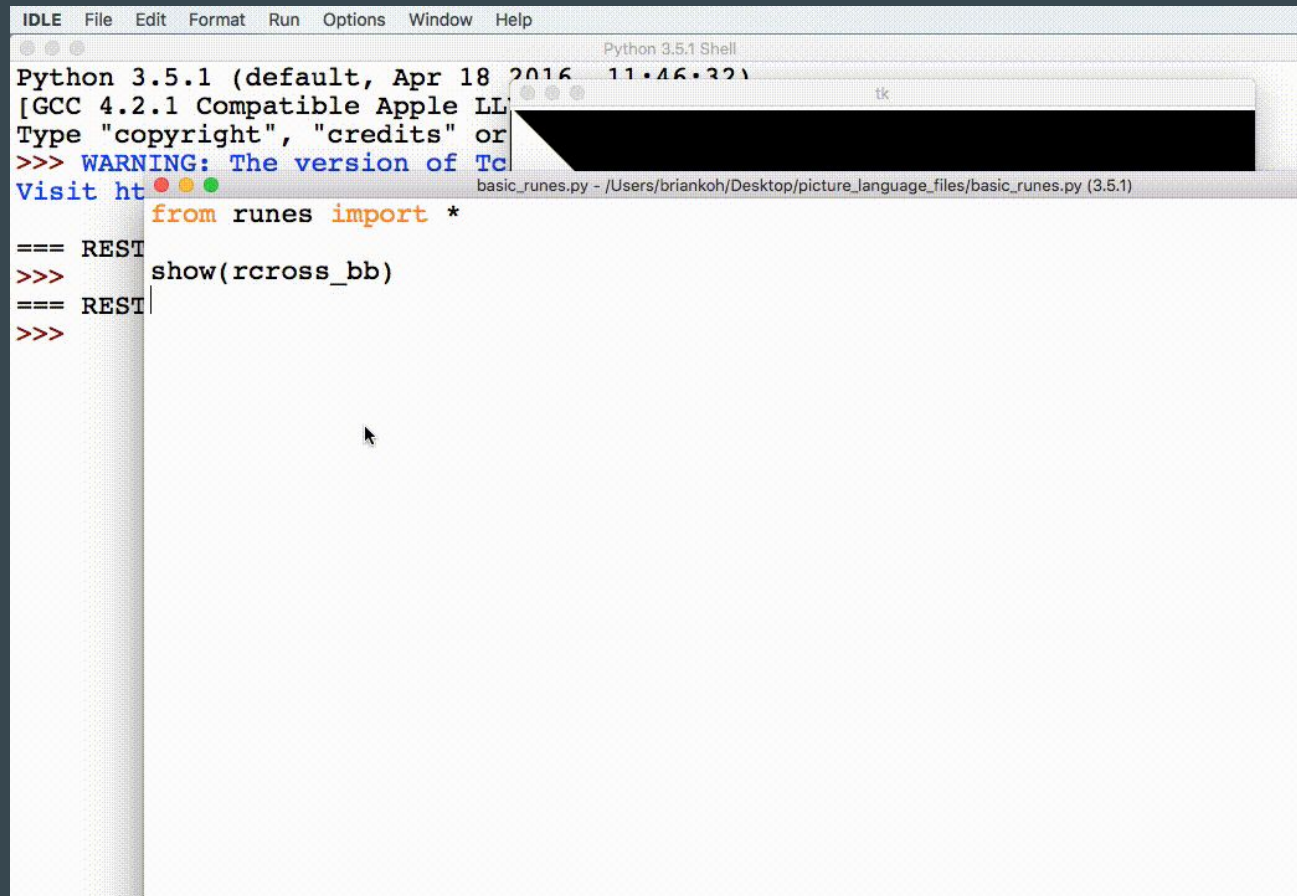


```
IDLE  File  Edit  Format  Run  Options  Window  Help
*Python 3.5.1 Shell*
Python 3.5.1 (default, Apr 18 2016, 11:46:32)
[GCC 4.2.1 Compatible Apple LLVM 7.3.0 (clang-703.0.29)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable
Visit http://www.python.org/download/releases/3.5.1/#tcltk
*basic_runes.py - /Users/briankoh/Desktop/picture_language_files/basic_runes.p
from runes import *
```

Showing Pictures

To show nova_bb,

- Comment out show(rcross_bb)
- Add show(nova_bb)
- Save the file
- Run the file

A screenshot of a Python 3.5.1 Shell window. The window has a menu bar with 'IDLE', 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. Below the menu bar, there's a status bar showing 'Python 3.5.1 Shell' and 'basic_runes.py - /Users/briankoh/Desktop/picture_language_files/basic_runes.py (3.5.1)'. The main area of the window displays the following text:

```
Python 3.5.1 (default, Apr 18 2016, 11:46:32)
[GCC 4.2.1 Compatible Apple LLVM 3.2.0] on darwin
Type "copyright", "credits" or "help()" to get more help.
>>> WARNING: The version of Tk is 8.5.5, which is not supported by the
Visit http://www.python.org/download/releases/3.5.1/ for more details.
>>> from runes import *
=== REST
>>> show(rcross_bb)
=== REST
>>>
```

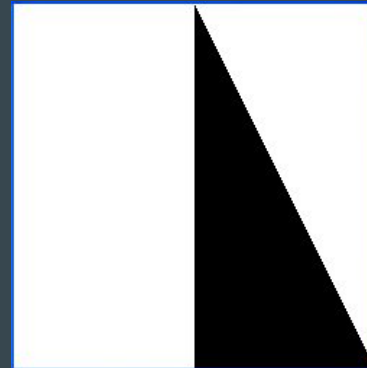
Can you show `sail_bb`?



`rcross_bb`

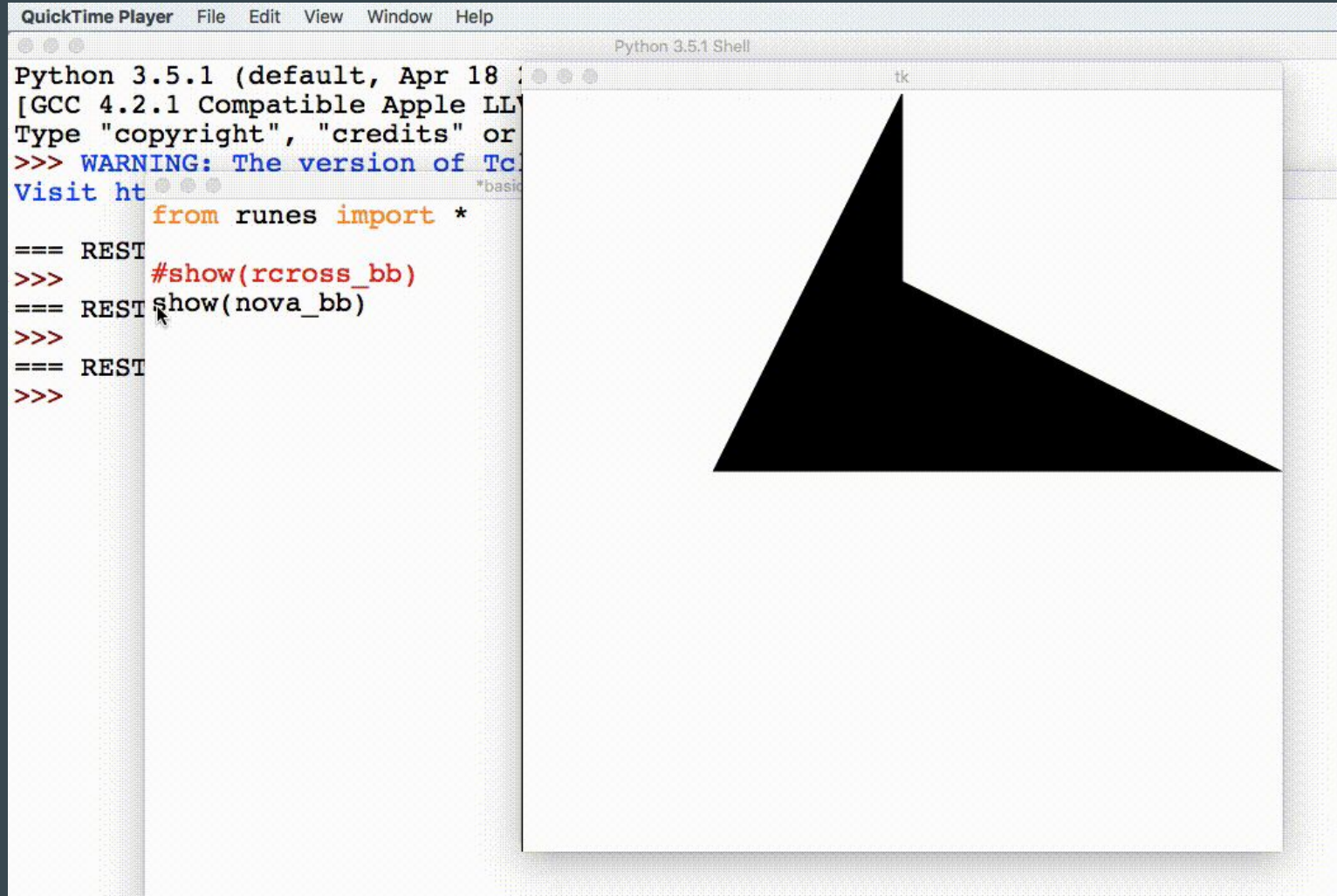


`nova_bb`

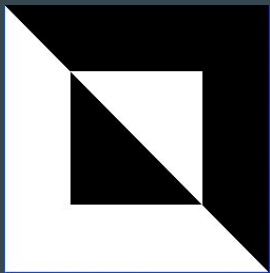


`sail_bb`

Can you show sail_bb?



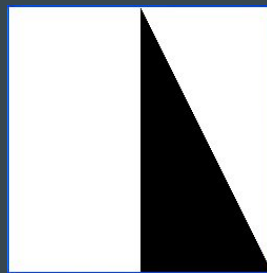
Runes Building Blocks



rcross_bb



heart_bb



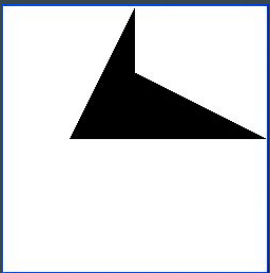
sail_bb



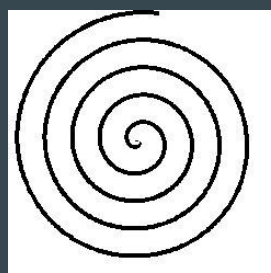
corner_bb



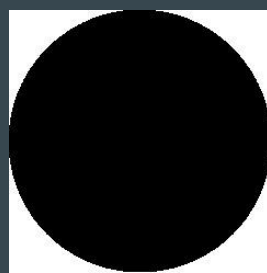
black_bb



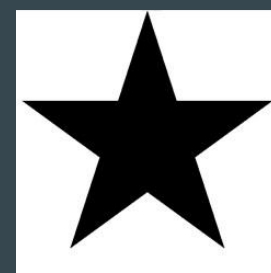
nova_bb



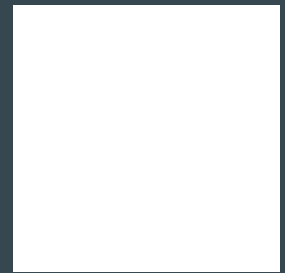
ribbon_bb



circle_bb



pentagram_bb



blank_bb

Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) Combining runes

Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them**
- 4) Combining runes

Importing

```
# Setup~
import graphics~
import math~
import time~
import PyGif~
~
# Constants~
viewport_size = 600~
spread = 20~
active_hollusion = None~
lastframe = None~
~
Posn = graphics.Posn~
Rgb = graphics.Rgb~
draw_solid_polygon~
~
graphics.init(view~
vp = graphics.oper~
lp = graphics.oper~
rp = graphics.oper~
~
def clear_all():~
    global active_hollusion~
    global vp, lp, rp~
    if(active_hollusion != None):~
        active_hollusion("kill")~
        active_hollusion = None~
    graphics.clear_viewport(vp)~
    graphics.clear_viewport(lp)~
    graphics.clear_viewport(rp)~
~
class Frame:~
    def __init__(self, p0, p1, p2, z1, z2):~
```

runes.py

Import

Import

Import

Import

Import

quests.py

```
from runes import *
```

show

rcross_bb

sail_bb

nova_bb

...

circle_bb

Importing

```
# Setup~
import graphics~
import math~
import time~
import PyGif~
~
# Constants~
viewport_size = 600~
spread = 20~
active_hollusion = None~
lastframe = None~
~
Posn = graphics.Posn~
Rgb = graphics.Rgb~
draw_solid_polygon~
~
graphics.init(view~
vp = graphics.oper~
lp = graphics.oper~
rp = graphics.oper~
~
def clear_all():~
    global active_hollusion~
    global vp, lp, rp~
    if(active_hollusion != None):~
        active_hollusion("kill")~
        active_hollusion = None~
    graphics.clear_viewport(vp)~
    graphics.clear_viewport(lp)~
    graphics.clear_viewport(rp)~
~
class Frame:~
    def __init__(self, p0, p1, p2, z1, z2):~
```

runes.py

Import

show

Import

rcross_bb

Import

sail_bb

Import

nova_bb

...

Import

circle_bb

Import

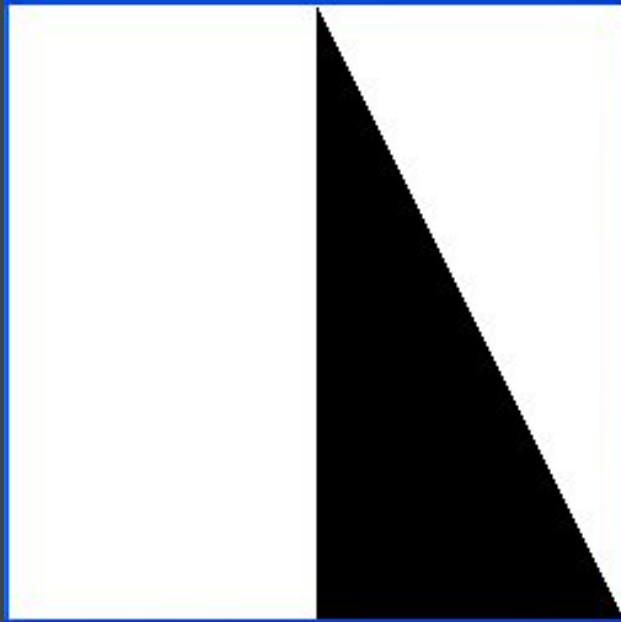
quarter_turn_right



quests.py

*from runes import **

```
quarter_turn_right(rune)
```



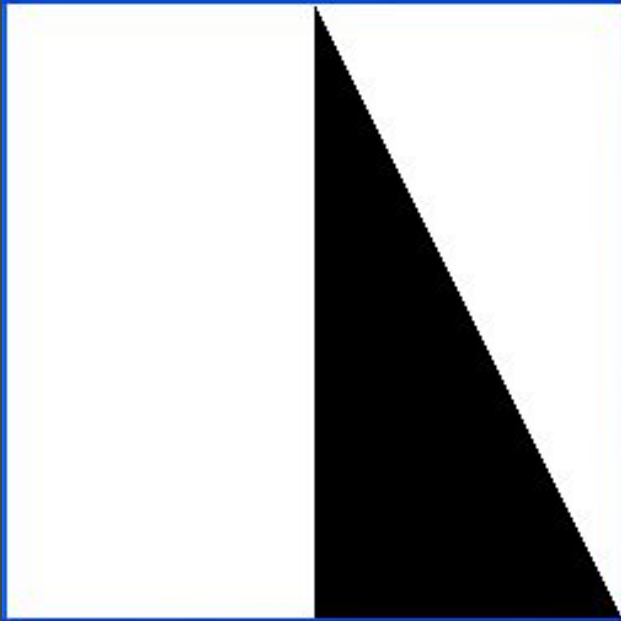
```
show(sail_bb)
```

`quarter_turn_right(rune)`

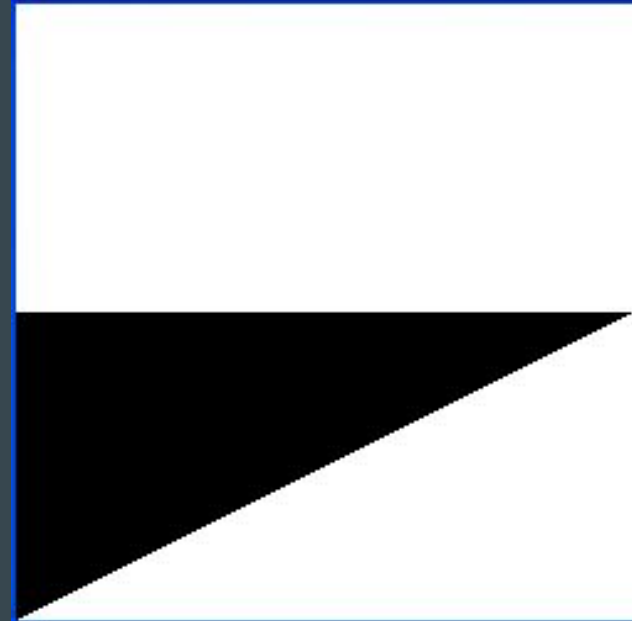


`show(sail_bb)`

quarter_turn_right(rune)

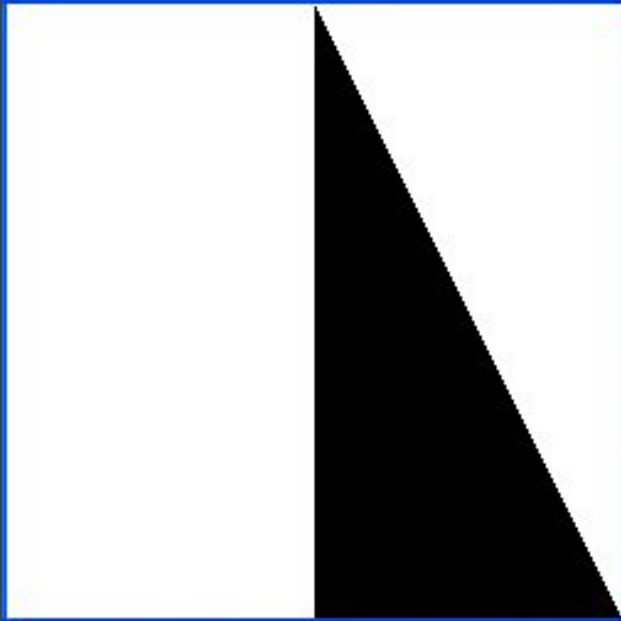


show(sail_bb)

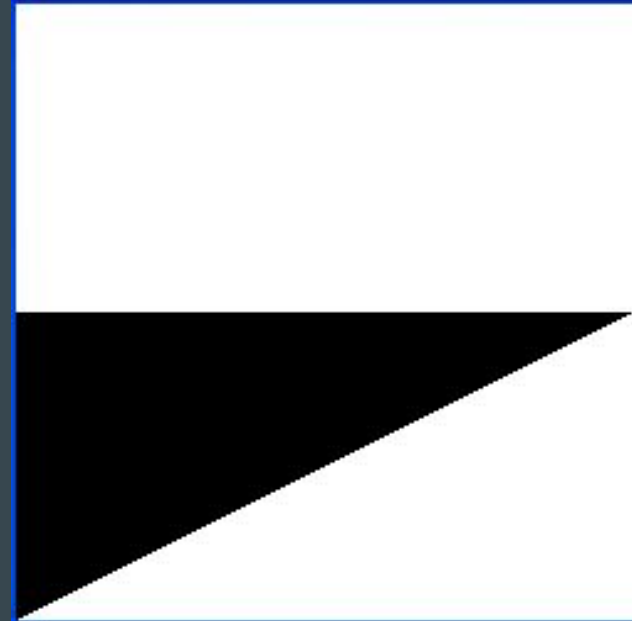


quarter_turn_right(sail_bb)

quarter_turn_right()

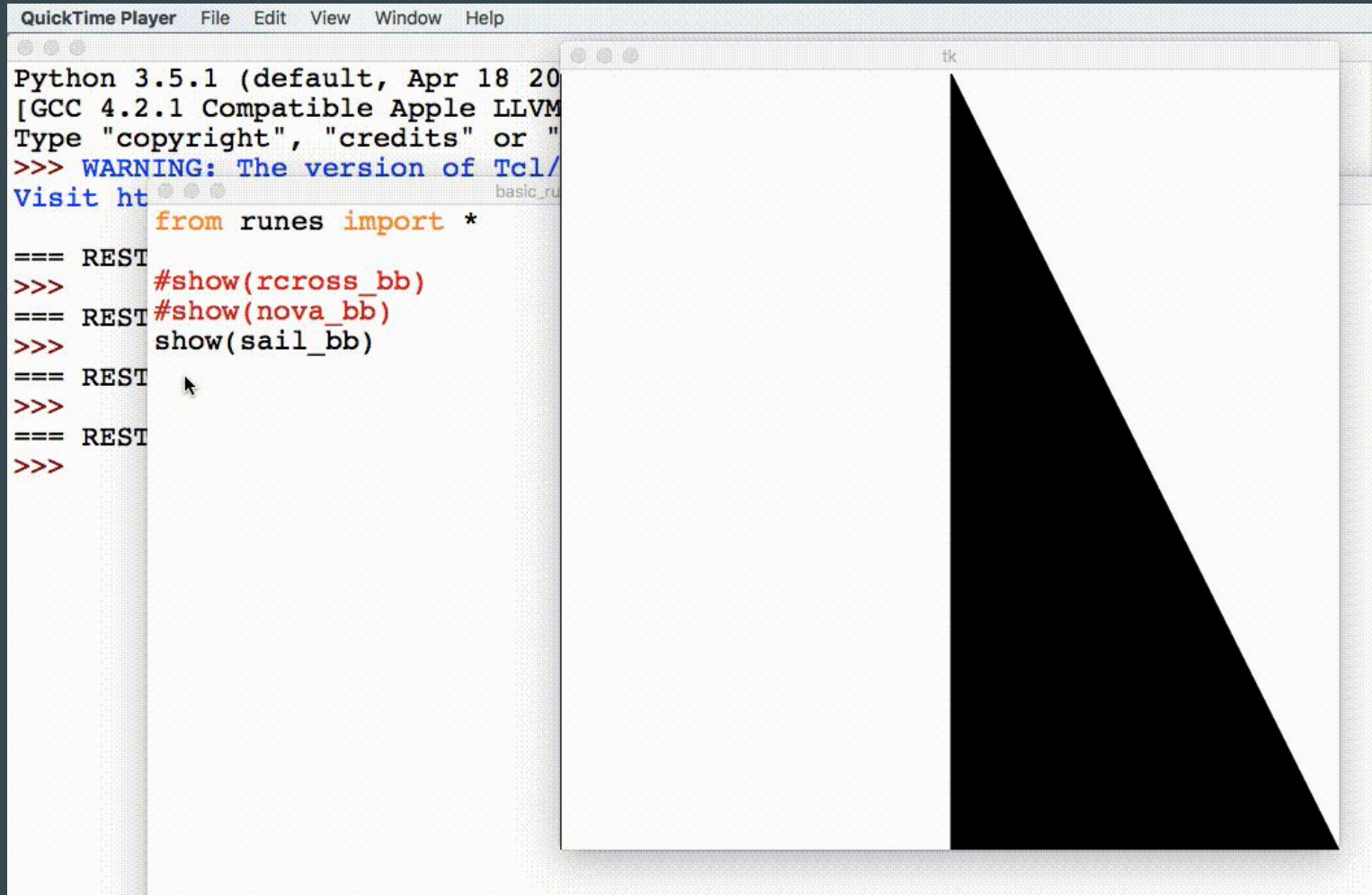


`show(sail_bb)`

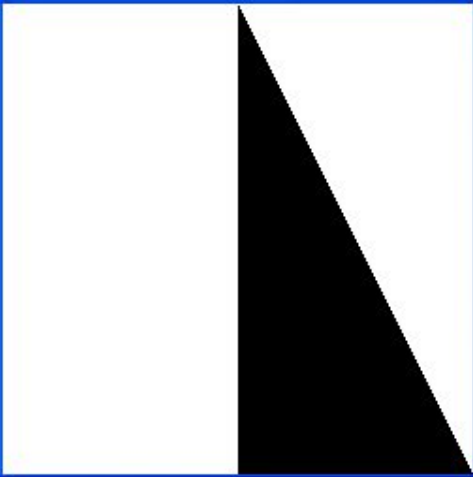


`show(quarter_turn_right(sail_bb))`

Quarter Turn Right



Turn Upside Down



`show(sail_bb)`

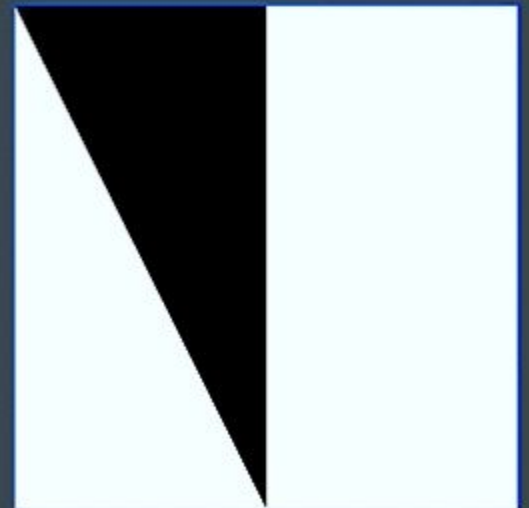


Turn Upside Down



```
show(sail_bb)
```

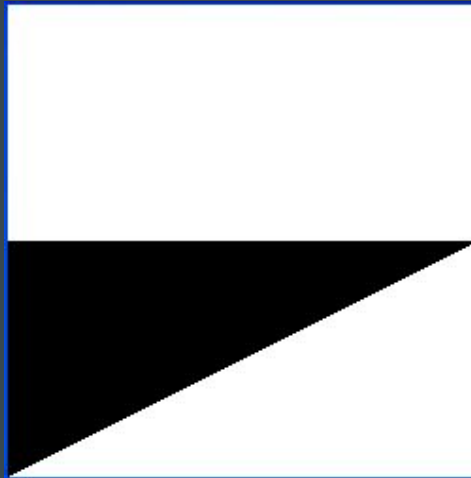
```
show(  
    quarter_turn_right(  
        sail_bb))
```



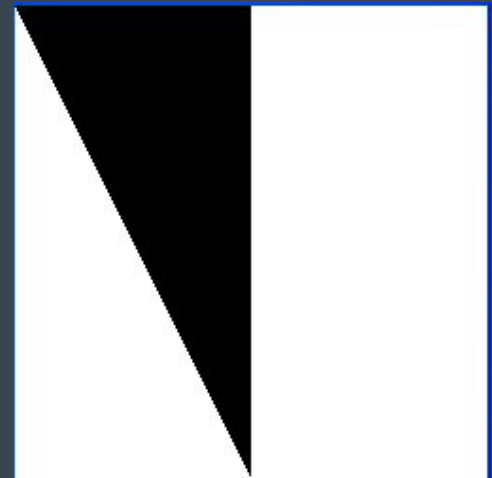
Turn Upside Down



```
show(sail_bb)
```

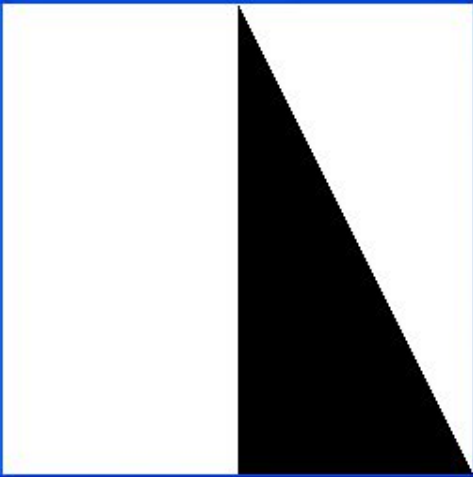


```
show(  
  quarter_turn_right(  
    sail_bb))
```



```
show(  
  quarter_turn_right(  
    quarter_turn_right(  
      sail_bb)))
```

Turn Upside Down



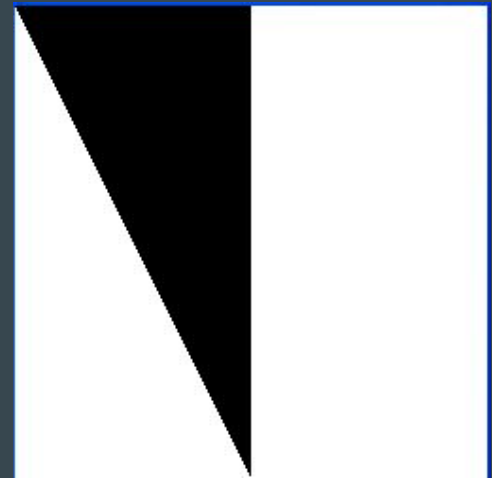
```
show(sail_bb)
```



```
show(  
  quarter_turn_right(  
    quarter_turn_right(  
      sail_bb)))
```

Turn Upside Down

```
from runes import *
```



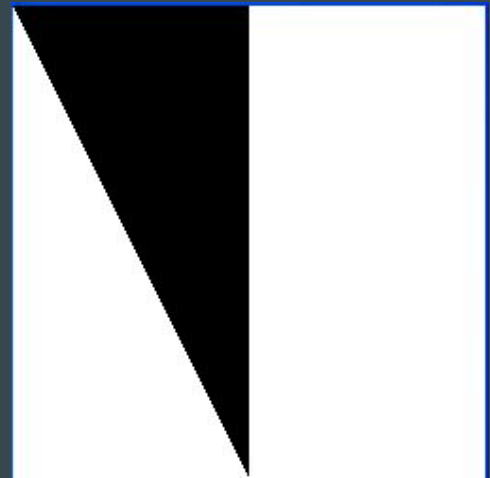
```
show(  
    quarter_turn_right(  
        quarter_turn_right(  
            sail_bb)))
```

Turn Upside Down

```
from runes import *  
def turn_upside_down(  

```

How many parameters?

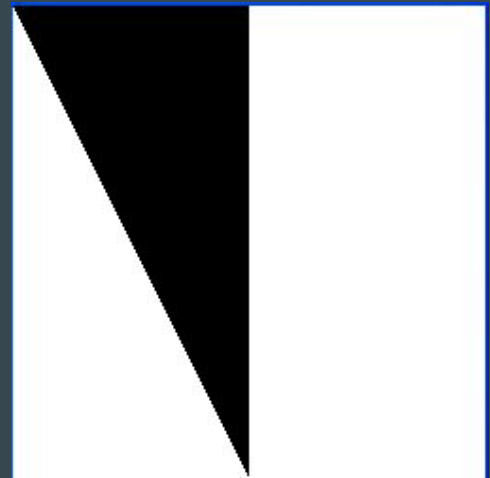


```
show(  
    quarter_turn_right(  
        quarter_turn_right(  
            sail_bb)))
```

Turn Upside Down

```
from runes import *  
  
def turn_upside_down(  
    pic:  
    ...  
):
```

How many parameters?
Just one. Let's call it `pic`.

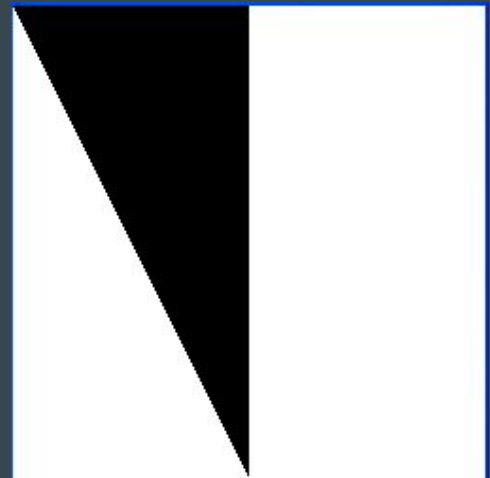


```
show(  
    quarter_turn_right(  
        quarter_turn_right(  
            sail_bb)))
```

Turn Upside Down

```
from runes import *  
  
def turn_upside_down(pic):
```

How many parameters?
Just one. Let's call it `pic`.

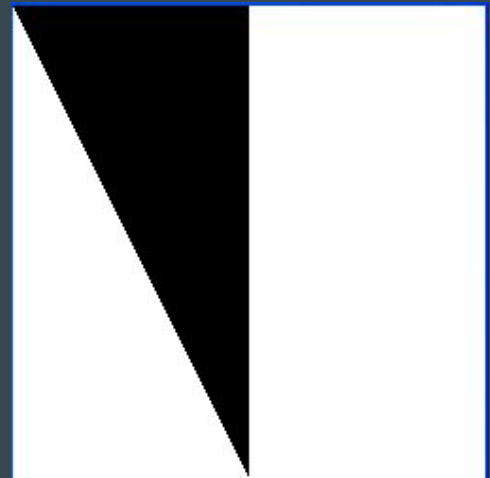


```
show(  
    quarter_turn_right(  
        quarter_turn_right(  
            sail_bb))
```

Turn Upside Down

```
from runes import *  
  
def turn_upside_down(pic):
```

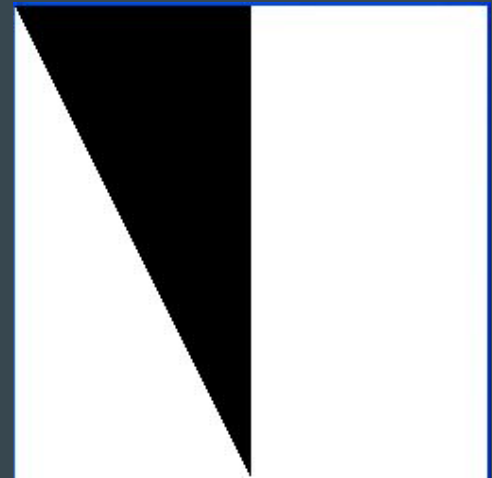
How do we turn `pic`
upside down?



```
show(  
    quarter_turn_right(  
        quarter_turn_right(  
            sail_bb)))
```

Turn Upside Down

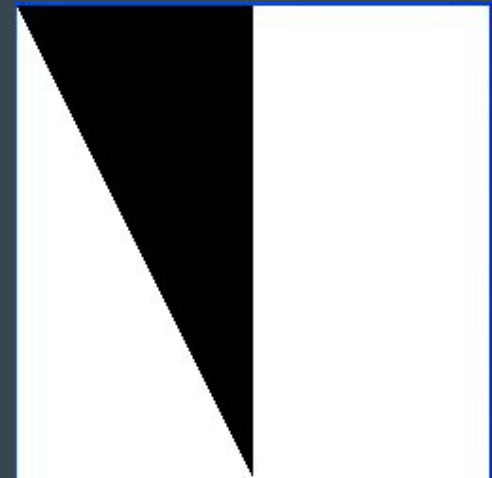
```
from runes import *  
  
def turn_upside_down(pic):
```



```
show(  
    quarter_turn_right(  
        quarter_turn_right(  
            sail_bb)))
```


Turn Upside Down

```
from runes import *  
  
def turn_upside_down(pic):  
    return quarter_turn_right(  
        quarter_turn_right(  
            sail_bb)))
```

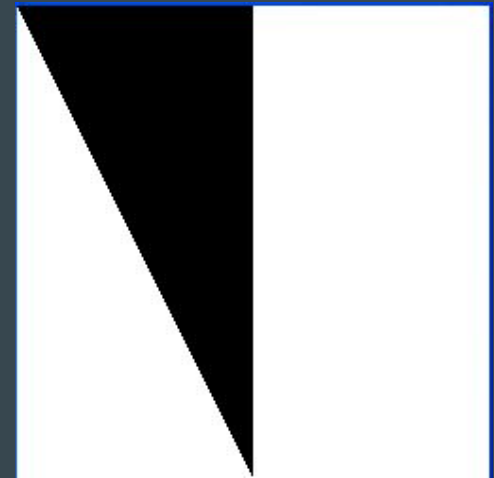


```
show(  
    quarter_turn_right(  
        quarter_turn_right(  
            sail_bb)))
```



Turn Upside Down

```
from runes import *  
  
def turn_upside_down(pic):  
    return quarter_turn_right(  
        quarter_turn_right(  
            pic)))
```



```
show(  
    quarter_turn_right(  
        quarter_turn_right(  
            sail_bb)))
```

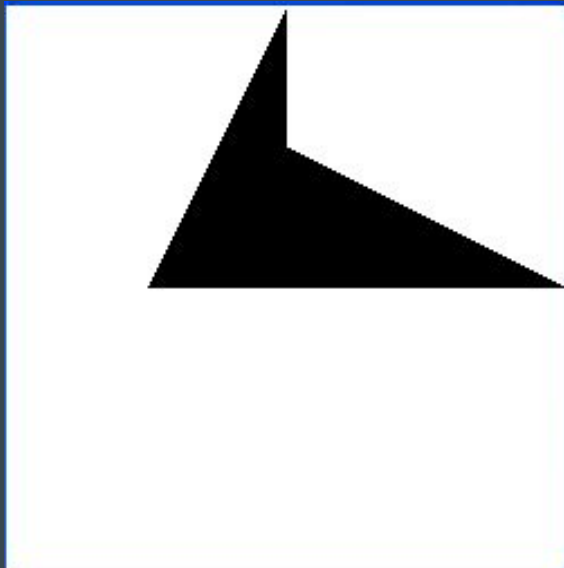


Turn Upside Down

```
from runes import *  
  
def turn_upside_down(pic):  
    return quarter_turn_right(quarter_turn_right(pic))  
  
show(turn_upside_down(sail_bb))
```

Turn Upside Down

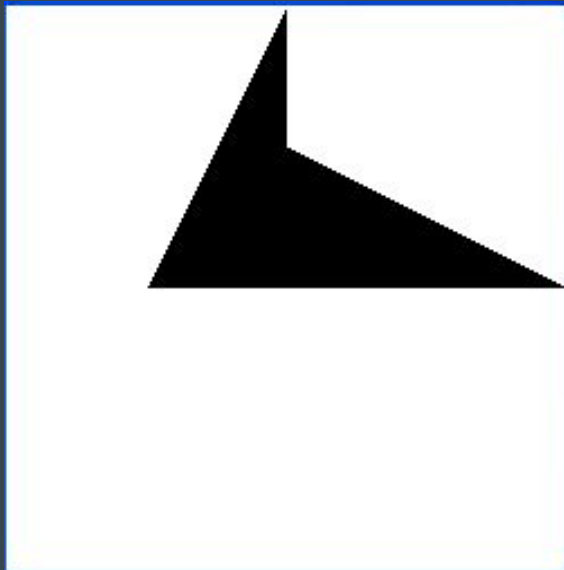
```
from runes import *  
  
def turn_upside_down(pic):  
    return quarter_turn_right(quarter_turn_right(pic))  
  
show(turn_upside_down(sail_bb))
```



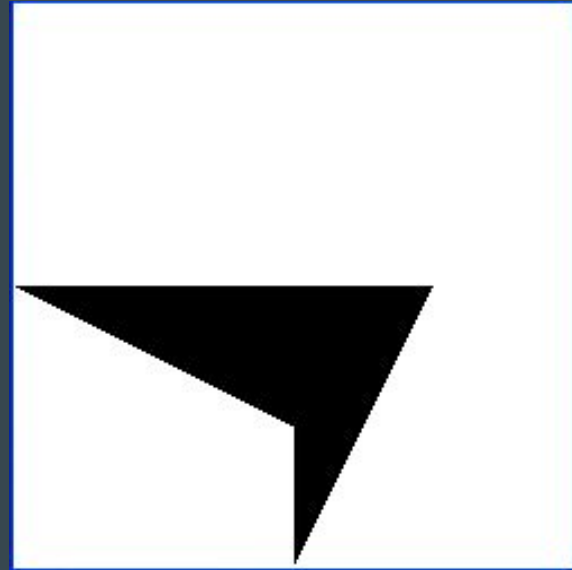
nova_bb

Turn Upside Down

```
from runes import *  
  
def turn_upside_down(pic):  
    return quarter_turn_right(quarter_turn_right(pic))  
  
show(turn_upside_down(nova_bb))
```



nova_bb



Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them**
- 4) Combining runes

How to complete quests
questions that involves
runes

How to complete quests

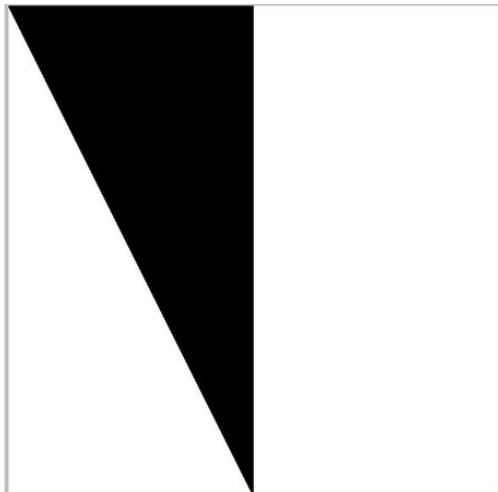
Turn Upside Down

Description

Write a method `turn_upside_down` that takes in a rune and returns a rune is that is the input rune turned upside down.

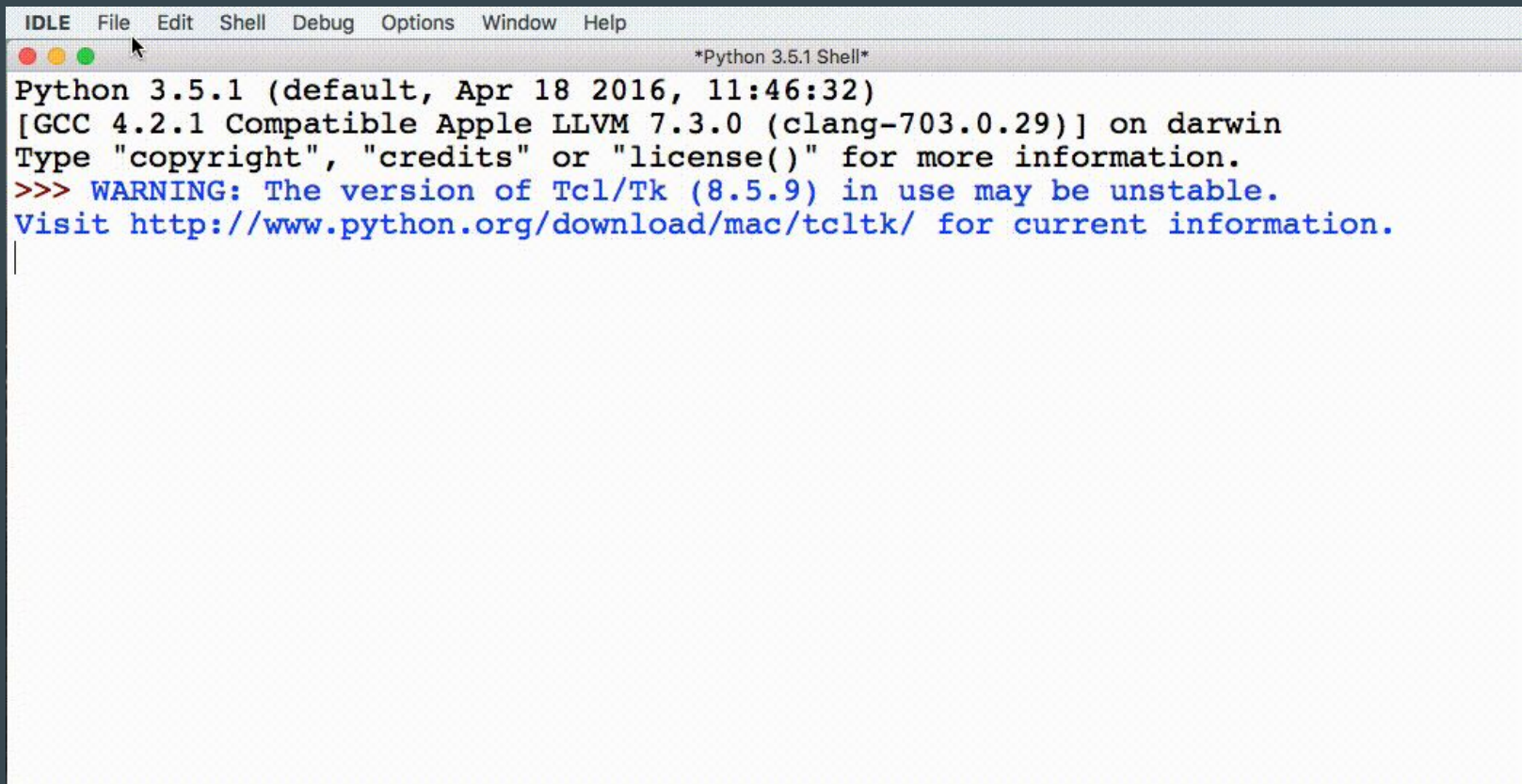
Sample Execution:

```
show(turn_upside_down(sail_bb))
```



Open Template

Templates for the test code can be found in `quests.py`. Open it.

A screenshot of a Python 3.5.1 Shell window. The window has a title bar with standard macOS window controls (red, yellow, green buttons) and a title text '*Python 3.5.1 Shell*'. The menu bar at the top includes 'IDLE', 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area displays the following output:

```
Python 3.5.1 (default, Apr 18 2016, 11:46:32)
[GCC 4.2.1 Compatible Apple LLVM 7.3.0 (clang-703.0.29)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.
```

A cursor is visible at the end of the last line of output.

Type code in

Attempt to complete the function for the quest. You don't have to get the code right the first time.

```
IDLE  File  Edit  Format  Run  Options  Window  Help
*Python 3.5.1 Shell*
Python 3.5.1 (default, Apr 18 2016, 11:46:32)
[GCC 4.2.1 Compatible Apple LLVM 7.3.0 (clang-703.0.29)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/releases/3.5.1/ for more details.
*quests.py - /Users/briankoh/Desktop/picture_language_files/quests.py (3.5.1)*
from runes import *

#####
# Turn Upside Down
#####

def turn_upside_down(pic):
    return # fill your code in

#show(turn_upside_down(sail_bb))
#show(turn_upside_down(nova_bb))

#####
```

Test your code

```
IDLE File Edit Format Run Options Window Help
Python 3.5.1 Shell
Python 3.5.1 (default, Apr 18 2016, 11:46:32)
[GCC 4.2.1 Compatible Apple LLVM 7.3.0 (clang-703.0.29)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit Python.org for more details.
*quests.py - /Users/briankoh/Desktop/picture_language_files/quests.py (3.5.1)*

from runes import *

=====
>>> #####
# Turn Upside Down
#####

def turn_upside_down(pic):
    return quarter_turn_right(quarter_turn_right(pic))

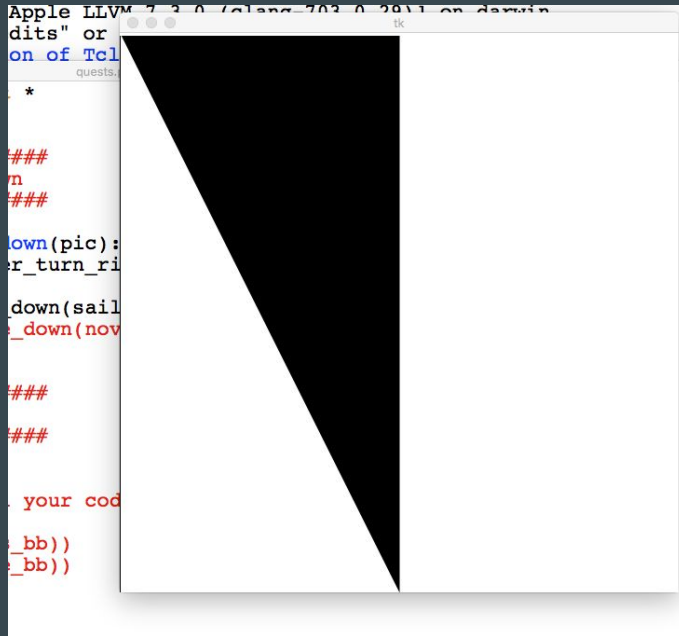
#show(turn_upside_down(sail_bb))
#show(turn_upside_down(nova_bb))

#####
# Twin
#####

def twin(pic):
    return # fill your code in

#show(twin(rcross_bb))
#show(twin(circle_bb))
```

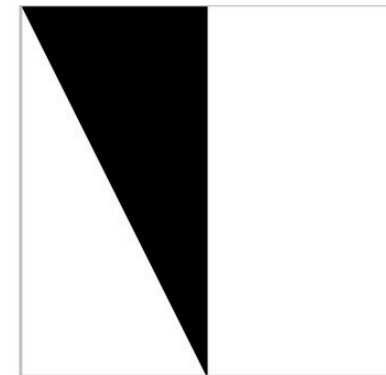
Compare with Sample Execution



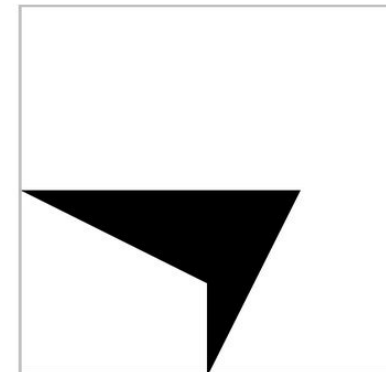
Write a method `turn_upside_down` that takes in a rune and returns

Sample Execution:

```
show(turn_upside_down(sail_bb))
```



```
show(turn_upside_down(nova_bb))
```



Check second example

```
IDLE  File  Edit  Format  Run  Options  Window  Help
Python 3.5.1 Shell
Python 3.5.1 (default, Apr 18 2016, 11:46:32)
[GCC 4.2.1 Compatible Apple LLVM 7.3.0 (clang-702.0.28) on darwin
Type "copyright", "credits" or
>>> WARNING: The version of Tcl
Visit
quests.py - /Users/briankoh/Desktop/picture_language_files/quests.py (3.5.1)

from runes import *

=====
>>>
=====
>>> #####
# Turn Upside Down
#####

def turn_upside_down(pic):
    return quarter_turn_right(quarter_turn_right(pic))

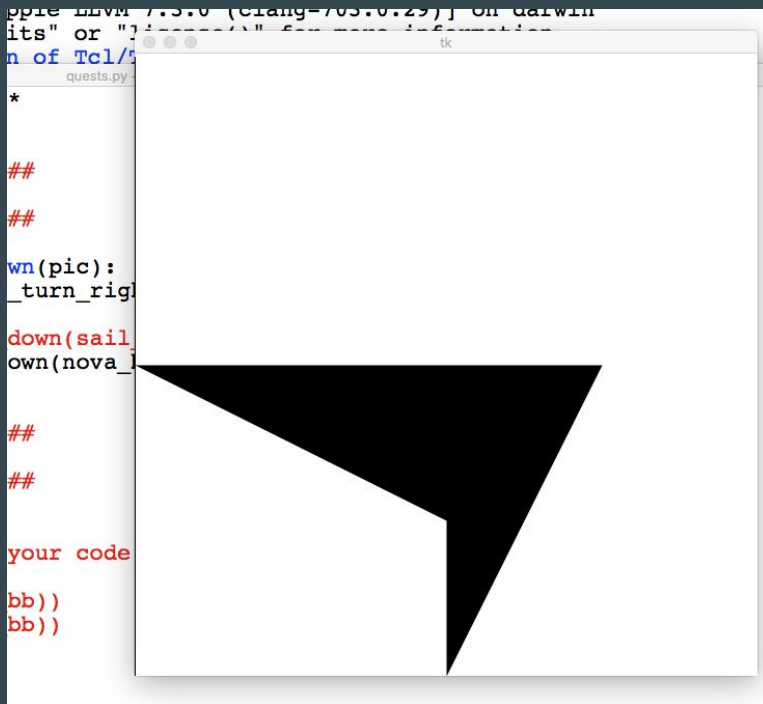
show(turn_upside_down(sail_bb))
#show(turn_upside_down(nova_bb))

#####
# Twin
#####

def twin(pic):
    return # fill your code in

#show(twin(rcross_bb))
#show(twin(circle_bb))
```

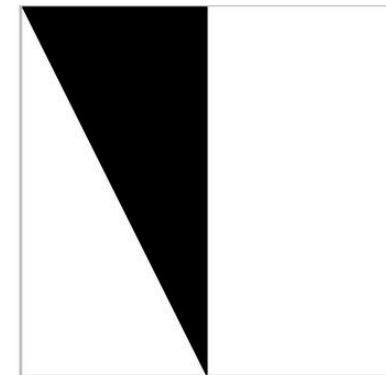
Compare with Sample Execution



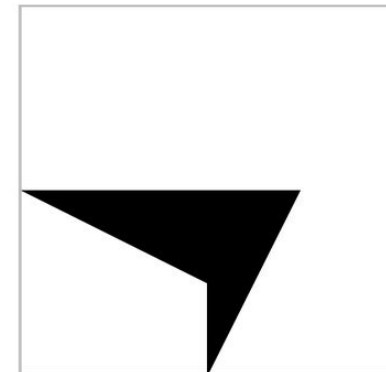
Write a method `turn_upside_down` that takes in a rune and returns

Sample Execution:

```
show(turn_upside_down(sail_bb))
```



```
show(turn_upside_down(nova_bb))
```



Submitting code on Coursemology

Once you are sure of your code, copy it.

```
*quests.py - /Users/briankoh/Desktop/picture_language_files/quests.py (3.5.1)*  
from runes import *  
  
#####  
# Turn Upside Down  
#####  
  
def turn_upside_down(pic):  
    return quarter_turn_right(quarter_turn_right(pic))  
  
#show(turn_upside_down(sail_bb))  
show(turn_upside_down(nova_bb))
```

Submitting code on Coursemology

Paste it and submit it on Coursemology. The quest is completed!

Public Tests

The following expressions will be executed.

Expression	Expected Output
turn_upside_down(sail_bb)	answer

Your Answer

```
1 def turn_upside_down(rune):  
2     pass # replace the function body with your code
```

Evaluation Result

Click Run to see your evaluation result.

Run (ALT+R)

Continue

How to complete quests questions that involves runes

1. Test out code in template file `quest.py`
2. Check generated rune against image in Quest
3. If it matches, copy code in Coursemology.
4. Press Run. If it works, you can continue!

Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) **Combining runes**

Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) Combining runes
 - top and bottom → left and right
 - complex patterns

Functions so far

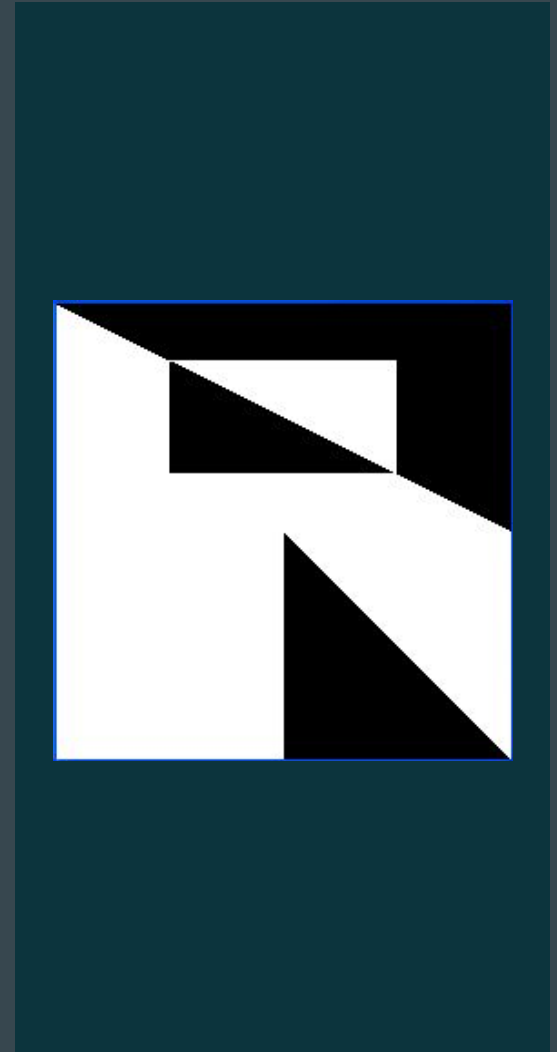
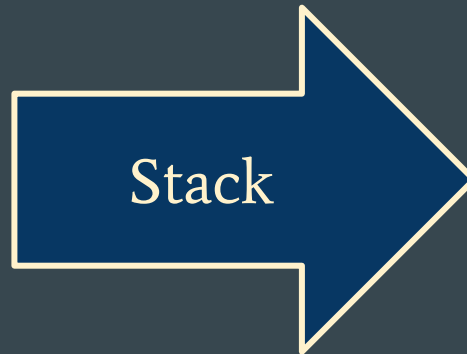
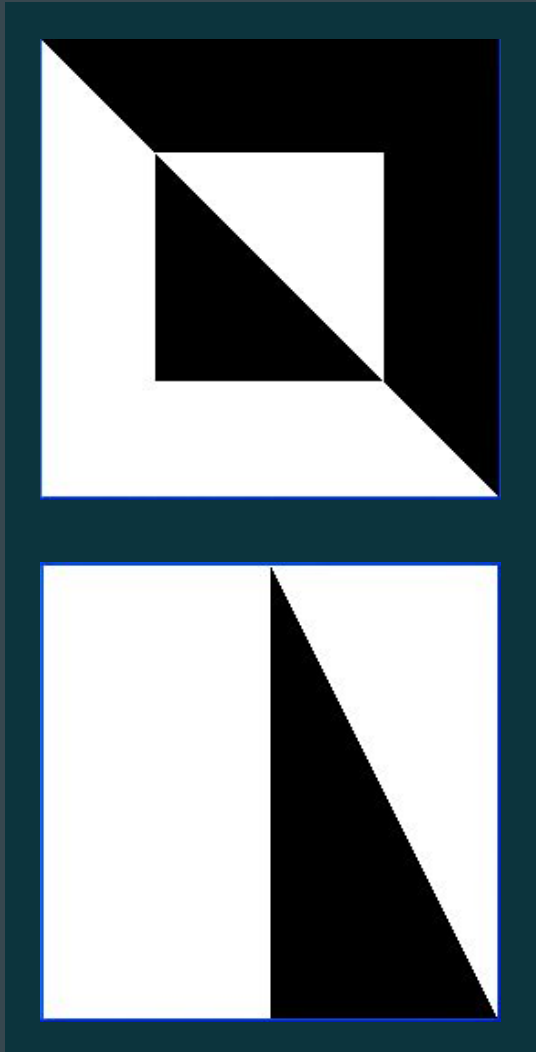
`quarter_turn_right()`
`quarter_turn_left()`

Functions so far

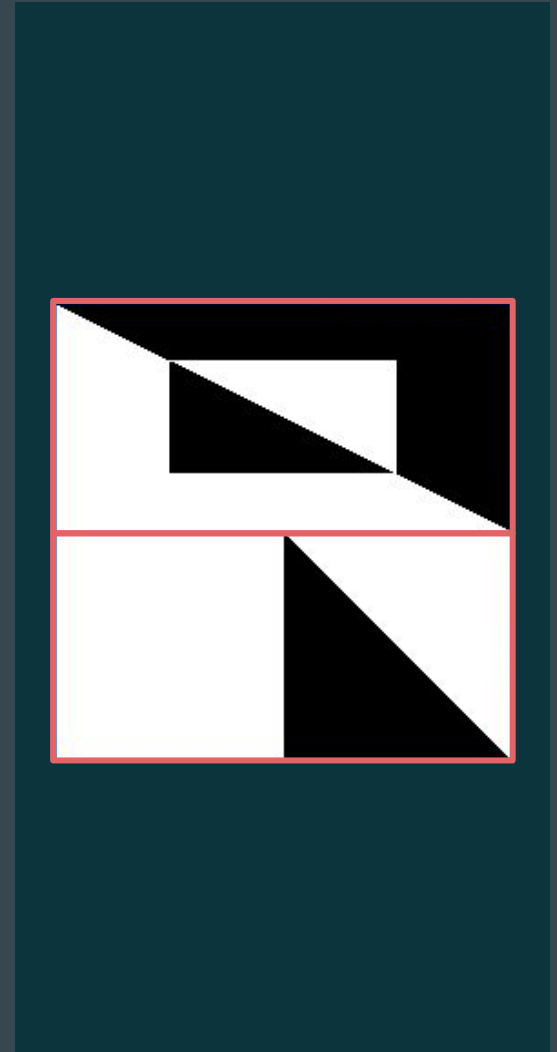
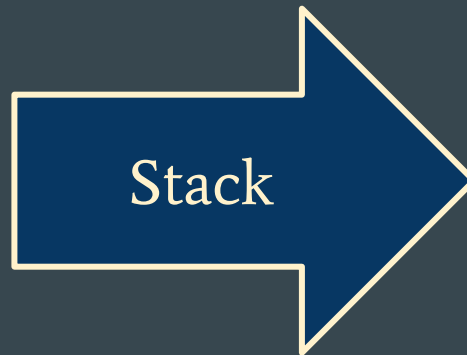
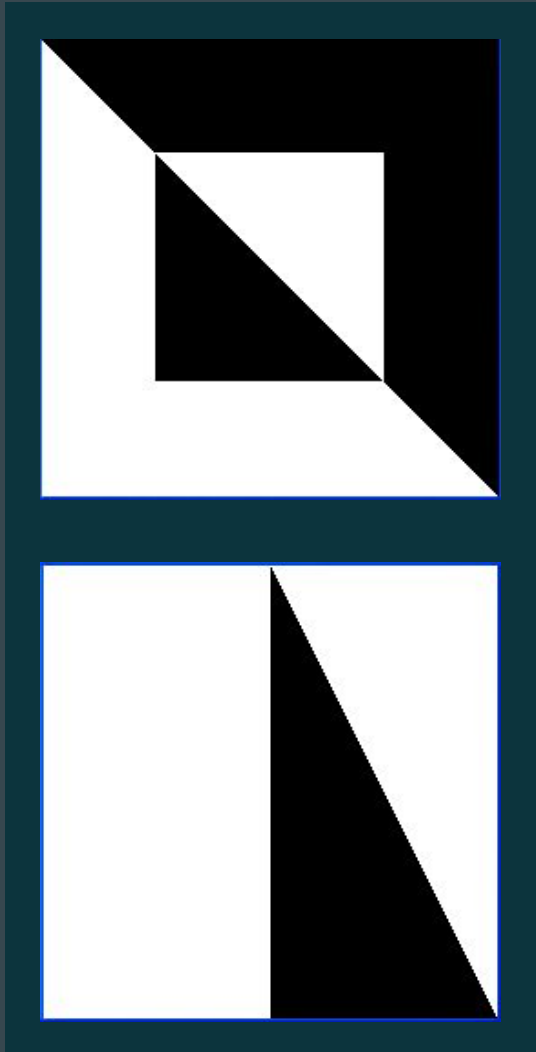
quarter_turn_right()
quarter_turn_left()

turn_upside_down()

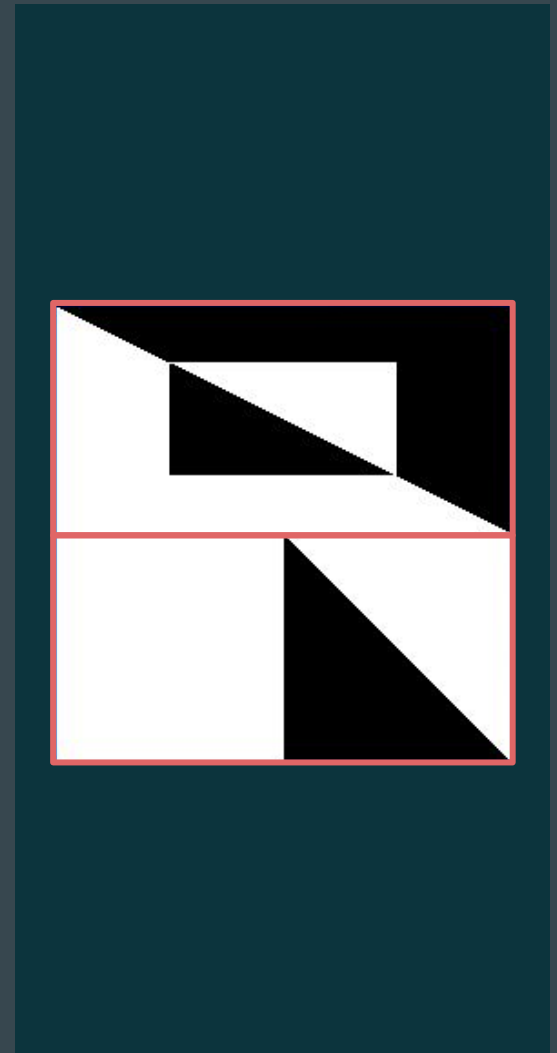
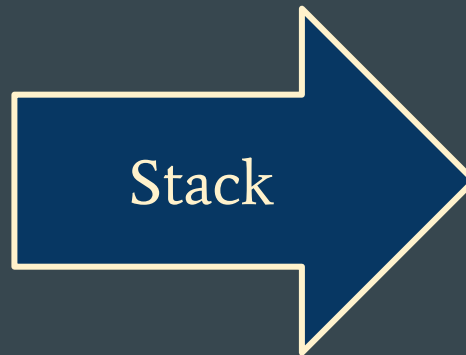
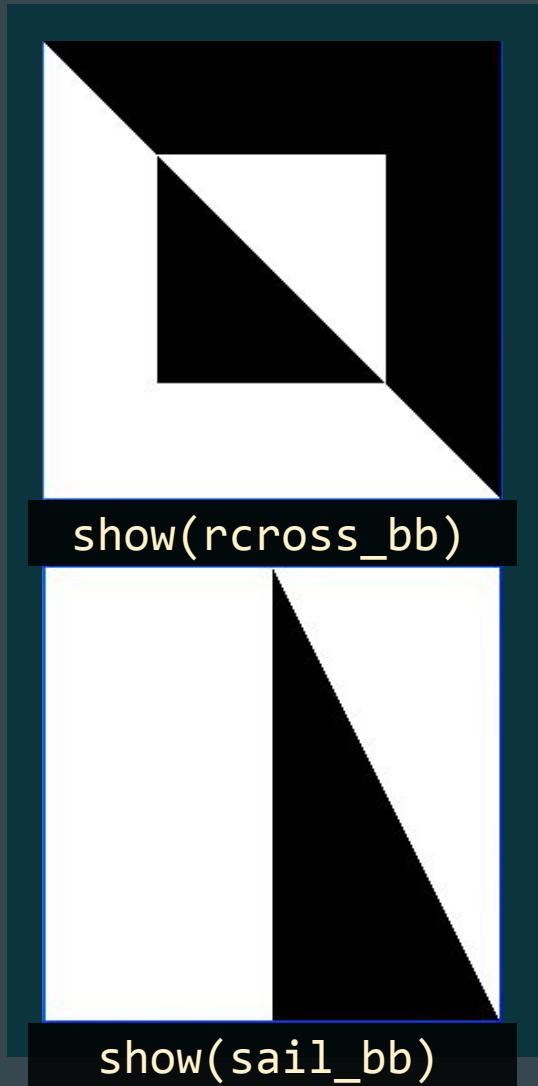
To make complex runes, we need ways to combine runes



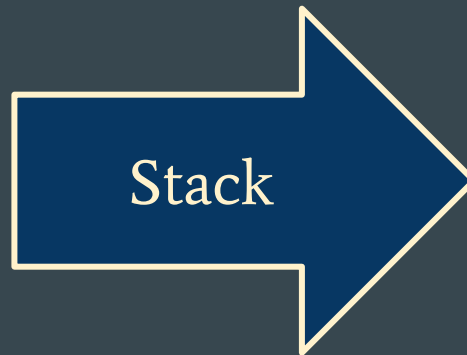
Stacking Pictures



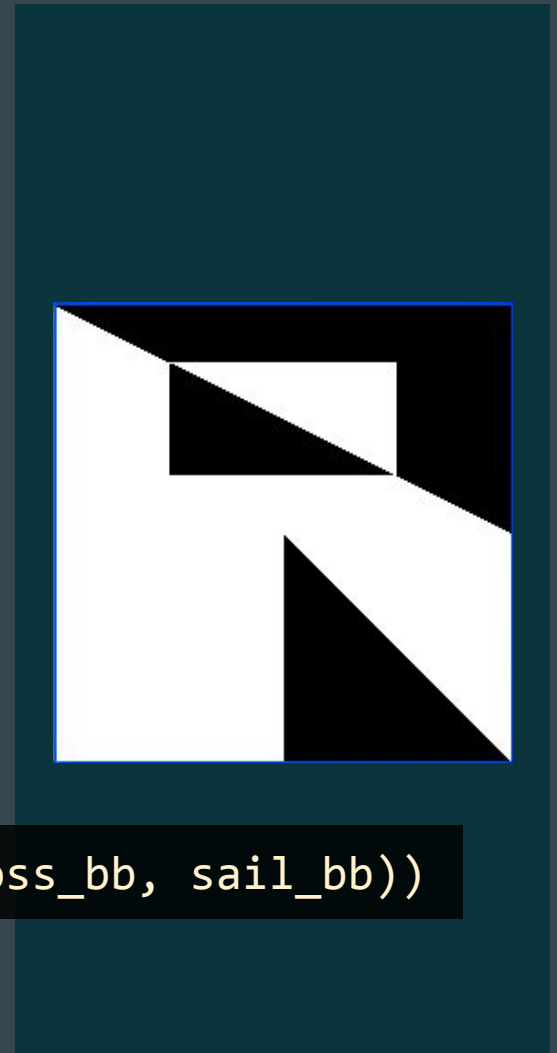
Stacking Pictures



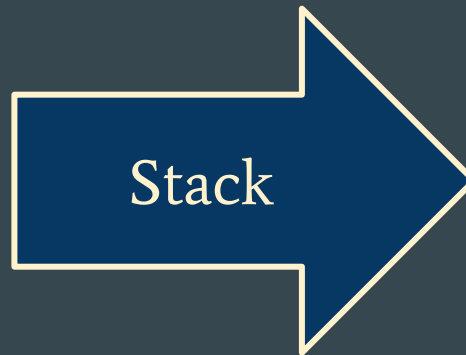
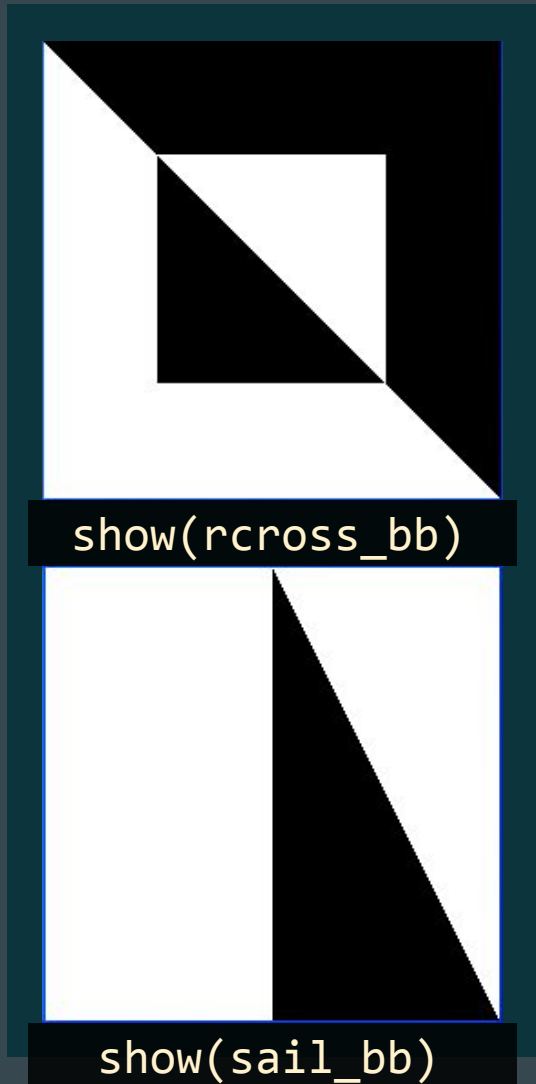
Stacking Pictures



`show(stack(rcross_bb, sail_bb))`



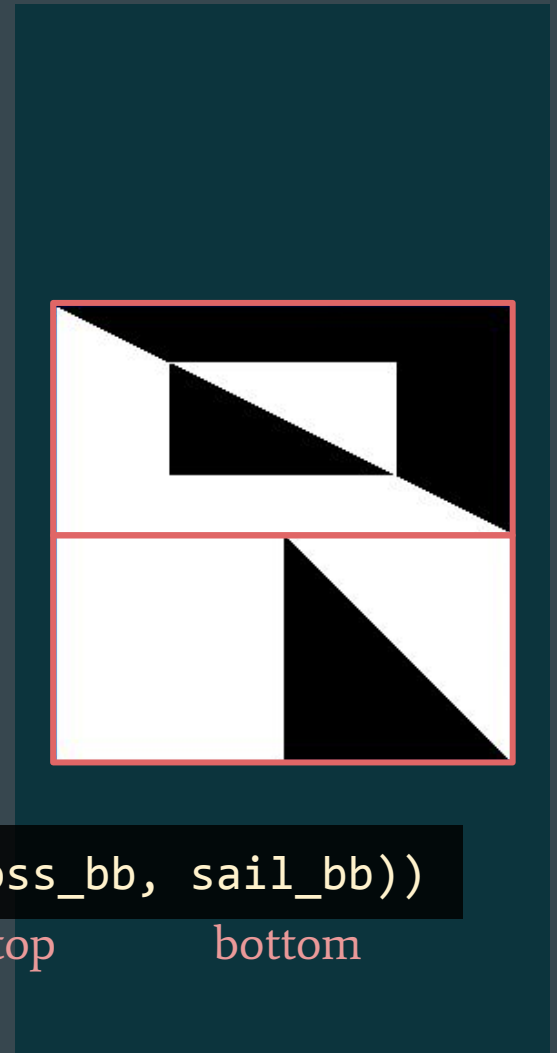
Stacking Pictures



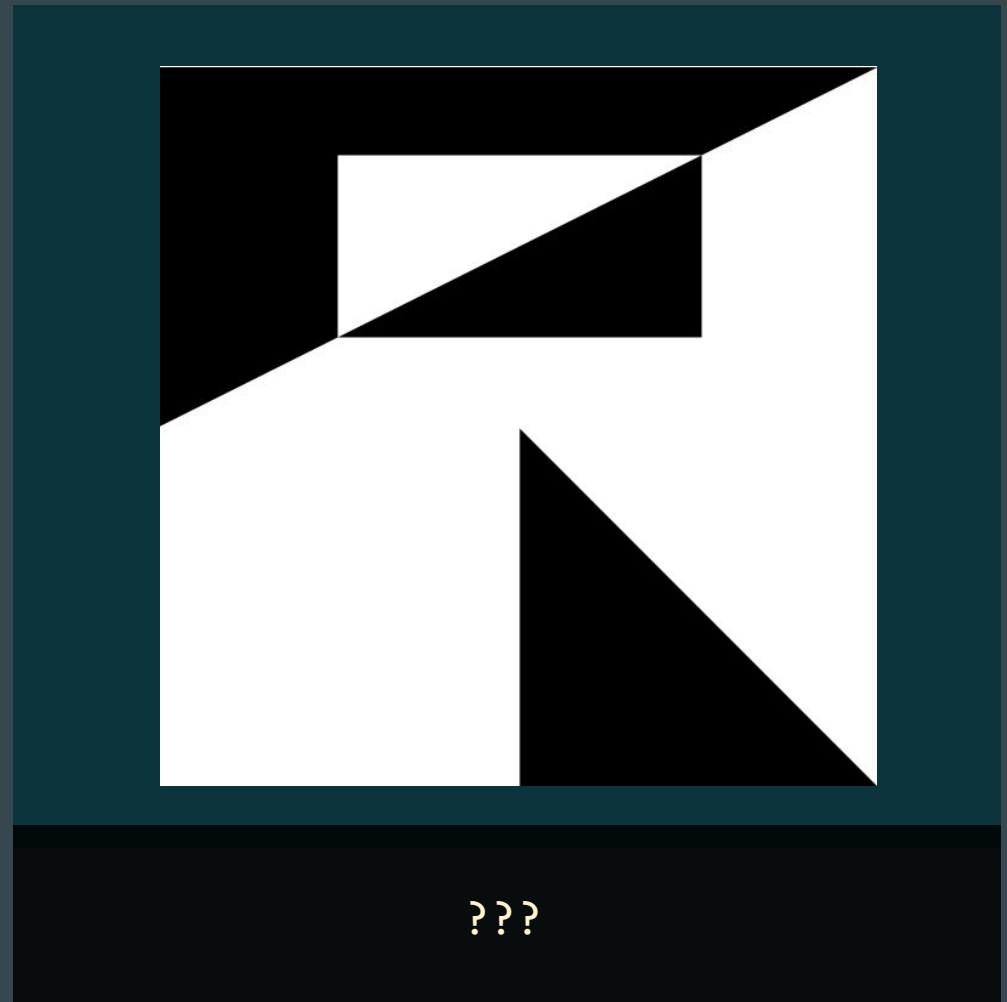
```
show(stack(rcross_bb, sail_bb))
```

top

bottom



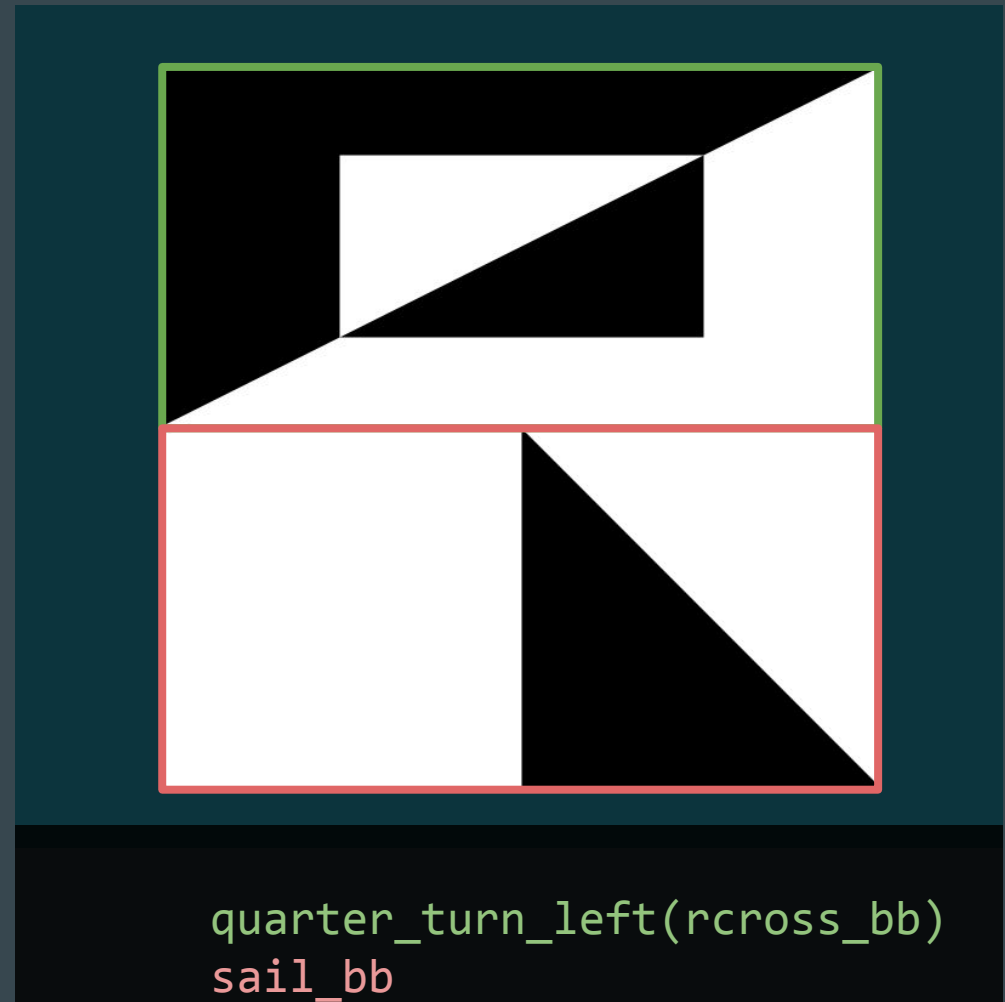
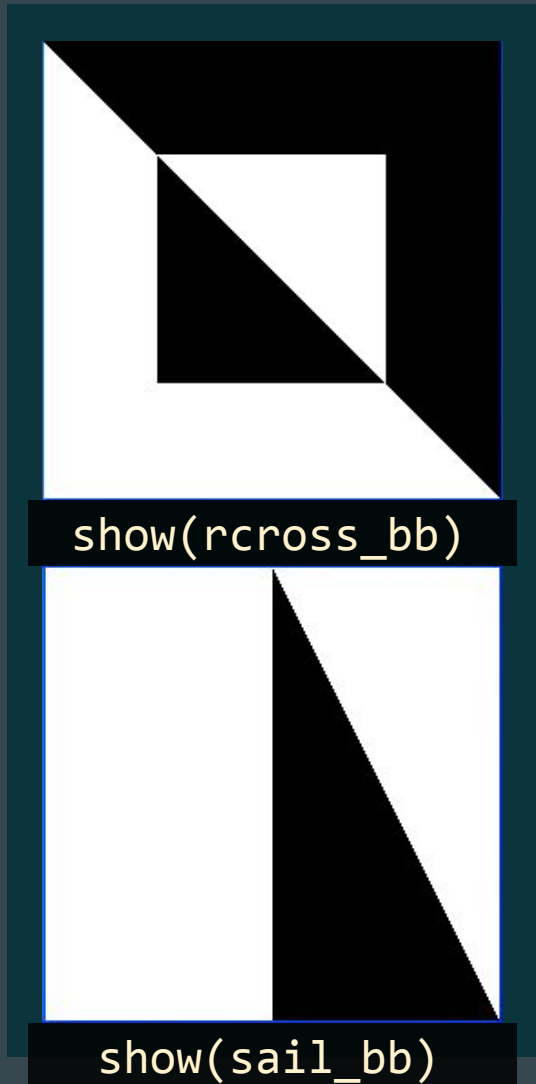
Turning and Stacking



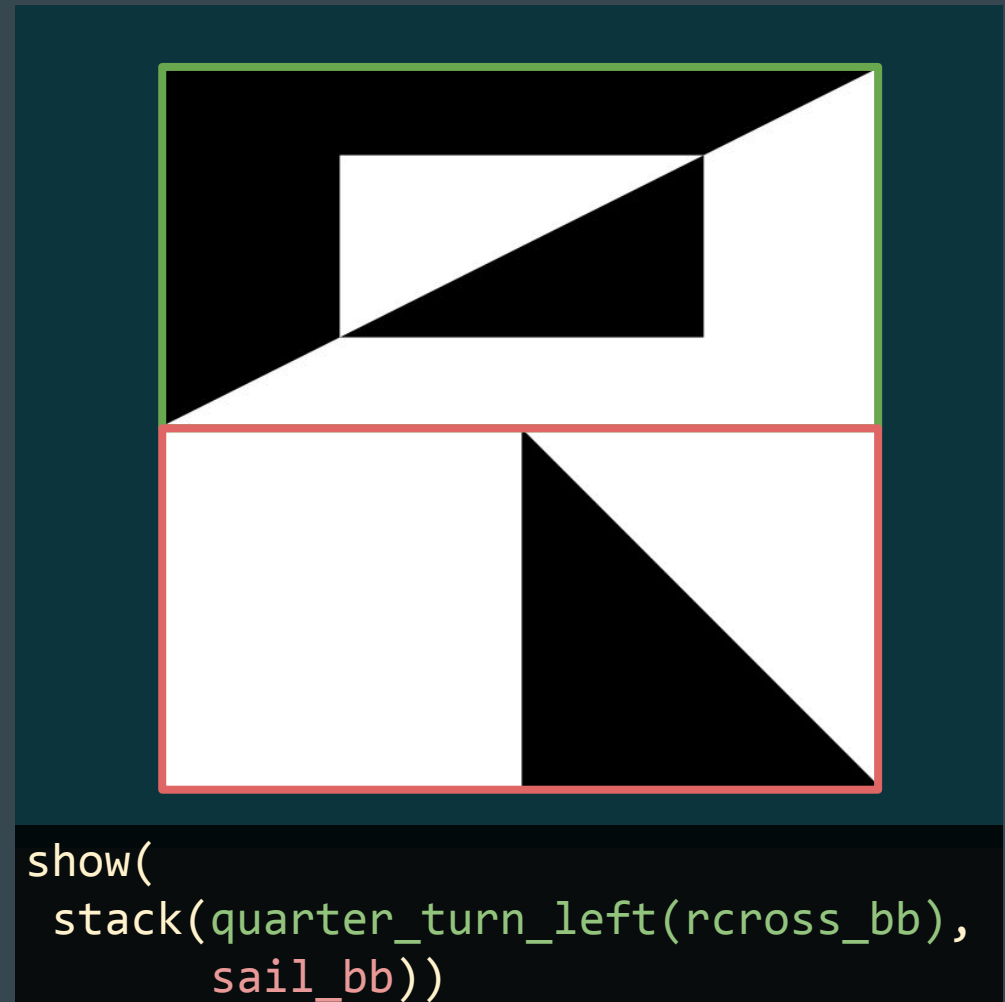
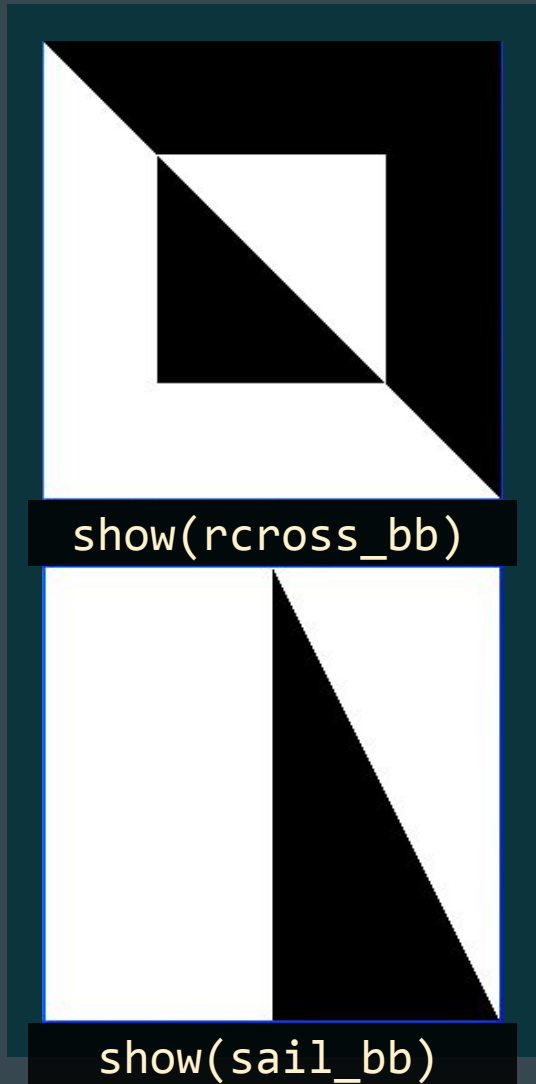
Turning and Stacking



Turning and Stacking



Turning and Stacking



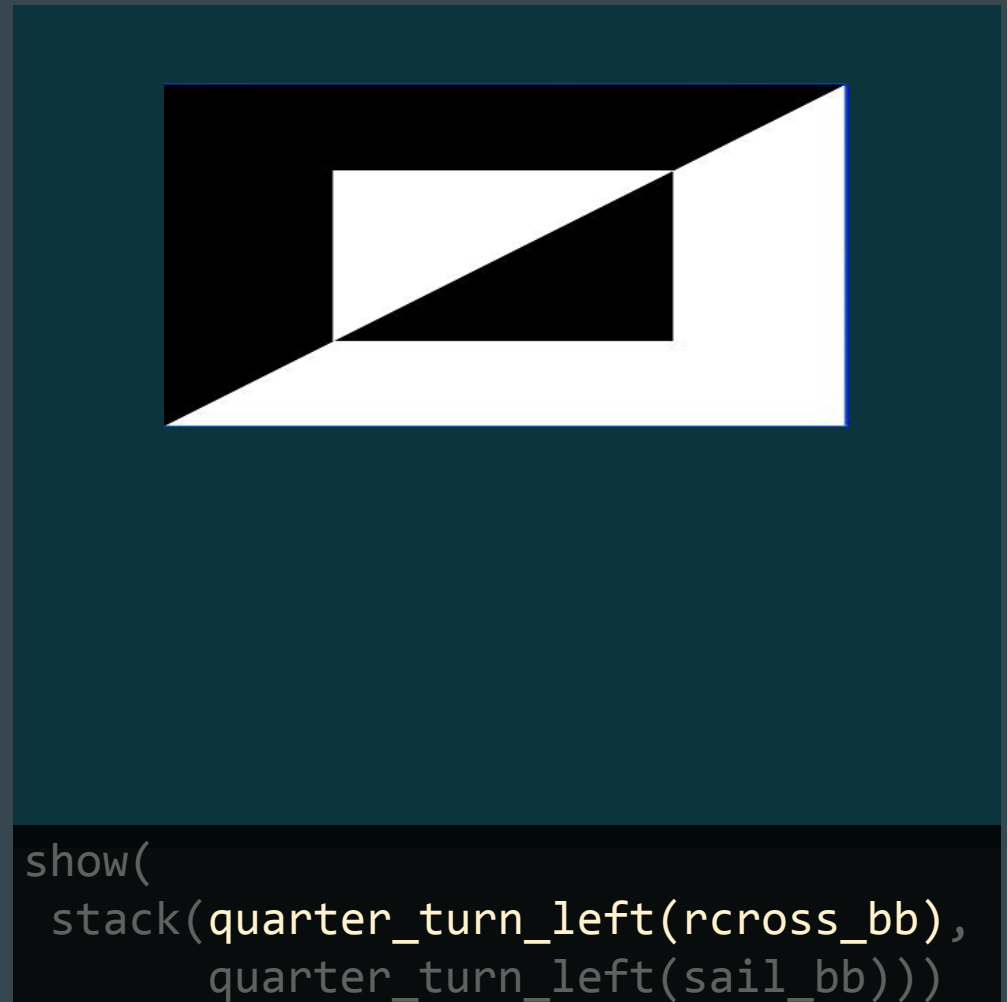
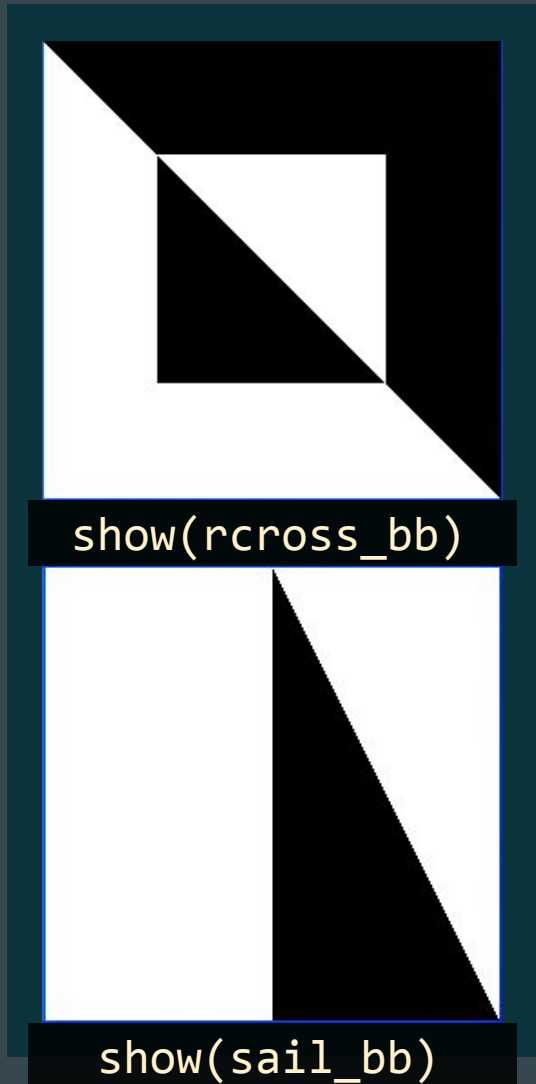
Turning and Stacking



???

```
show(  
    stack(quarter_turn_left(rcross_bb),  
          quarter_turn_left(sail_bb)))
```

Turning and Stacking



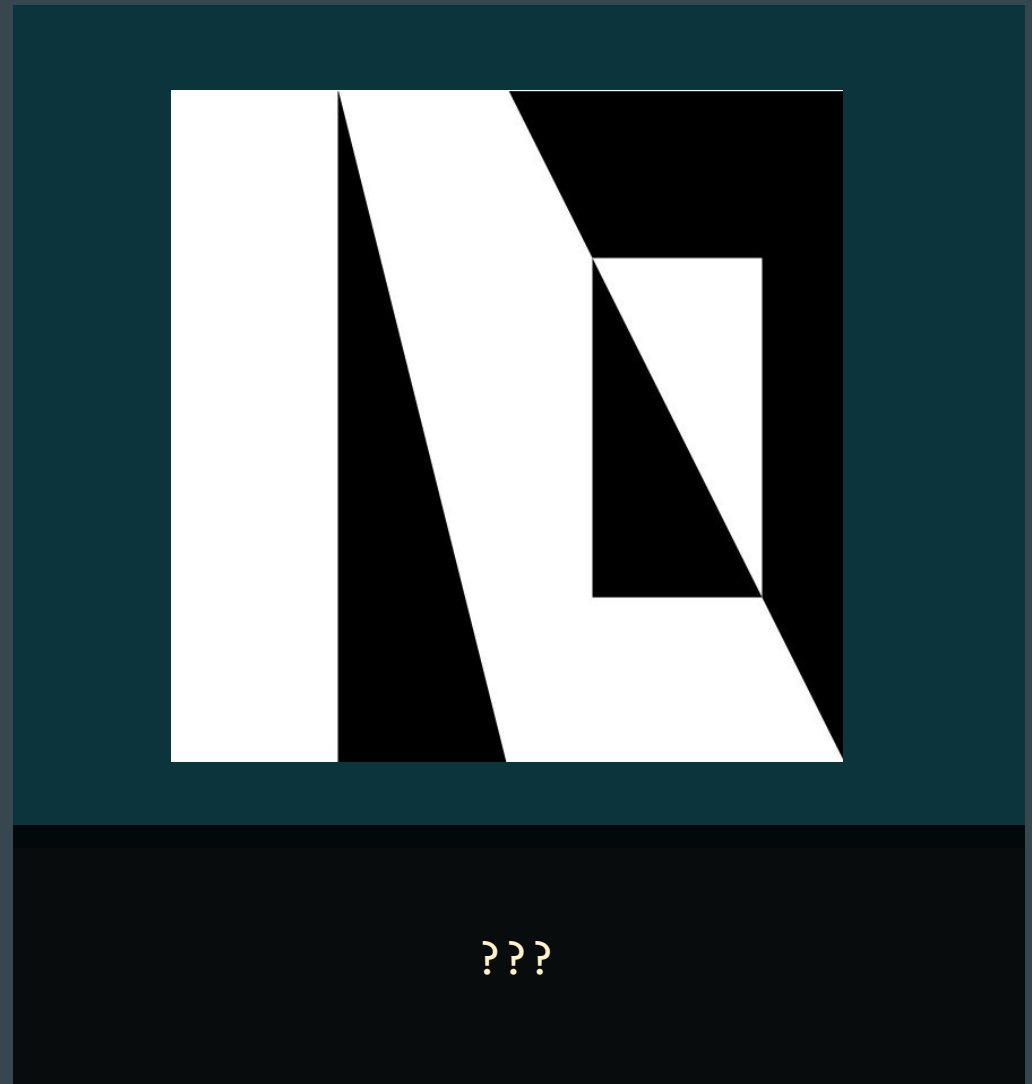
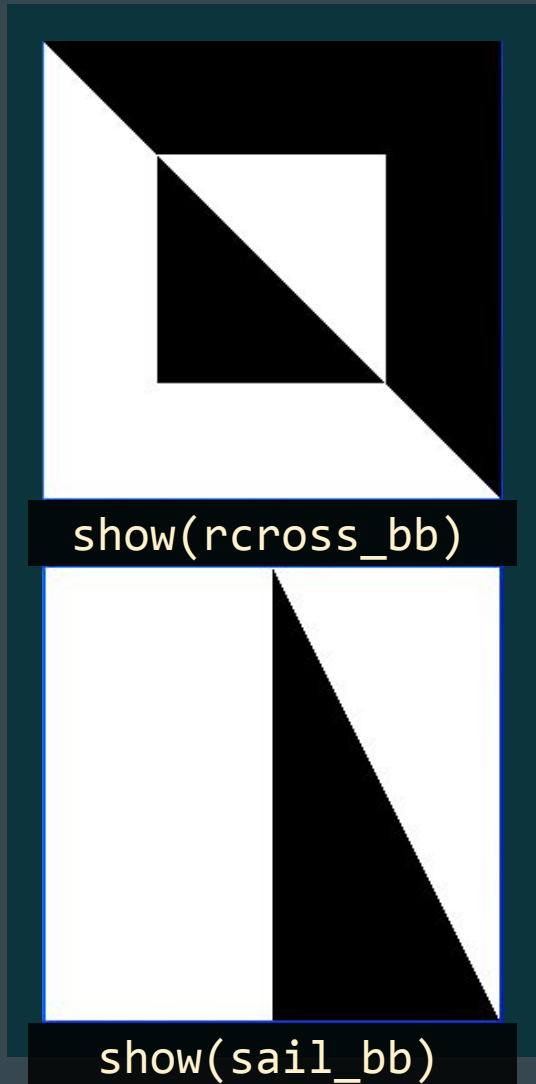
Turning and Stacking



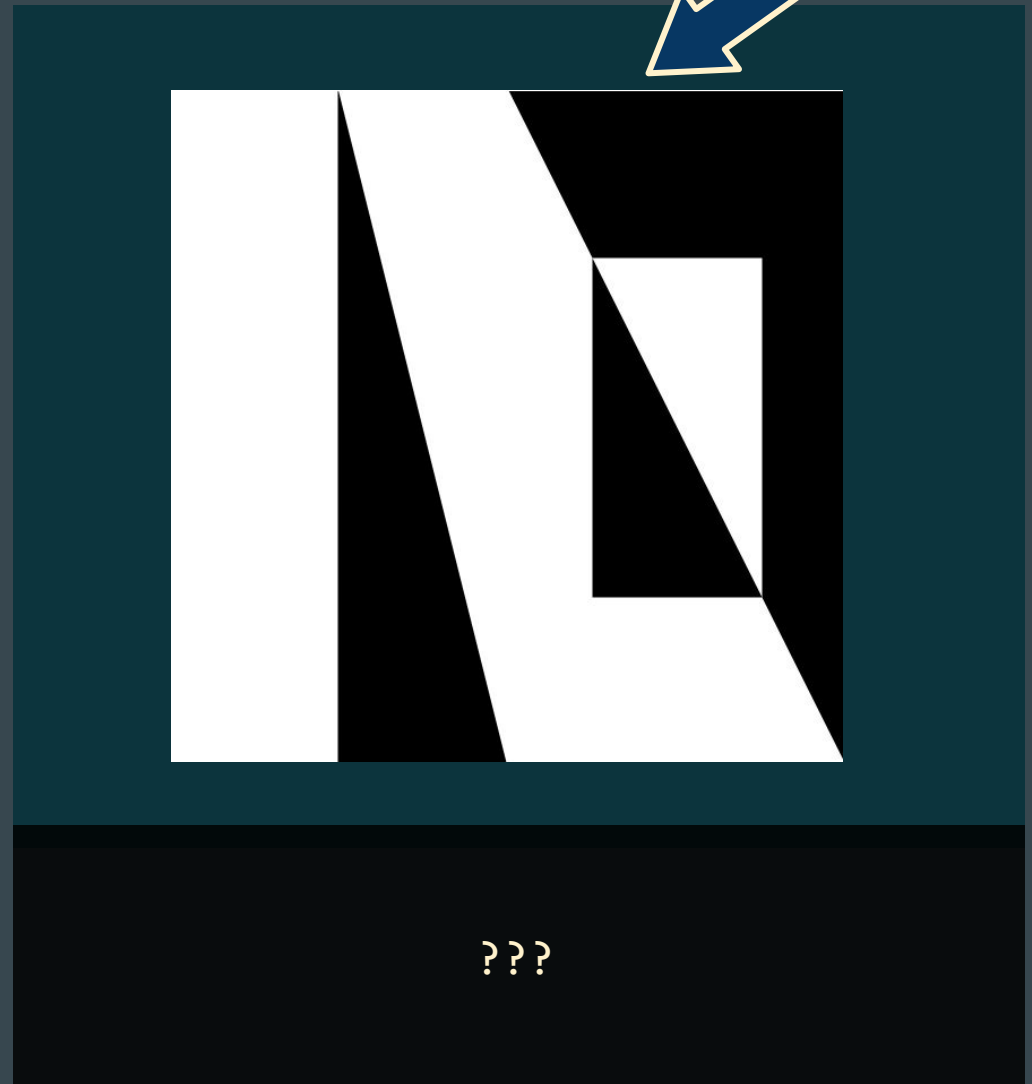
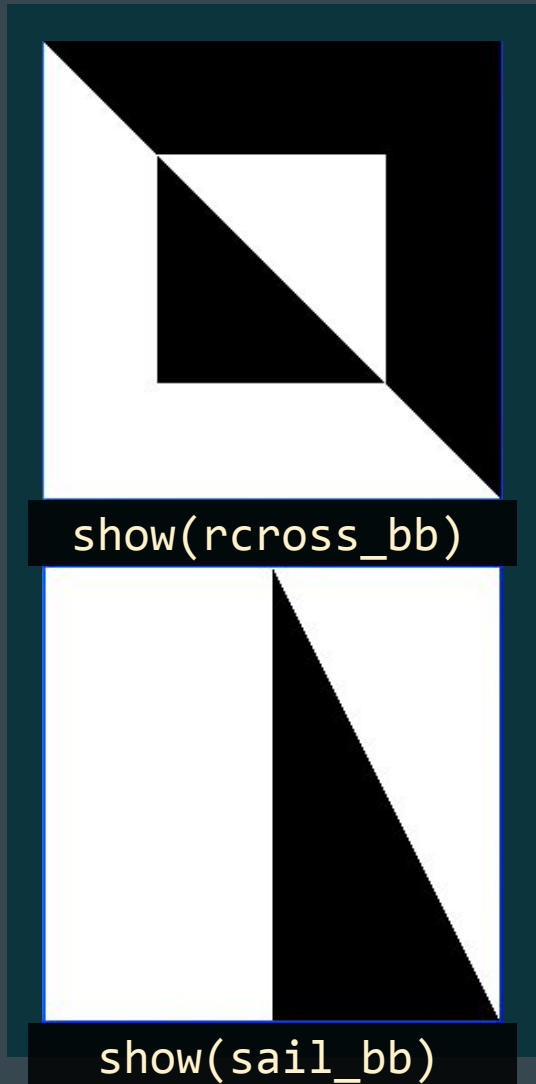
Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) Combining runes
 - top and bottom → left and right
 - complex patterns

Putting runes beside one another



Putting runes beside one another



Putting runes beside one another



quarter_turn_right



`show(rcross_bb)`

`show(sail_bb)`

???

Putting runes beside one another




quarter_turn_right



`show(rcross_bb)`



`show(sail_bb)`



```
show(  
  quarter_turn_right(  
    stack(quarter_turn_left(rcross_bb),  
          quarter_turn_left(sail_bb)))
```

Putting runes beside one another



quarter_turn_right

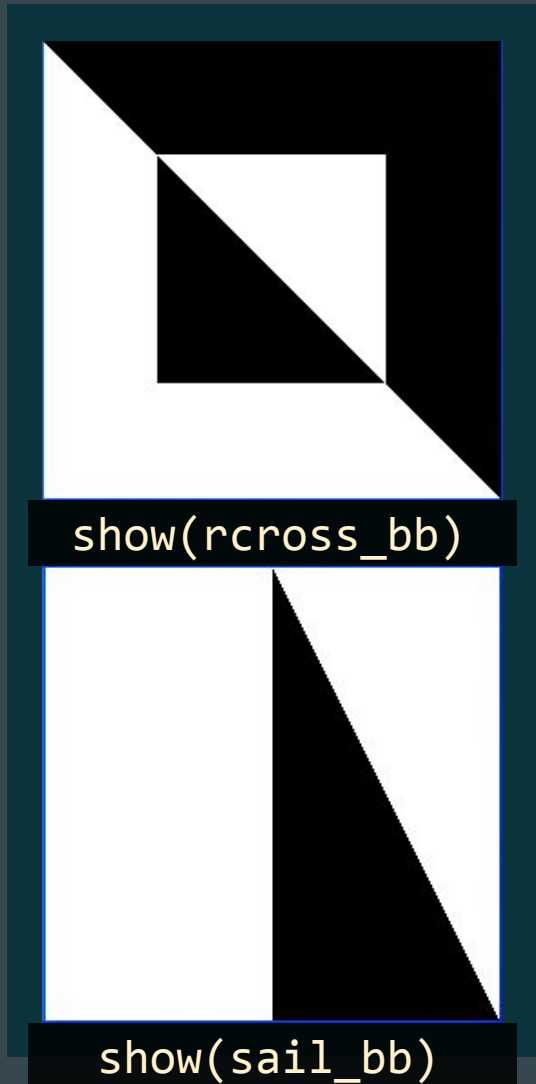


`show(rcross_bb)`

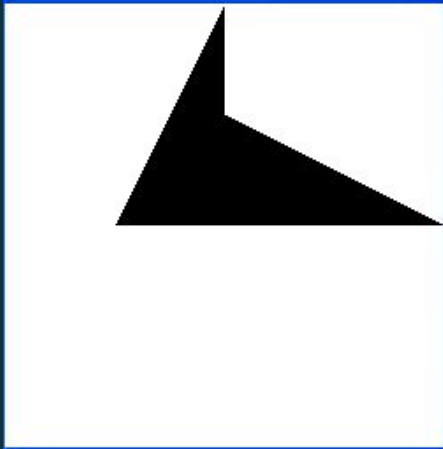
`show(sail_bb)`

```
show(  
  quarter_turn_right(  
    stack(quarter_turn_left(rcross_bb),  
          quarter_turn_left(sail_bb)))
```

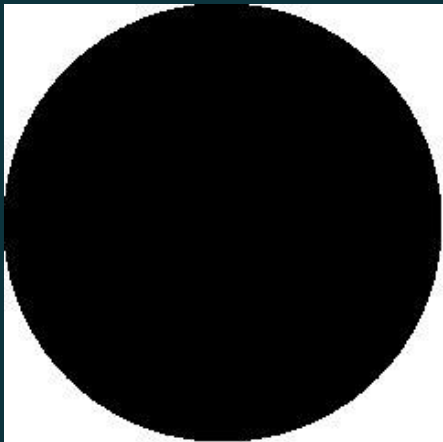
Wouldn't it be nice to have the function beside



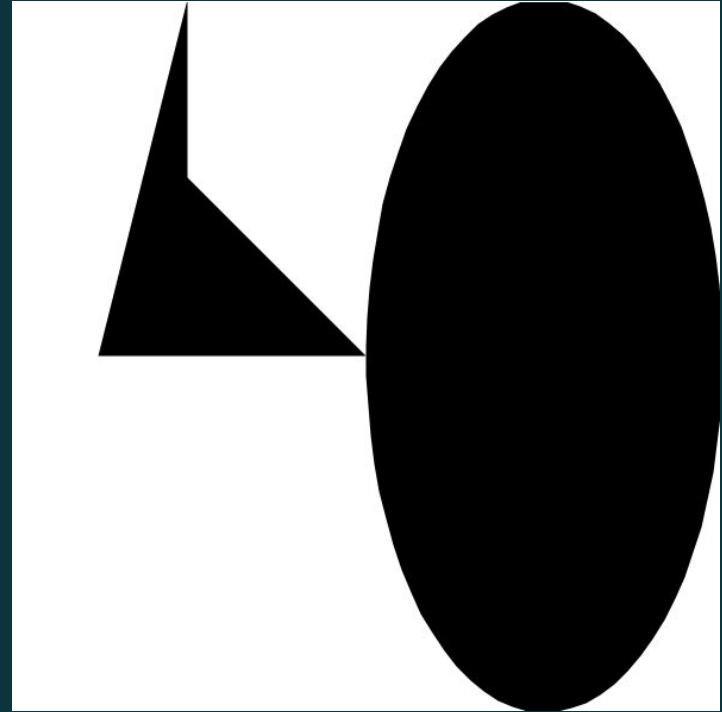
Wouldn't it be nice to have the function beside



```
show(nova_bb)
```

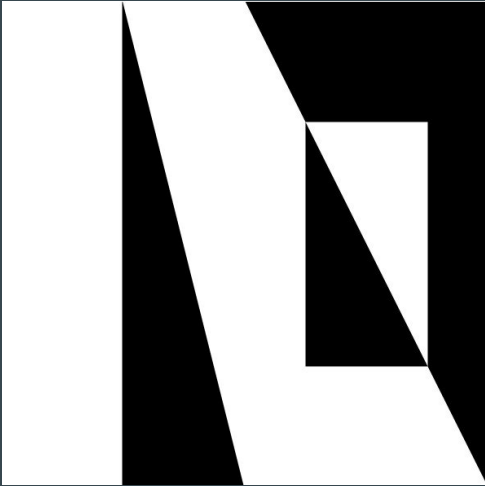


```
show(circle_bb)
```



```
show(beside(nova_bb, circle_bb))
```

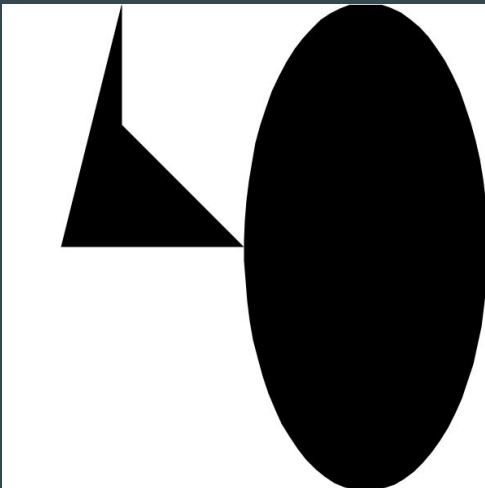
Writing Beside



```
beside(sail_bb, rcross_bb)
```



```
quarter_turn_right(  
    stack(quarter_turn_left(rcross_bb),  
          quarter_turn_left(sail_bb)))
```

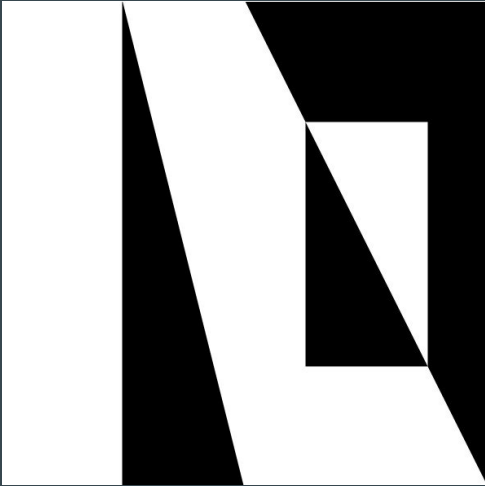


```
beside(nova_bb, circle_bb)
```



```
quarter_turn_right(  
    stack(quarter_turn_left(circle_bb),  
          quarter_turn_left(nova_bb)))
```

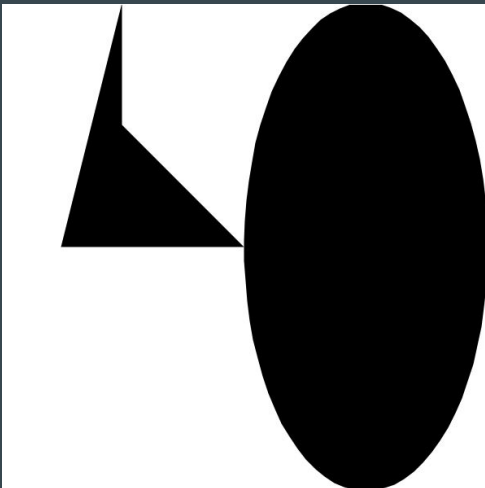
Writing Beside



```
beside(sail_bb, rcross_bb)
```



```
quarter_turn_right(  
  stack(quarter_turn_left(rcross_bb),  
        quarter_turn_left(sail_bb)))
```

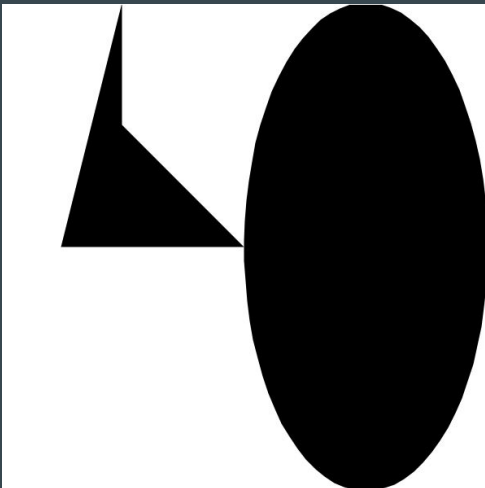
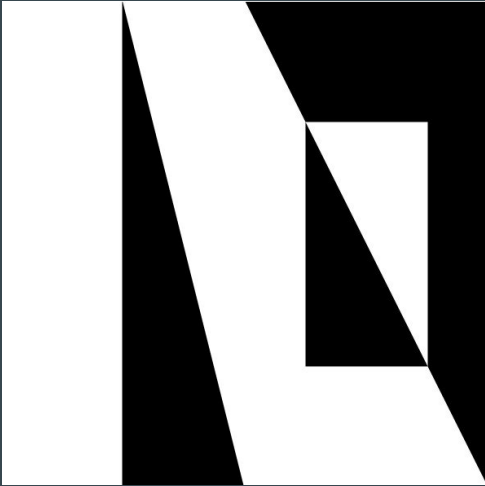


```
beside(nova_bb, circle_bb)
```



```
quarter_turn_right(  
  stack(quarter_turn_left(circle_bb),  
        quarter_turn_left(nova_bb)))
```

Writing Beside



```
beside(sail_bb, rcross_bb)
```



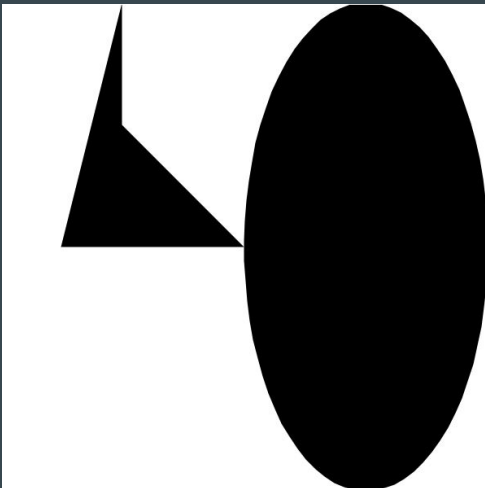
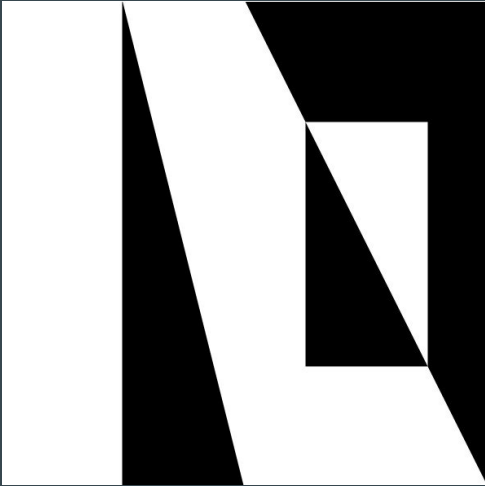
```
quarter_turn_right(  
  stack(quarter_turn_left(rcross_bb),  
        quarter_turn_left(sail_bb)))
```

```
beside(nova_bb, circle_bb)
```



```
quarter_turn_right(  
  stack(quarter_turn_left(circle_bb),  
        quarter_turn_left(nova_bb)))
```

Writing Beside



```
beside(pic1, rcross_bb)
```



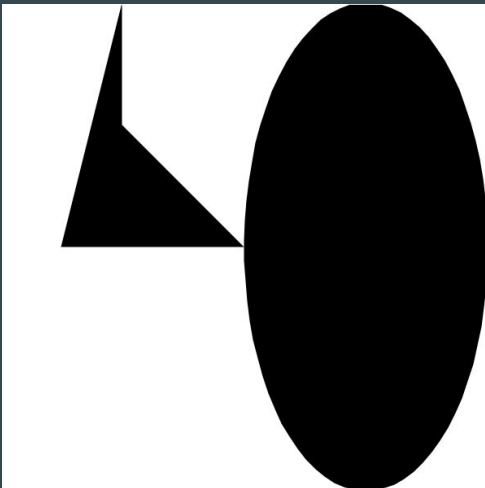
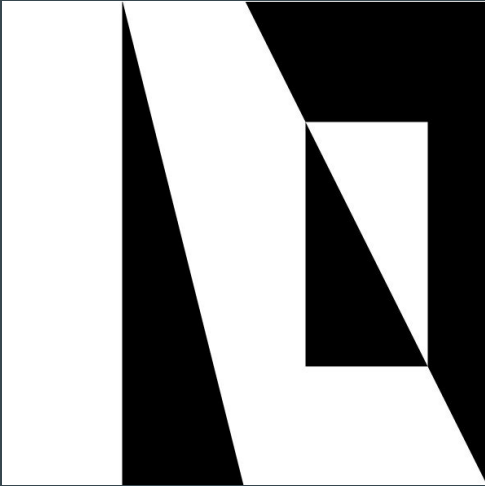
```
quarter_turn_right(  
  stack(quarter_turn_left(rcross_bb),  
        quarter_turn_left(pic1)))
```

```
beside(pic1, circle_bb)
```



```
quarter_turn_right(  
  stack(quarter_turn_left(circle_bb),  
        quarter_turn_left(pic1)))
```

Writing Beside



```
beside(pic1, pic2)
```



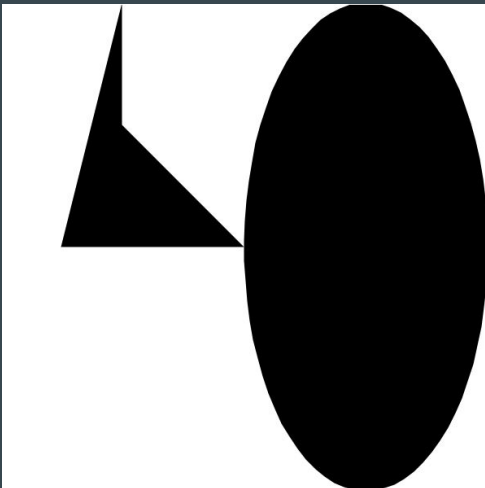
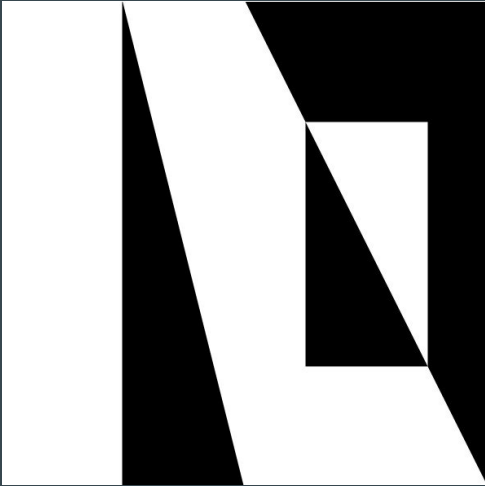
```
quarter_turn_right(  
  stack(quarter_turn_left(pic2),  
        quarter_turn_left(pic1)))
```

```
beside(pic1, pic2)
```



```
quarter_turn_right(  
  stack(quarter_turn_left(pic2),  
        quarter_turn_left(pic1)))
```

Writing Beside



```
beside(pic1, pic2)
```



```
quarter_turn_right(  
  stack(quarter_turn_left(pic2),  
        quarter_turn_left(pic1)))
```

```
beside(pic1, pic2)
```



```
quarter_turn_right(  
  stack(quarter_turn_left(pic2),  
        quarter_turn_left(pic1)))
```

Writing Beside

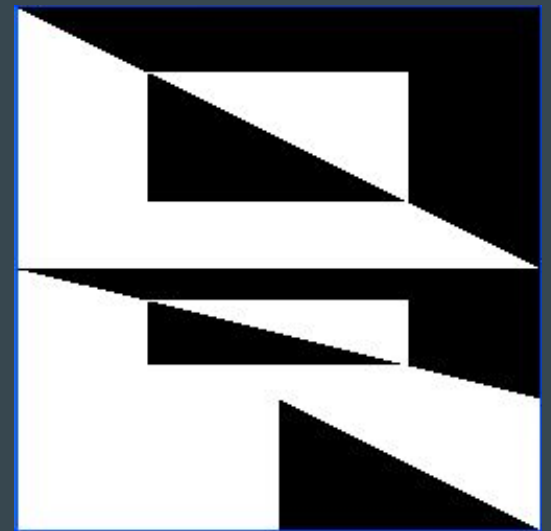
```
def beside(pic1, pic2):  
    return quarter_turn_right(  
        stack(quarter_turn_left(pic2),  
            quarter_turn_left(pic1)))
```


Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) Combining runes
 - top and bottom → left and right
 - complex patterns

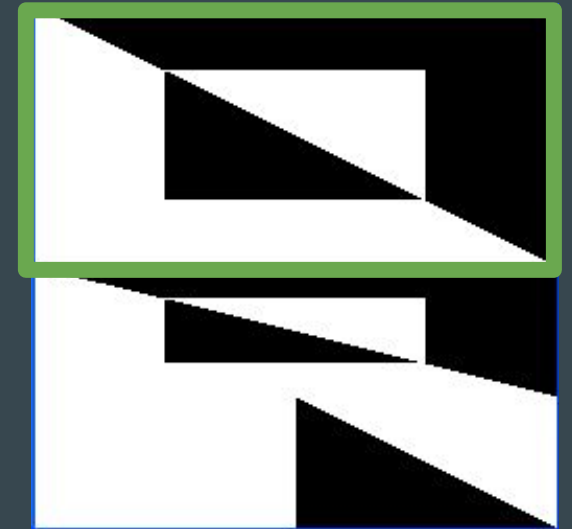
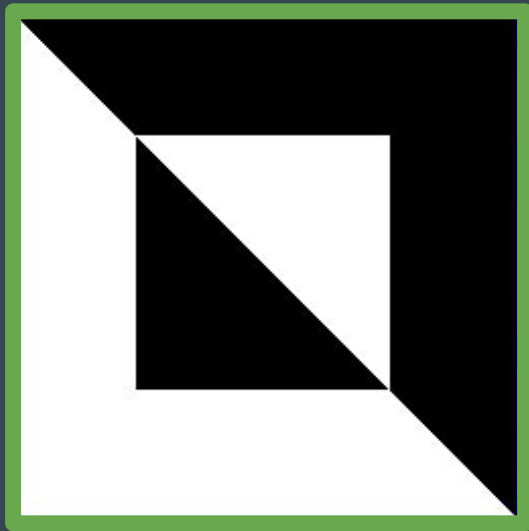
More than one rune in a picture?

???



More than one rune in a picture?

???



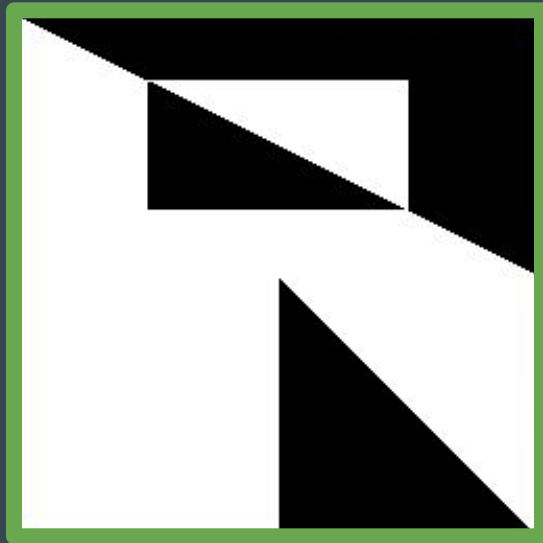
rcross_bb

More than one rune in a picture?

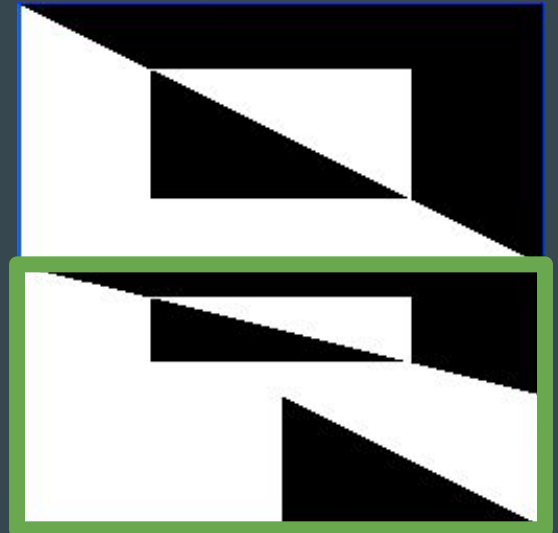
???



`rcross_bb`

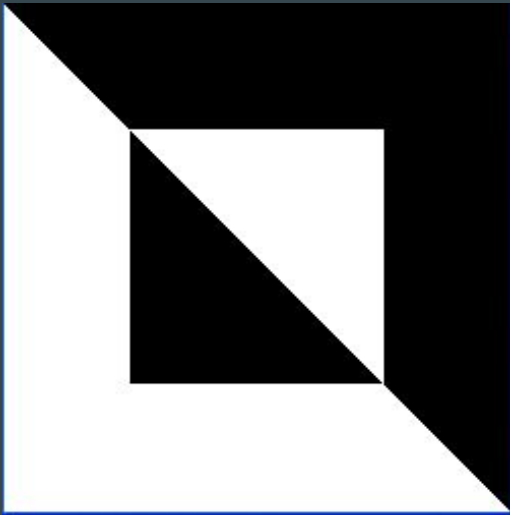


`stack(rcross_bb,sail_bb)`

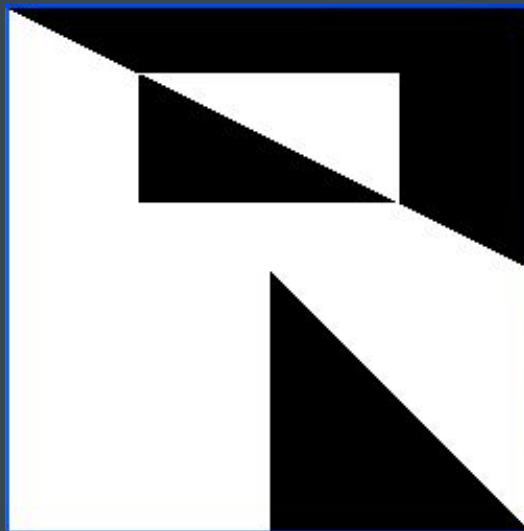


More than one rune in a picture?

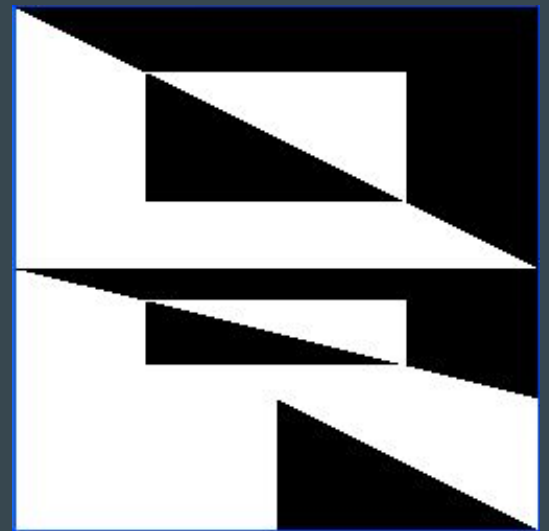
```
stack(           ,           )
```



`rcross_bb`

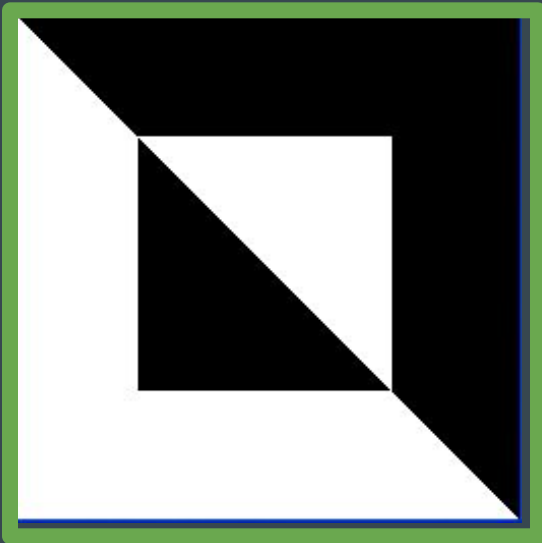


`stack(rcross_bb,sail_bb)`

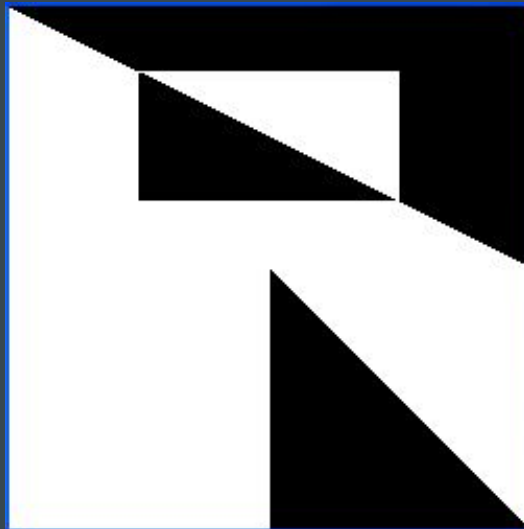


More than one rune in a picture?

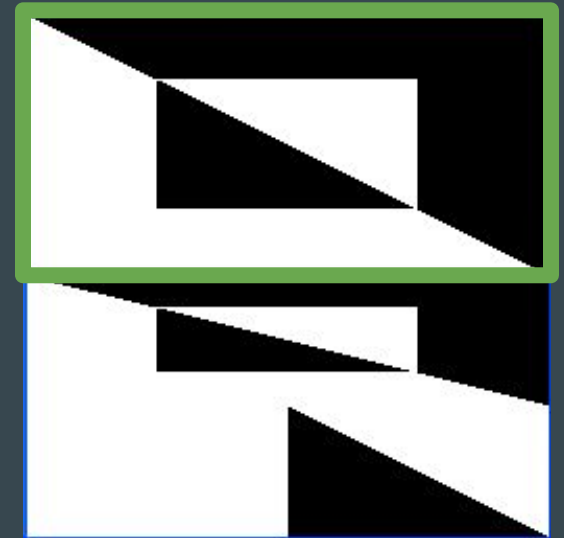
```
stack(rcross_bb, )
```



`rcross_bb`

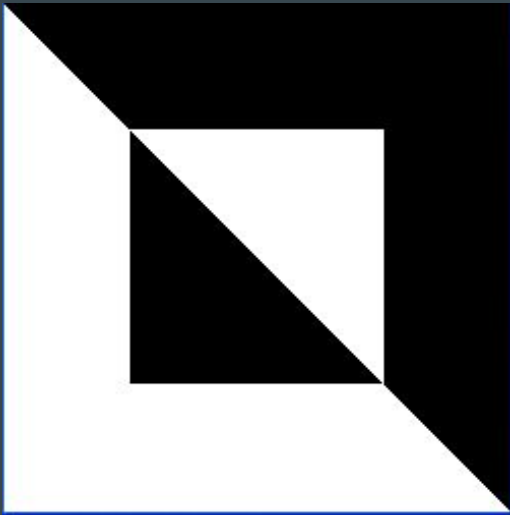


`stack(rcross_bb, sail_bb)`



More than one rune in a picture?

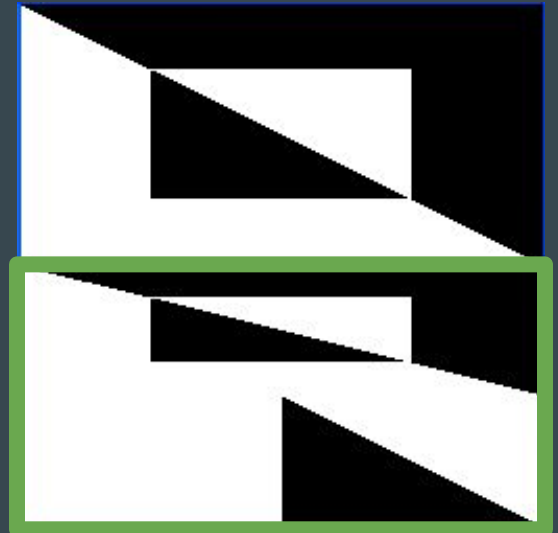
```
stack(rcross_bb,  
      stack(rcross_bb,sail_bb))
```



rcross_bb

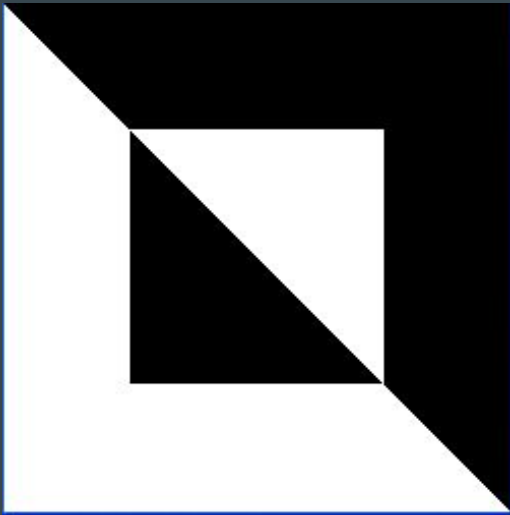


stack(rcross_bb,sail_bb)

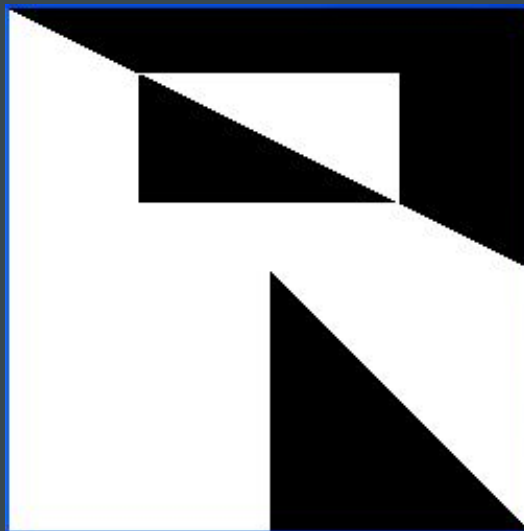


More than one rune in a picture?

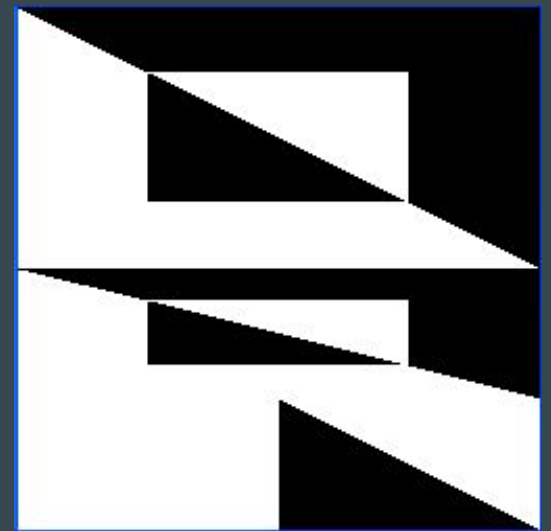
```
show(stack(rcross_bb,  
          stack(rcross_bb,sail_bb)))
```



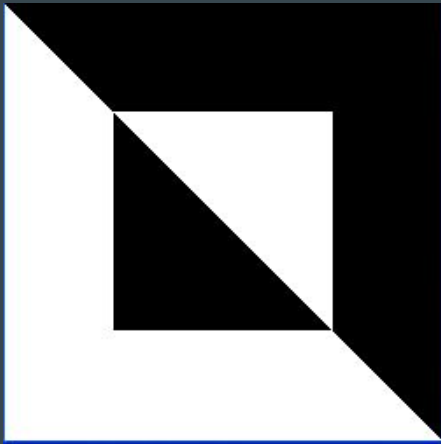
rcross_bb



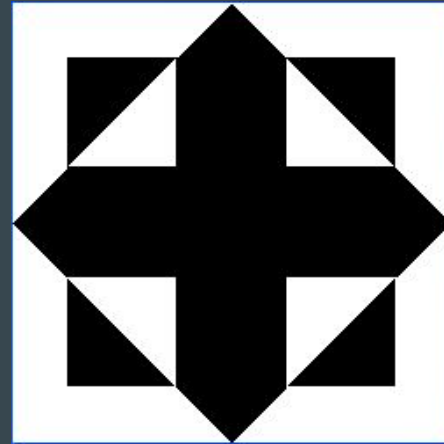
stack(rcross_bb,sail_bb)



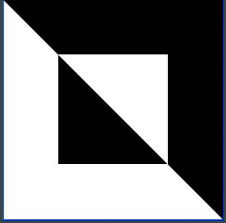
Nice patterns using both `stack` and `beside`



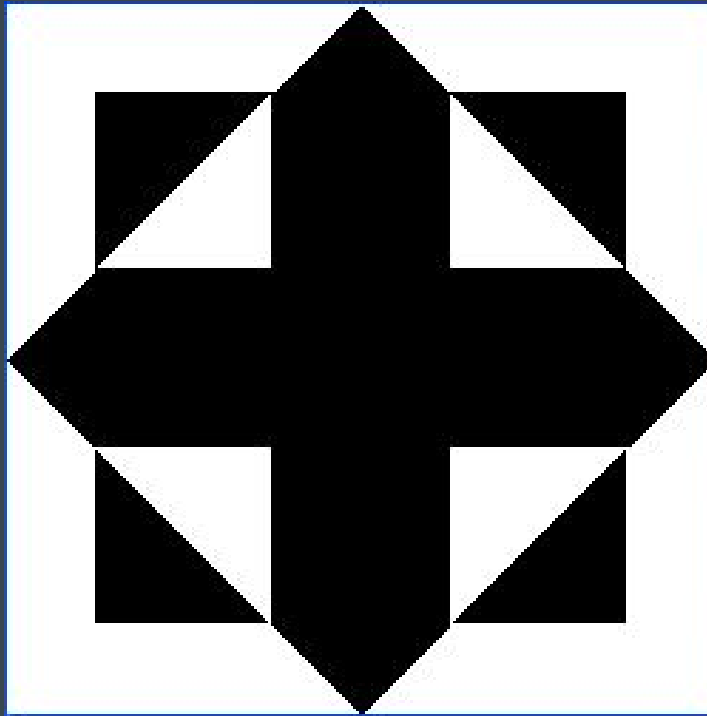
`show(rcross_bb)`



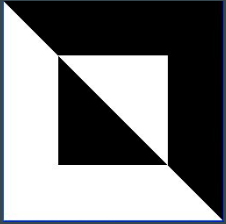
A closer look



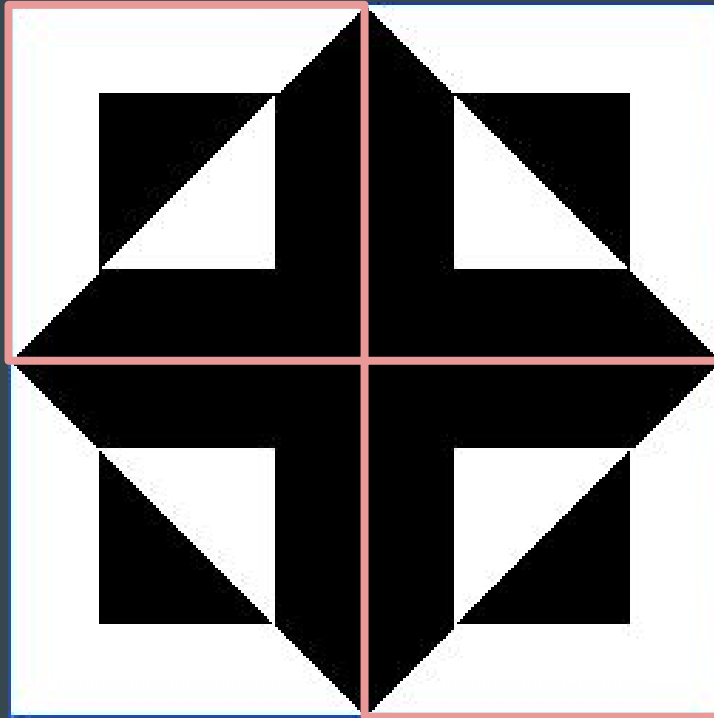
`show(rcross_bb)`



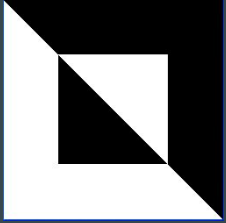
What are the parts?



`show(rcross_bb)`



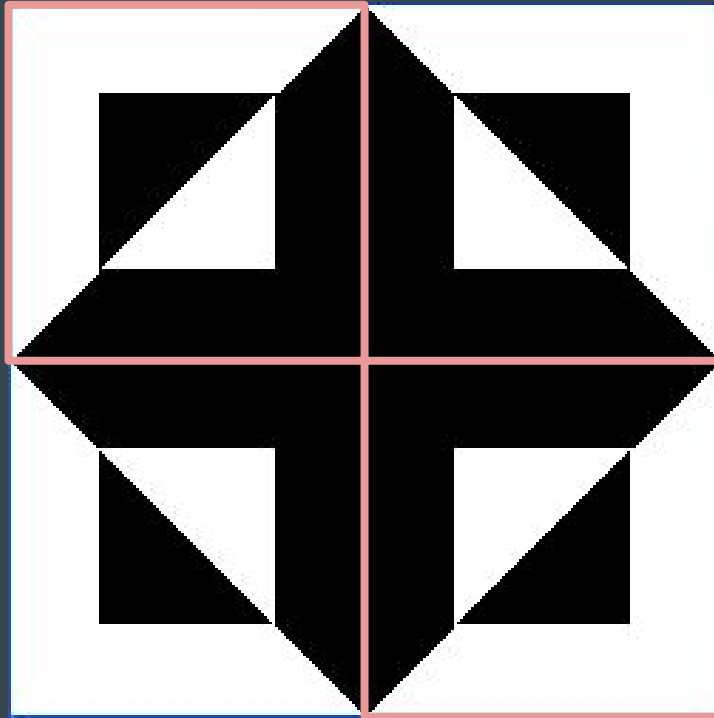
What are the parts?



`show(rcross_bb)`

`quarter_turn_right`

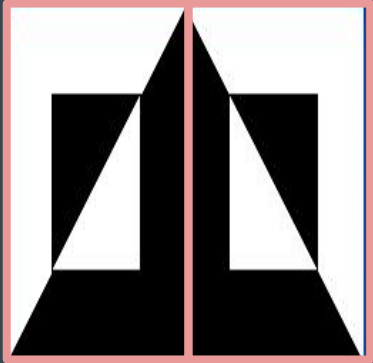
`turn_upside_down`



`quarter_turn_left`

Deconstructing

`quarter_turn_right`



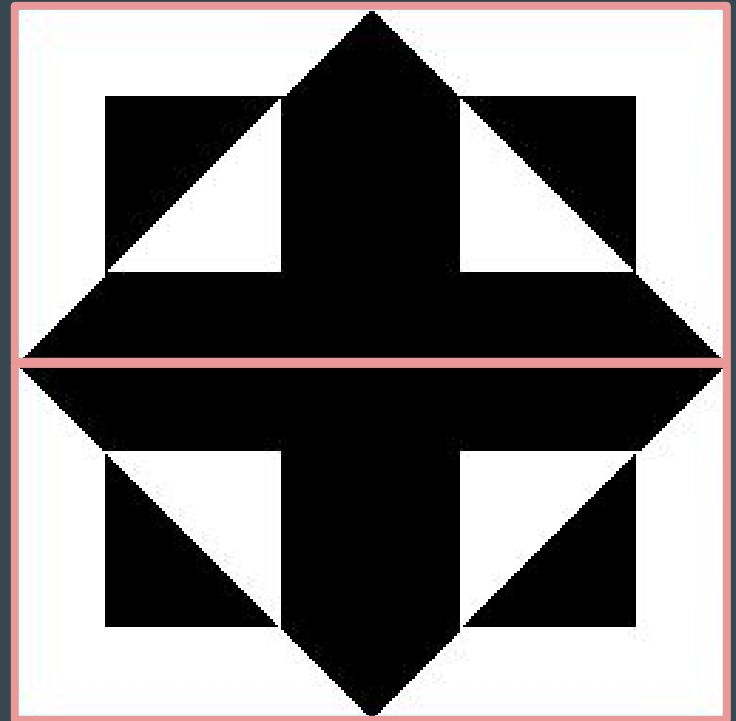
`turn_upside_down`



`quarter_turn_left`

Stack

`show(make_cross(rcross_bb))`

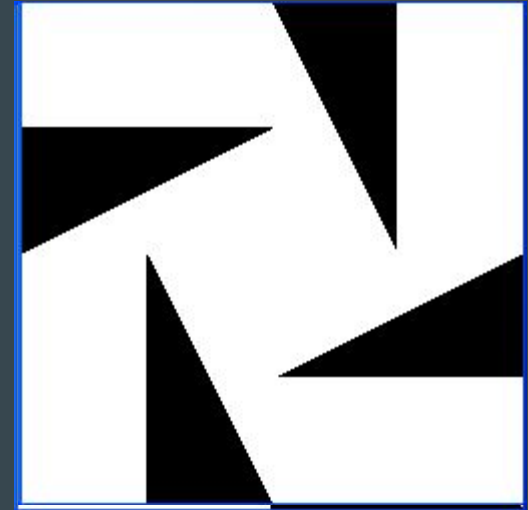


Make Cross

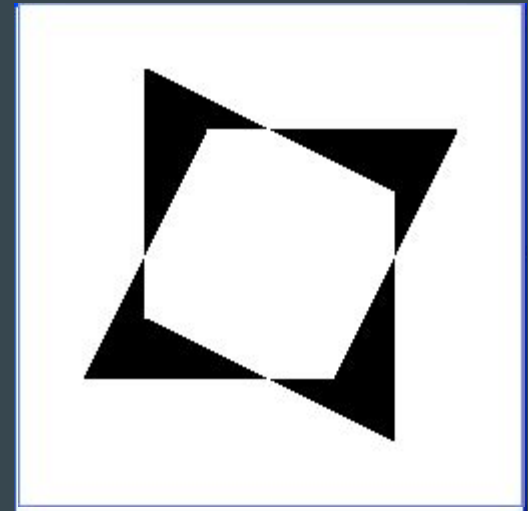
```
def make_cross(pic):  
    return stack(  
        beside(  
            quarter_turn_right(pic),  
            turn_upside_down(pic)),  
        beside(  
            rcross_bb,  
            quarter_turn_left(pic)))
```

Naming your Objects

```
my_pic = make_cross(sail_bb)  
show(my_pic)
```



```
shuriken = make_cross(nova_bb)  
show(shuriken)
```



Picture Language 1

- 1) Importing libraries
- 2) Showing runes
- 3) Basic functions & composing them
- 4) Combining runes
 - top and bottom → left and right
 - complex patterns