CS1010S

Tutorial 3: Advanced Iteration

Nicholas Russell Saerang (russellsaerang@u.nus.edu)

About Late Submission

For missions, the following penalty applies:

- < 24h late: 90%
- >= 24h late: 80%

Moral of the story: don’t need to rush assignments just to cost more mistakes than what 20% grade can give you!

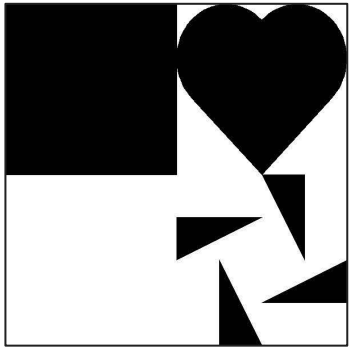
About Runes

Even more Runes

Runes

So you think you've mastered your runes

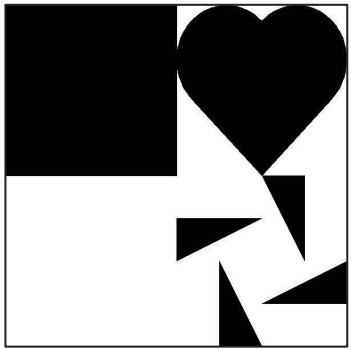
Mosaic



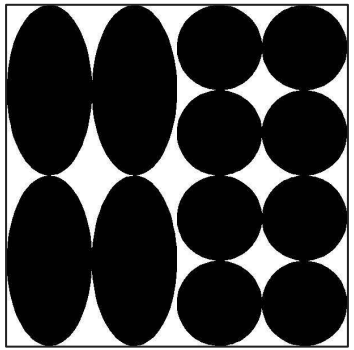
Runes

So you think you've mastered your runes

Mosaic



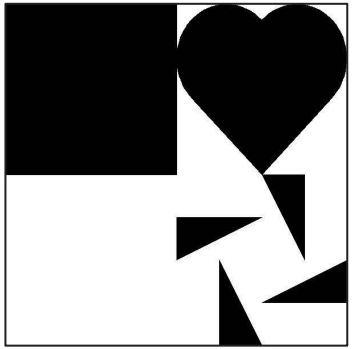
Simple Fractals



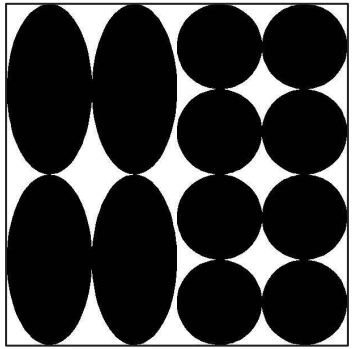
Runes

So you think you've mastered your runes

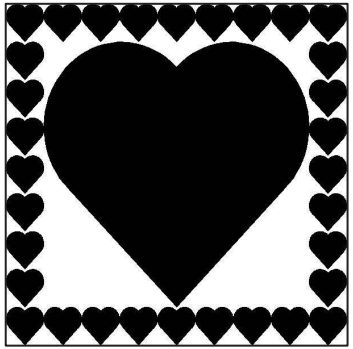
Mosaic



Simple Fractals



Egyptian



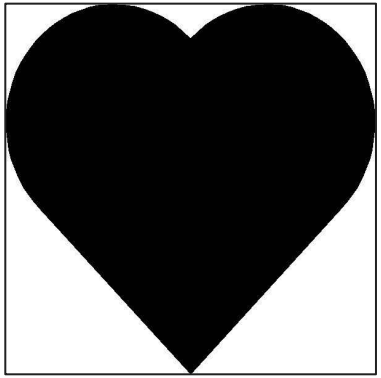
Runes

Runes

This week!

- Fractals

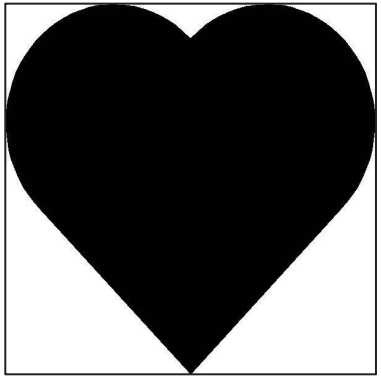
```
fractal(heart_bb, 1)
```



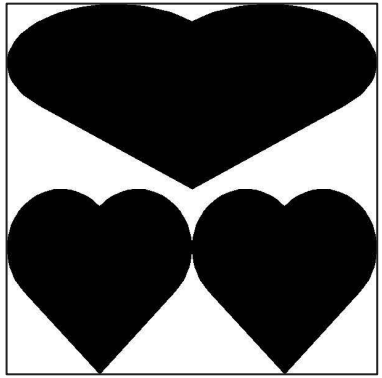
Runes

This week!
- Fractals

fractal(heart_bb, 1)



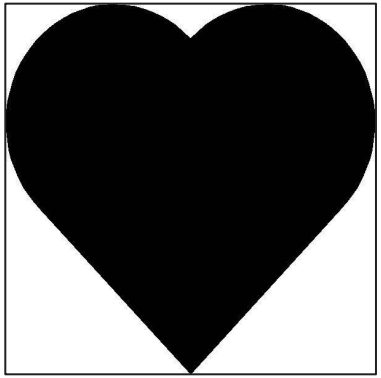
fractal(heart_bb, 2)



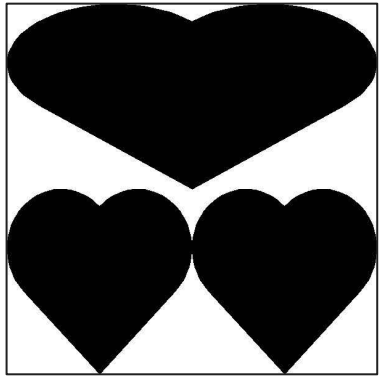
Runes

This week!
- Fractals

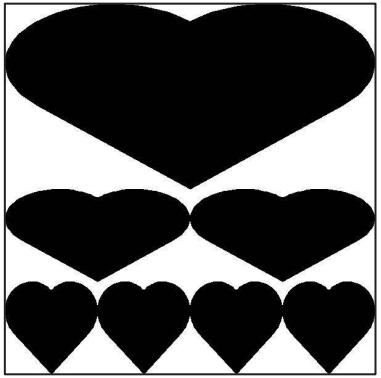
```
fractal(heart_bb, 1)
```



```
fractal(heart_bb, 2)
```



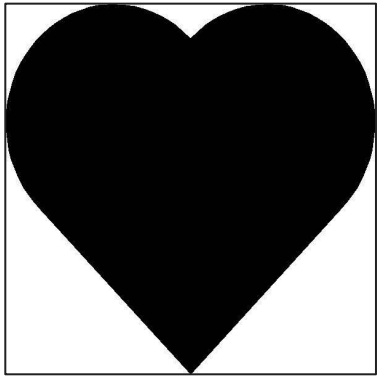
```
fractal(heart_bb, 3)
```



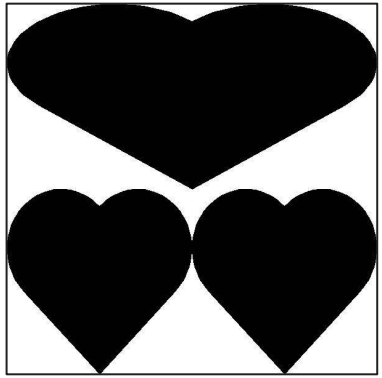
Runes

- What’s the pattern?

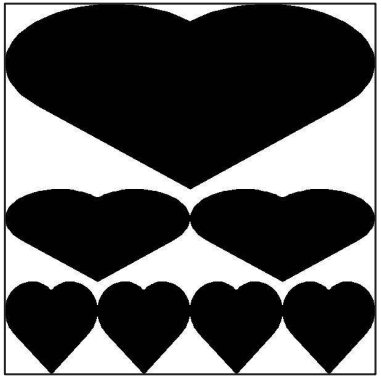
`fractal(heart_bb, 1)`



`fractal(heart_bb, 2)`



`fractal(heart_bb, 3)`

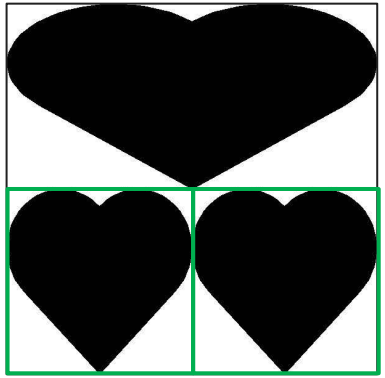


Runes

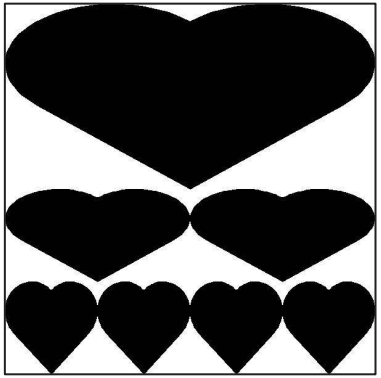
- What’s the pattern?
  - Can’t really tell from 1 to 2

`fractal(heart_bb, 1)`

`fractal(heart_bb, 2)`



`fractal(heart_bb, 3)`

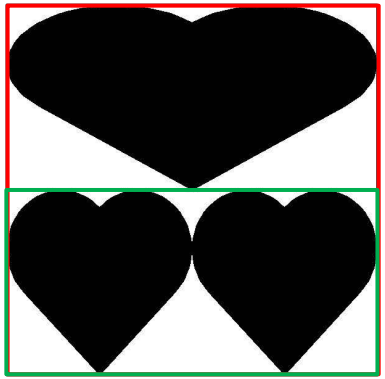


Runes

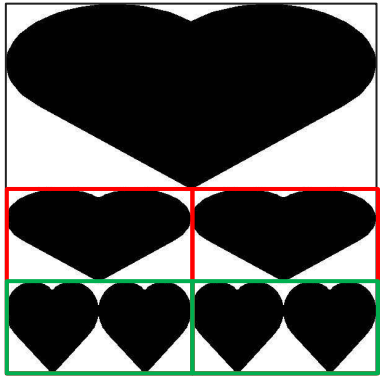
- What's the pattern?
  - How about 2 to 3?

`fractal(heart_bb, 1)`

`fractal(heart_bb, 2)`



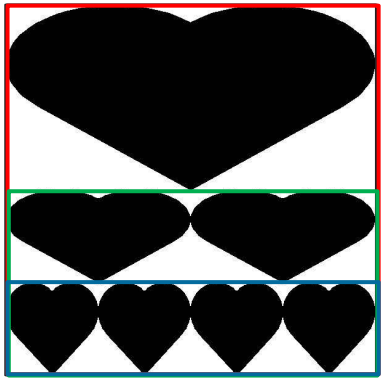
`fractal(heart_bb, 3)`



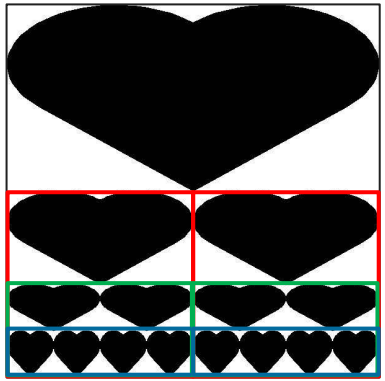
Runes

- What's the pattern?
  - And 3 to 4?

`fractal(heart_bb, 3)`



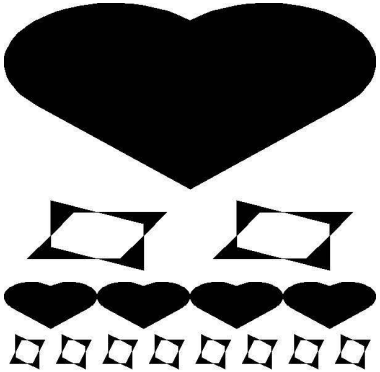
`fractal(heart_bb, 4)`



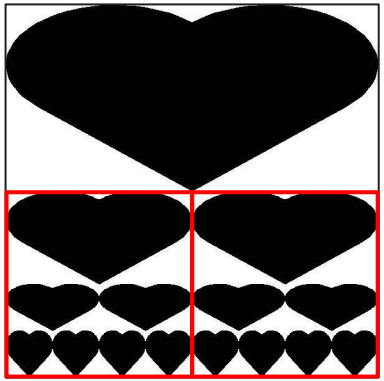
Runes

- Dual Fractals! Just the same as fractals, but with a twist

```
dual_fractal(heart_bb,
make_cross(nova_bb), 4)
```



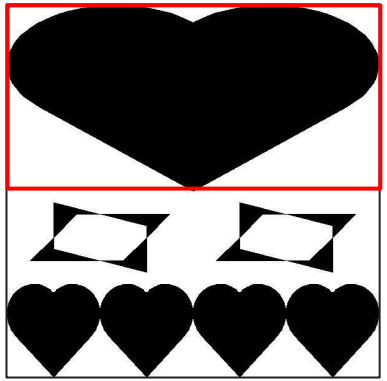
```
fractal(heart_bb, 4)
```



Runes

- Take note!
  - The first rune always appears at the top

```
dual_fractal(heart_bb,
make_cross(nova_bb), 3)
```



```
dual_fractal(heart_bb,
make_cross(nova_bb), 4)
```

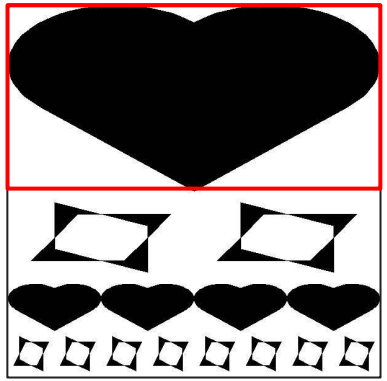


Table of contents

Advanced Iteration

Control Flow
For loops

- For loops contains initialization, increment and termination
- Usually when number of iterations are known

```
i = 0
while i < 10:
    print(i)
    i += 1
```

```
for i in range(0, 10, 1):
    print(i)
```

Control Flow
For loops

- For loops can iterate through other iterables

```
for i in range(3, 10, 2):
    print(i)

for val in [3, 5, 7, 9]:
    print(val)

for c in "hello":
    print(c)
```

Control Flow
Nested Loop

- You can nest for loops as well

```
for i in range(3, 10, 2):
    for j in range(1, 10, 1):
        print(i, j)
```

Tutorial 3

Advanced iteration

Hints

$$f(n) = \begin{cases} n & n < 3 \\ f(n-1) + 2f(n-2) + 3f(n-3) & n \geq 3 \end{cases}$$

Here is the hint for this question!!

Hints

$$f(n) = \boxed{f(n-1) + 2f(n-2) + 3f(n-3)}$$

$$\boxed{f(n-1) = f(n-2) + 2f(n-3)} + 3f(n-4)$$

$$f(n-2) = f(n-3) + 2f(n-4) + 3f(n-5)$$

$$f(n-3) = f(n-4) + 2f(n-5) + 3f(n-6)$$

d **a** **b** **c**

Notice the pattern?

Question 1: Evaluating Expressions

```
def foo1():
    i = 0
    result = 0
    while i < 10:
        result += i
        i += 1
    return result

foo1()
```

Question 1: Evaluating Expressions

```
def foo1():
    i = 0
    result = 0
    while i < 10:
        result += i
        i += 1
    return result

foo1() # result: 45, i: 10
```

```
        result = 0
i = 0, result = 0
i = 1, result = 1
i = 2, result = 3
...
i = 9, result = 45
i = 10
```

Question 1: Evaluating Expressions

```
def foo2():
    i = 0
    result = 0
    while i < 10:
        if i == 3:
            break
        result += i
        i += 1
    return result

foo2()
```

Question 1: Evaluating Expressions

```
def foo2():
    i = 0
    result = 0
    while i < 10:
        if i == 3:
            break
        result += i
        i += 1
    return result

foo2() # result: 3, i: 3
```

```
        result = 0
i = 0, result = 0
i = 1, result = 1
i = 2, result = 3
i = 3
```

Question 1: Evaluating Expressions

```
def bar1():
    result = 0
    for i in range(0, 10, 1):
        result += i
    return result

bar1()
```

Question 1: Evaluating Expressions

```
def bar1():
    result = 0
    for i in range(0, 10, 1):
        result += i
    return result

bar1() # result: 45, i: 9
```

```
        result = 0
i = 0, result = 0
i = 1, result = 1
i = 2, result = 3
...
i = 9, result = 45
```

Question 1: Evaluating Expressions

```
def bar2():
    result = 0
    for i in range(0, 10, 1):
        if i % 3 == 1:
            continue
        result += i
    return result

bar2()
```

Question 1: Evaluating Expressions

```
def bar2():
    result = 0
    for i in range(0, 10, 1):
        if i % 3 == 1:
            continue
        result += i
    return result

bar2() # result: 33, i: 9
```

```
        result = 0
i = 0, result = 0
i = 1
i = 2, result = 2
i = 3, result = 5
i = 4
i = 5, result = 10
...
i = 8, result = 24
i = 9, result = 33
```

Question 2: num_pairs

Write a function **num_pairs** that takes in **string** and finds number of consecutive character pairs.

```
>>> num_pairs(‘balloon’)
2
>>> num_pairs('missisippi’)
2
```

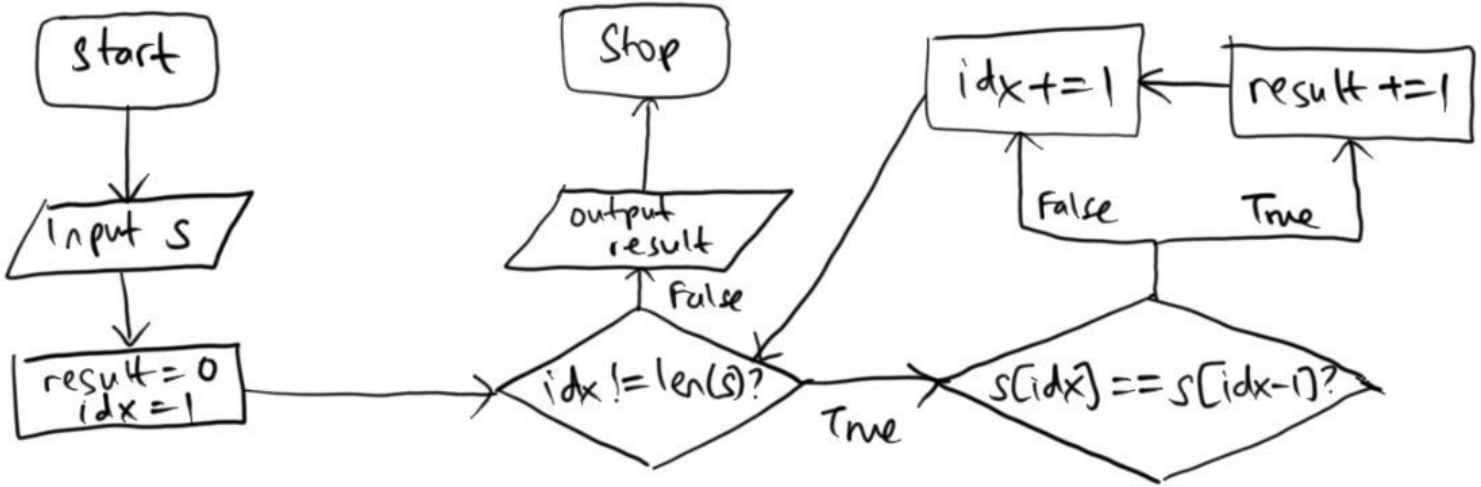
Question 2: num_pairs

Write a function **num_pairs** that takes in **string** and finds number of consecutive character pairs.

```
def num_pairs(s):
    result = 0
    idx = 1
    while idx != len(s):
        if s[idx] == s[idx-1]:
            result += 1
        idx += 1
    return result
```

Question 2: num_pairs

Write a function **num_pairs** that takes in **string** and finds number of consecutive character pairs.



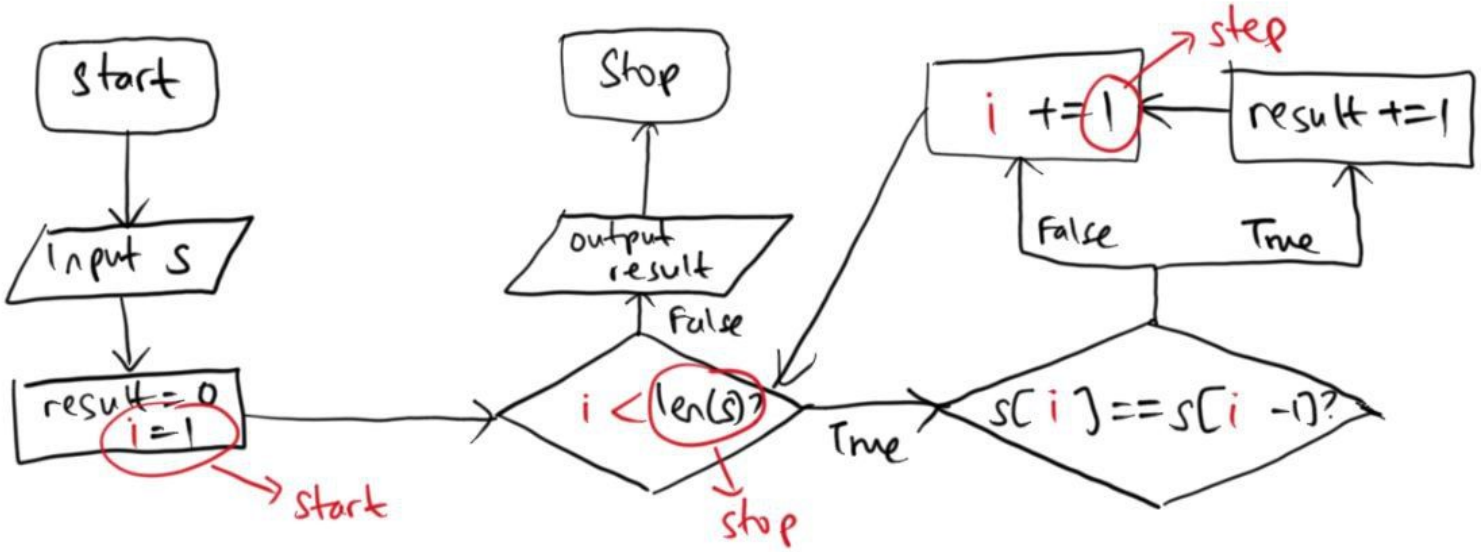
Question 2: num_pairs

Write a function **num_pairs** that takes in **string** and finds number of consecutive character pairs.

```
def num_pairs(s):
    result = 0
    for i in range(1,len(s),1):
        if s[i] == s[i-1]:
            result += 1
    return result
```

Question 2: num_pairs

Write a function **num_pairs** that takes in **string** and finds number of consecutive character pairs.



Question 3: Code Trace

```
count = 0
for i in range(1,6,1):
    if i % 3 > 0:
        count += 3
    if i % 2 == 0:
        continue
    print(count)

while count != 0:
    count //= 2
    if count <= 1:
        break
    print(count)
```

Draw the trace table!!

Question 3: Code Trace

```
count = 0
for i in range(1,6,1):
    if i % 3 > 0:
        count += 3
    if i % 2 == 0:
        continue
    print(count) # line A

while count != 0:
    count //= 2
    if count <= 1:
        break
    print(count)  # line B
```

```
3 # from line A
6 # from line A
12 # from line A
6 # from line B
3 # from line B
```

Extra Questions

```
def infinite_mahh(n):
    i = 0
    while i != n:
        print(i)
        i -= 1
        print(i)
        i += 1

print(infinite_mahh(5))
```

Extra Questions

```
def infinite_mahh(n):
    i = 0
    while i != n:
        print(i)
        i -= 1
        print(i)
        i += 1

print(infinite_mahh(5))
```

```
0
-1
0
-1
0
-1
.
.
.
```

Extra Questions

```
def infinite_mahh(n):
    for i in range(0, n, 1):
        print(i)
        i -= 1
        print(i)

print(infinite_mahh(5))
```

Extra Questions

```
def infinite_mahh(n):
    for i in range(0, n, 1):
        print(i)
        i -= 1
        print(i)

print(infinite_mahh(5))
```

```
0
-1
1
0
2
1
3
2
4
3
None
```

The End