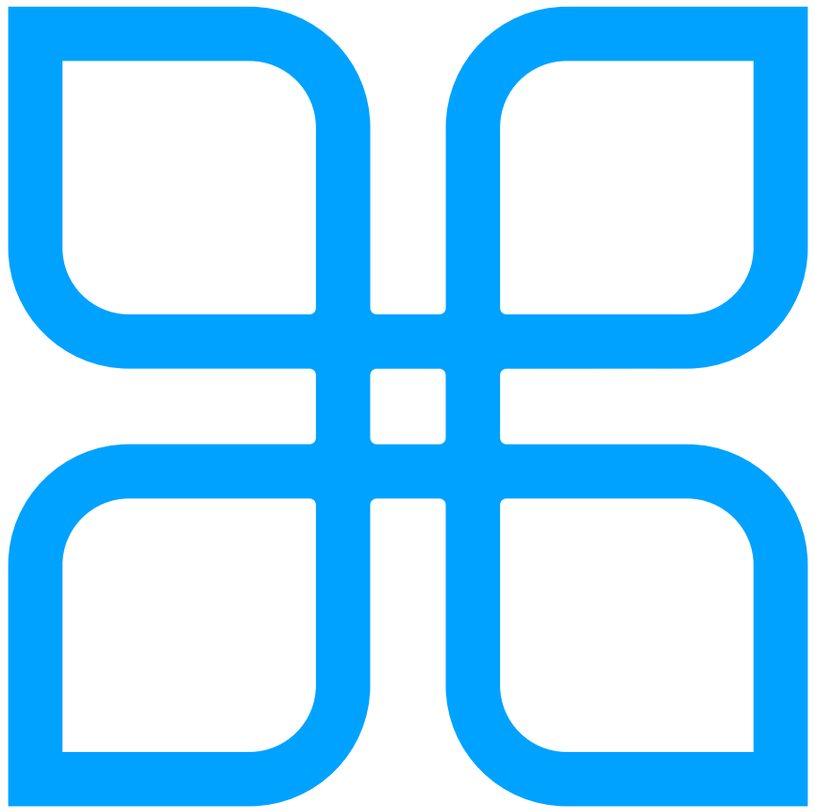
# Comprehension

Thomas Schwarz, SJ



# Loops and Strings

Repetition

# Accessing Elements

- In a slice, the first element is the start
- The second the stop value
- The third the stride, which can be negative

- `lista = [1,3,5,7,9,11,13,15,17,19,21,23,25,27]`
- What is `lista[2:5]`

# Accessing Elements

- In a slice, the first element is the start
- The second the stop value
- The third the stride, which can be negative
  - `lista = [1,3,5,7,9,11,13,15,17,19,21,23,25,27]`
  - What is `lista[2:5]`
    - `[5,7,9]`

# Accessing Elements

- In a slice, the first element is the start
- The second the stop value
- The third the stride, which can be negative
- `lista = [1,3,5,7,9,11,13,15,17,19,21,23,25,27]`
- What is `lista[:5]`
  -

# Accessing Elements

- In a slice, the first element is the start
- The second the stop value
- The third the stride, which can be negative
  - `lista = [1,3,5,7,9,11,13,15,17,19,21,23,25,27]`
  - What is `lista[:5]`
    - `[1, 3, 5, 7, 9]`

# Accessing Elements

- In a slice, the first element is the start
- The second the stop value
- The third the stride, which can be negative
- `lista = [1,3,5,7,9,11,13,15,17,19,21,23,25,27]`
- What is `lista[7:2:-2]`

# Accessing Elements

- In a slice, the first element is the start
- The second the stop value
- The third the stride, which can be negative
  - `lista = [1,3,5,7,9,11,13,15,17,19,21,23,25,27]`
  - What is `lista[7:2:-2]`
    - `[15, 11, 7]`

# Creating Lists

- Create a list of the first 100 numbers $a_i$ defined by
  - $a_0 = 1$
  - $a_1 = 1$
  - $a_i = \sqrt{a_{i-2} \times a_{i-1} + 1}$

# Creating Lists

- Create a list of the first 100 numbers $a_i$ defined by
  - $a_0 = 1$
  - $a_1 = 1$
  - $a_i = \sqrt{a_{i-1} \times a_i + 1}$
- Need to create a list, that starts with [1,1]

# Creating Lists

- Create a list of the first 100 numbers $a_i$ defined by
  - $a_0 = 1$
  - $a_1 = 1$
  - $a_i = \sqrt{a_{i-1} \times a_i + 1}$
- Need to create a list, that starts with [1,1]
- Then we add to the list the latest element

# Creating Lists

- Create a list of the first 100 numbers $a_i$ defined by
  - $a_0 = 1$
  - $a_1 = 1$
  - $a_i = \sqrt{a_{i-1} \times a_i + 1}$
- Need to create a list, that starts with [1,1]
- Then we add to the list the latest element
  - Which we calculate using the last and second-last element in the list

# Creating Lists

- Create a list of the first 100 numbers $a_i$ defined by
  - $a_0 = 1$
  - $a_1 = 1$
  - $a_i = \sqrt{a_{i-2} \times a_{i-1} + 1}$
- Need to create a list, that starts with [1,1]
- Then we add to the list the latest element
  - Which we calculate using the last and second-last element in the list

```
import math
lista = [1,1]
for i in range(2,101):
    lista.append(math.sqrt(lista[-1]*lista[-2]+1))
```

# Processing Strings

- Write a function that tests whether the string has the pattern 'aba' in it.

```
def aba(astring):
    return 'aba' in astring
```

# Processing Strings

- Write a function that takes a string as an input. It then replaces all occurrences of 'ss' with a '3'

# Processing Strings

- Write a function that takes a string as an input. It then replaces all occurrences of 'ss' with a '3'

```
def ss3(astring):
    result = [ ]
    for letter in astring:
        …
    return ''.join(result)
```

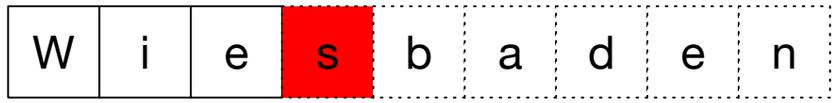
- When we do not see an 's': we just copy it
- If we see an 's', it depends on what the next letter is.
  - We can set a Boolean *flag*
  - But we do not copy yet

# Processing Strings

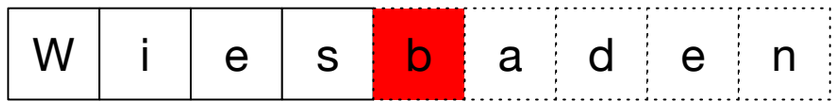

Copy:


Copy:

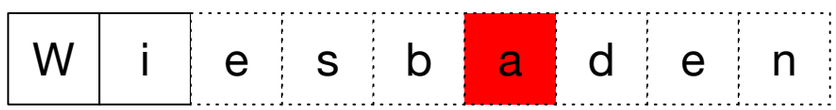

Copy:




Delay action: we can not decide



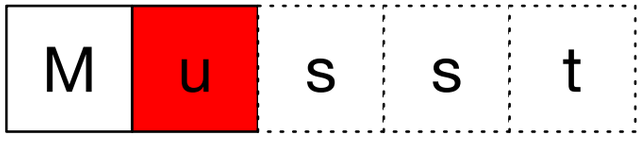

Now we know what to do: copy both letters



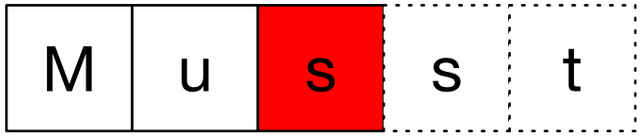

Copy:

# Processing Strings

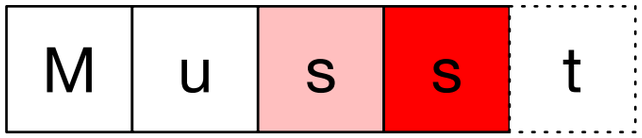

Copy



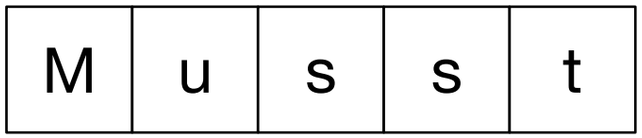

Copy




Delay action, but remember that we saw an ’s’:




Because we just saw an ’s’, push ‘3’




Copy

# Processing Strings

- If we see an 's', it depends on what the next letter is.
  - We can set a Boolean *flag* if this is the case
  - If we have seen an 's' but the current letter is not an 's', then put in an 's' and the current letter
  - If we have just seen an 's' and the current letter is an 's', then we put a '3'

# Processing Strings

| | just seen s | not just seen s |
|---|---|---|
| **letter is s** | push '3' | remember to have seen s |
| letter is not s | push s, then the letter | remember to not have seen s, push letter |

# Processing Strings

```
def ss3(astring):
    result = []
    seen_S = False
    for letter in astring:
        if letter == 's' and seen_S:
            result.append('3')
            seen_S = False
        elif letter == 's' and not seen_S:
            seen_S = True
        elif letter != 's' and seen_S:
            result.append('s')
            result.append(letter)
            seen_S = False
        elif letter != 's' and not seen_S:
            seen_S = False
            result.append(letter)
    return ''.join(result)
```