i116: Basic of Programming

1. Some program constructs (1)

Kazuhiro Ogata, Canh Minh Do

Roadmap

- What is programming?
- Comments
- Variables and assignments
- **while** (loop) statements
- **for** (loop) statements
- **if** (conditional) statements

What is programming?

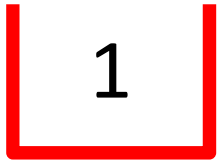
- To write documents called *programs* in a language called *a programming language*, such as Java, C and Python.
- In a program, we write *what* we would like a computer to do, such as calculation of approximate values of $\sqrt{2}$ and π and making it possible to create cool presentation slides as the ones I am using.

What is programming?

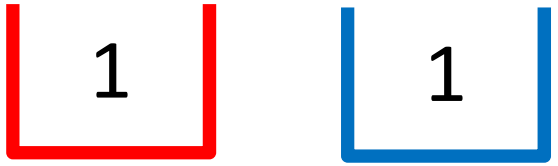
- Not enough usually just to write *what* we would like a computer to do.
- Necessary to write *how* a computer does what we would like to do

What is programming?

Put 1 in the red box



Put 1 in the blue box

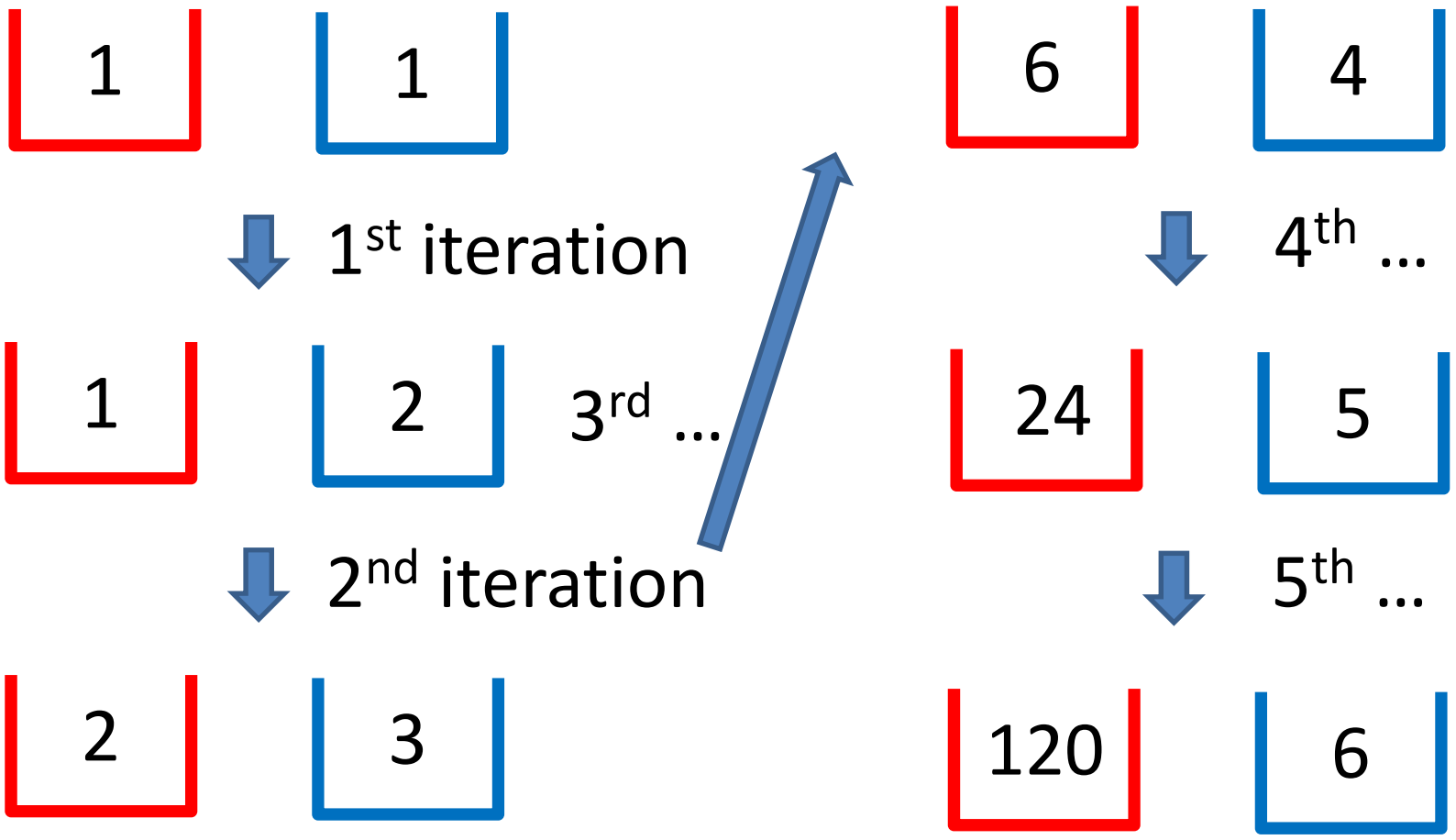


Do the following while the value in the blue box is less than or equal to 5:

(1) Extract the values from the red and blue boxes, multiply the values and put the result in the red box

(2) Extract the value from the blue box, increment the value and put the result in the blue box

What is programming?



What is programming?

…

5^{th} …

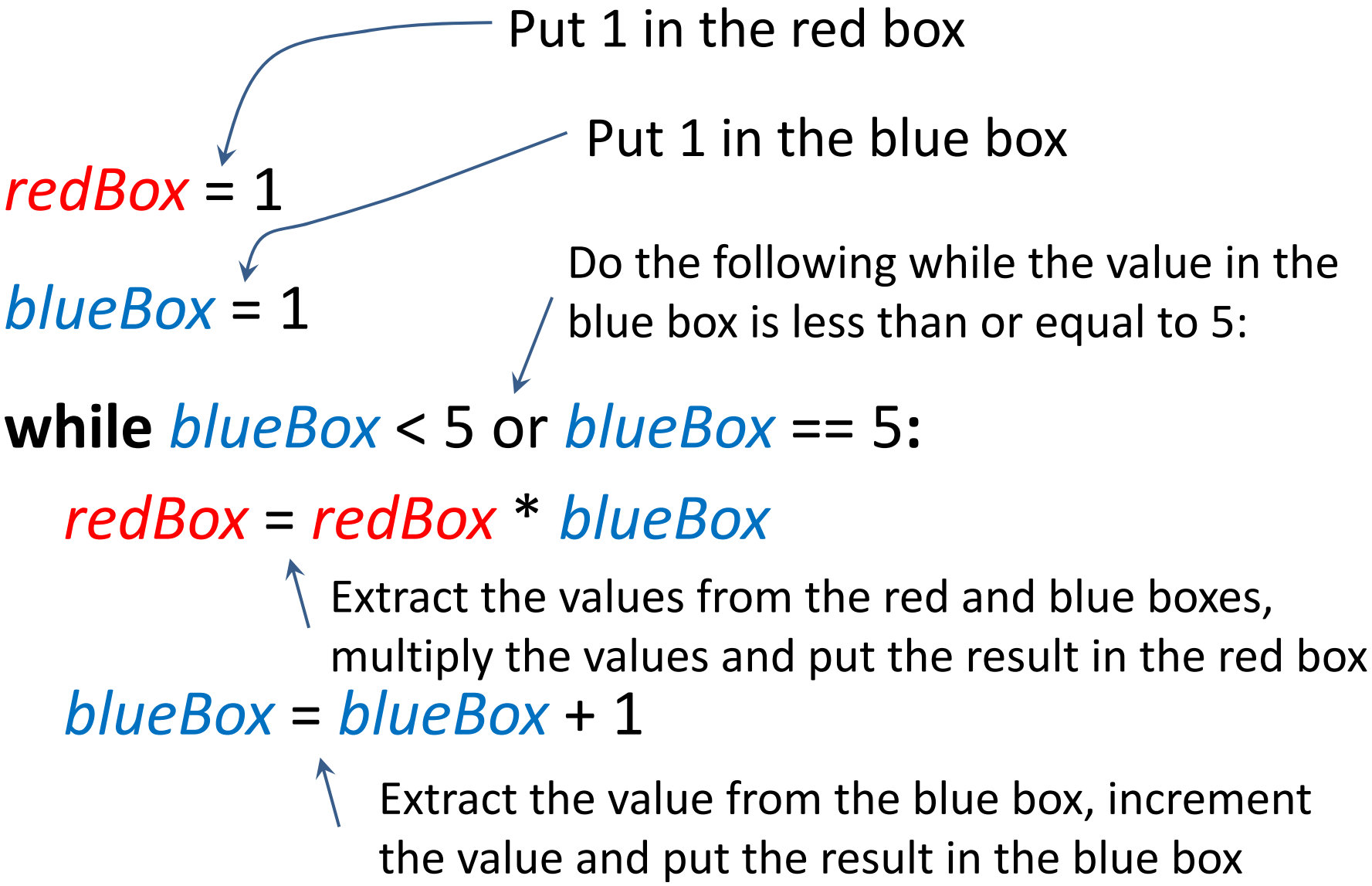
Because the value in the blue box is not less than or equal to 5, we stop here.
What do we get in the red box?
120, which equals to 5!.
This is one possible way to calculate 5!.

What is programming?

What we have just seen can be written as a program in a programming language!

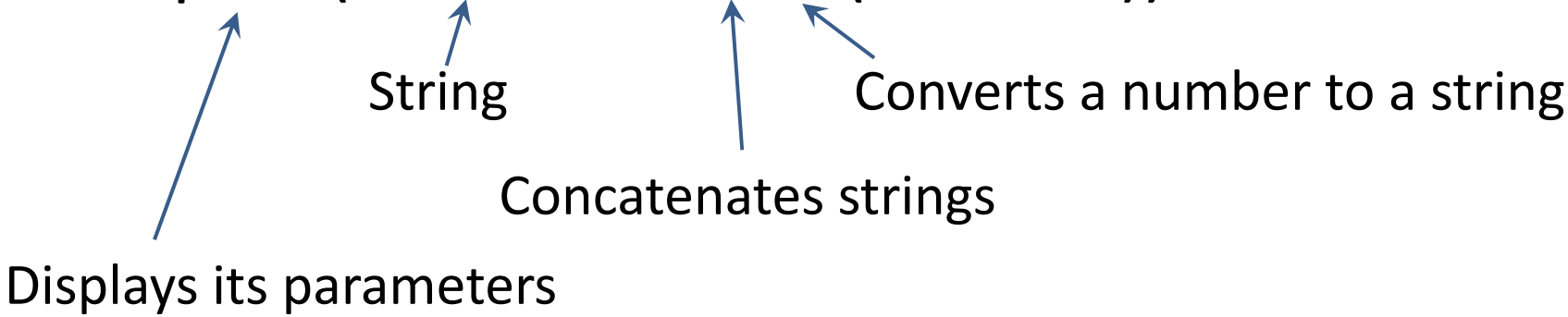
```
redBox = 1
blueBox = 1
while blueBox < 5 or blueBox == 5:
    redBox = redBox * blueBox
    blueBox = blueBox + 1
```

What is programming?



What is programming?

```
redBox = 1
blueBox = 1
while blueBox < 5 or blueBox == 5:
    redBox = redBox * blueBox
    blueBox = blueBox + 1
print('redBox: ' + str(redBox))
print('blueBox: ' + str(blueBox))
```



Comments

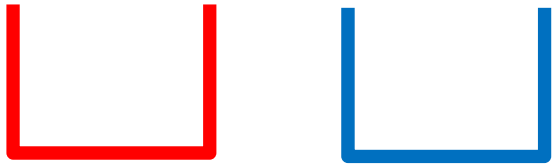
- A comment starts with # (inclusive) and ends with the end of the line.
- A segment (that can be multiple lines) enclosed with """ and """ is also a comment.
- Comments are just ignored by Python, but very important for humans, letting them know something about programs.

Comments

```
"""
A program that calculates 5! with loop and stores its result in
redBox when it terminates
"""
redBox = 1 # used to store the result of 5!
blueBox = 1 # used as a loop variable
while blueBox < 5 or blueBox == 5:
    """
    The two assignments are done while blueBox
    is less than or equal to 5.
    """
    redBox = redBox * blueBox # Store the product in redBox
    blueBox = blueBox + 1 # Increment the loop var
```

Variables and assignments

Variables are like boxes, such as *redBox* and *blueBox*.



We can put, store, save, etc. something (or a value) in a variable (or a box).

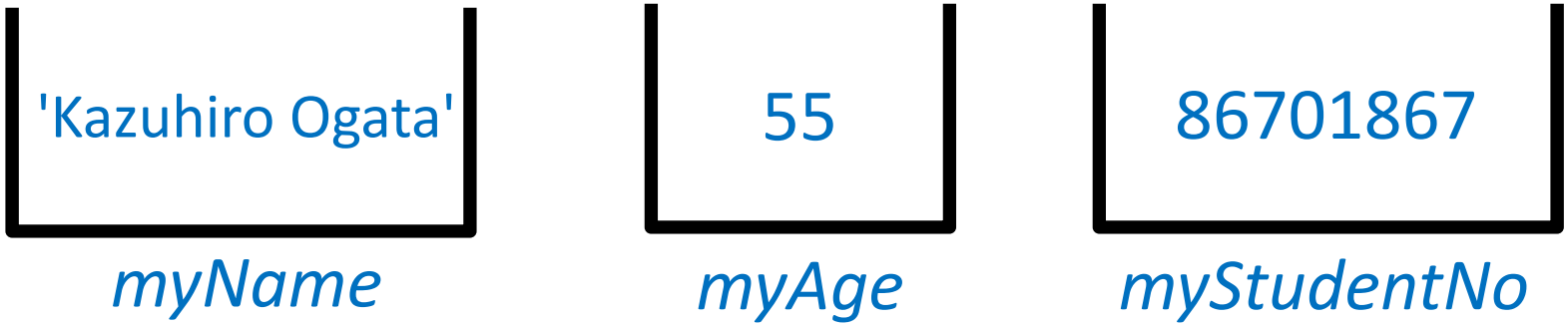
A variable is given its *name*, such as *redBox* and *blueBox*.

Let us consider some variables whose names are *myName, myAge* and *myStudentNo*.

Variables and assignments

An assignment (or an assignment statement) puts or stores a value in a variable.

myName = 'Kazuhiro Ogata'
myAge = 56
myStudentNo = 86701867



Variables and assignments

```
myName = 'Kazuhiro Ogata'
myAge = 55
myStudentNo = 86701867
print('My name is ' + myName + '.')
print('I am ' + str(myAge) + ' years old.')
print('My student number is ' + str(myStudentNo) + '.')
```

while (loop) statements

While *condition* is true, what is written in *block* (or *fragment*) is repeatedly done.

The spaces or tabs (namely *indentations*) are extremely important.

while (loop) statements

```
redBox = 1
blueBox = 1
while blueBox < 5 or blueBox == 5:
    redBox = redBox * blueBox
    blueBox = blueBox + 1
```

The two programs are totally different!

```
redBox = 1
blueBox = 1
while blueBox < 5 or blueBox == 5:
    redBox = redBox * blueBox
blueBox = blueBox + 1
```

for (loop) statements

for *variable* **in** *ordered collection* **:**

    block (fragment)

A string, such as 'JAIST', is an example of *ordered collection*.

Let *N* be the number of elements in *ordered collection*.

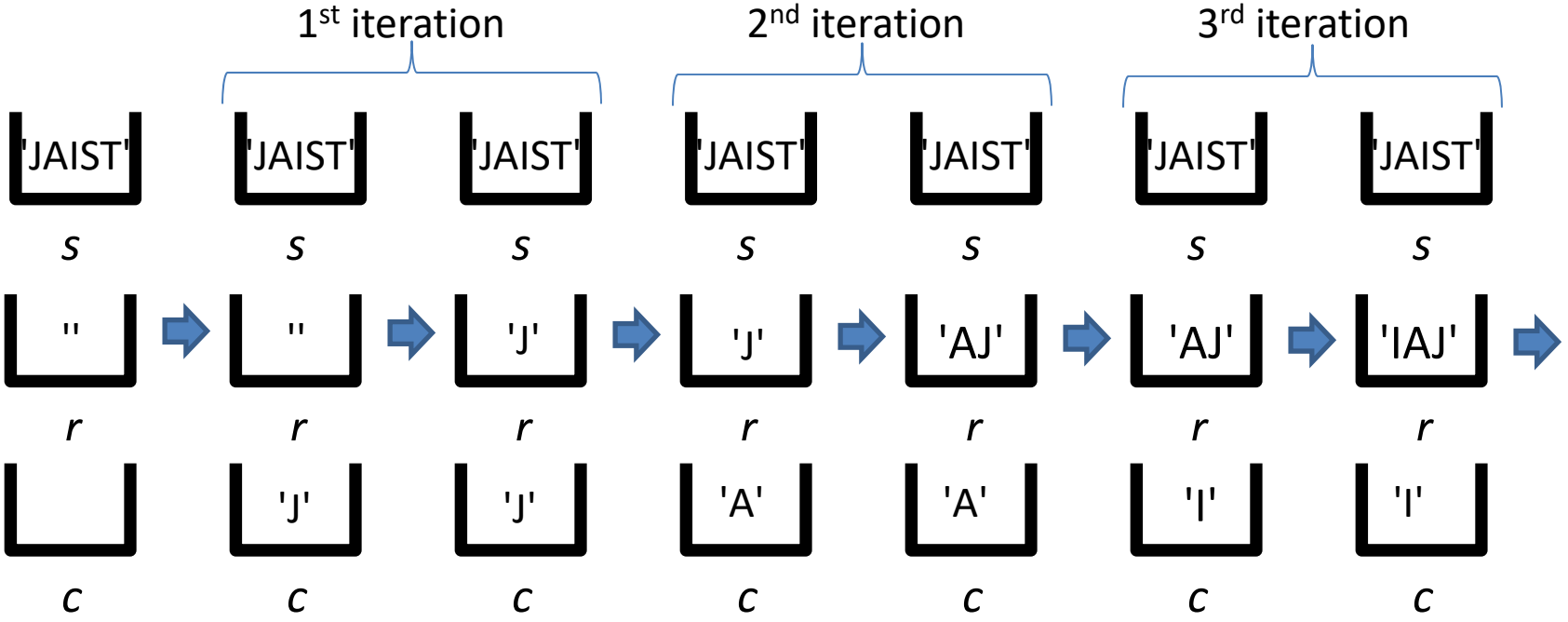
What is written in *block* is done repeatedly *N* times.

In the i^{th} iteration, the i^{th} element is put in *variable*.

variable can be used in *block (fragment)*.

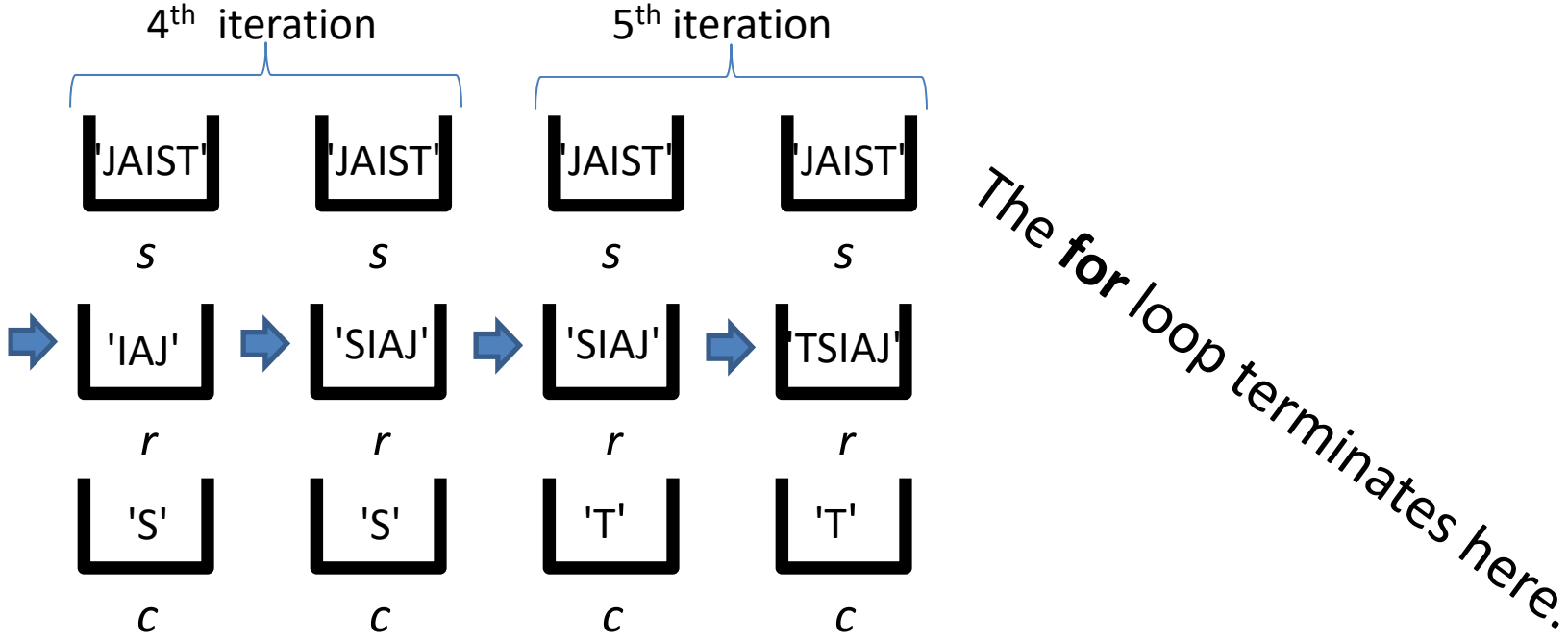
for (loop) statements

```
s = 'JAIST'
r = ''
for c in s:
    r = c + r
print('The reverse of ' + s + ' is ' + r + '.')
```



for (loop) statements

```
s = 'JAIST'
r = ''
for c in s:
    r = c + r
print('The reverse of ' + s + ' is ' + r + '.')
```



for (loop) statements

```
s = 'JAIST'
r = ''
for c in s:
    r = c + r
print('The reverse of ' + s + ' is ' + r + '.')
```

The two programs are different!

```
s = 'JAIST'
r = ''
for c in s:
    r = c + r
    print('The reverse of ' + s + ' is ' + r + '.')
```

if (conditional) statements

If *condition* is true, what is written in *block* (or *fragment*) is done.

The spaces or tabs (namely *indentations*) are extremely important.

if (conditional) statements

```
v0 = 100
for i in range(v0):
    if i * i > v0:
        v1 = i - 1
        break
print('sr(', v0, ') = ', v1)
```

The **for** loop is repeated *v*0+1 times: *i* = 0 in the 1st iteration, *i* = 1 in the 2nd iteration, and *i* = *v*0+1 in the final iteration.

The loop exits whenever **break** is encountered.

The program computes the positive integral part of $\sqrt{100}$.

if (conditional) statements

If *condition* is true, what is written in *block*1 (or *fragment*1) is done; otherwise what is written in *block*2 (or *fragment*2) is done

```
if   condition :
        block1 (fragment1)
else:
        block2 (fragment2)
```

if (conditional) statements

```
v0 = 100
v1 = 0
v2 = v0
while v1 != v2:
  if (v2-v1)%2 == 0:
    v3 = v1+(v2-v1)//2
  else:
    v3 = v1+(v2-v1)//2+1
  if v3*v3 > v0:
    v2 = v3-1
  else:
    v1 = v3
print('sr(', v0, ') = ', v1)
```

Another program that computes the positive integral part of $\sqrt{100}$.

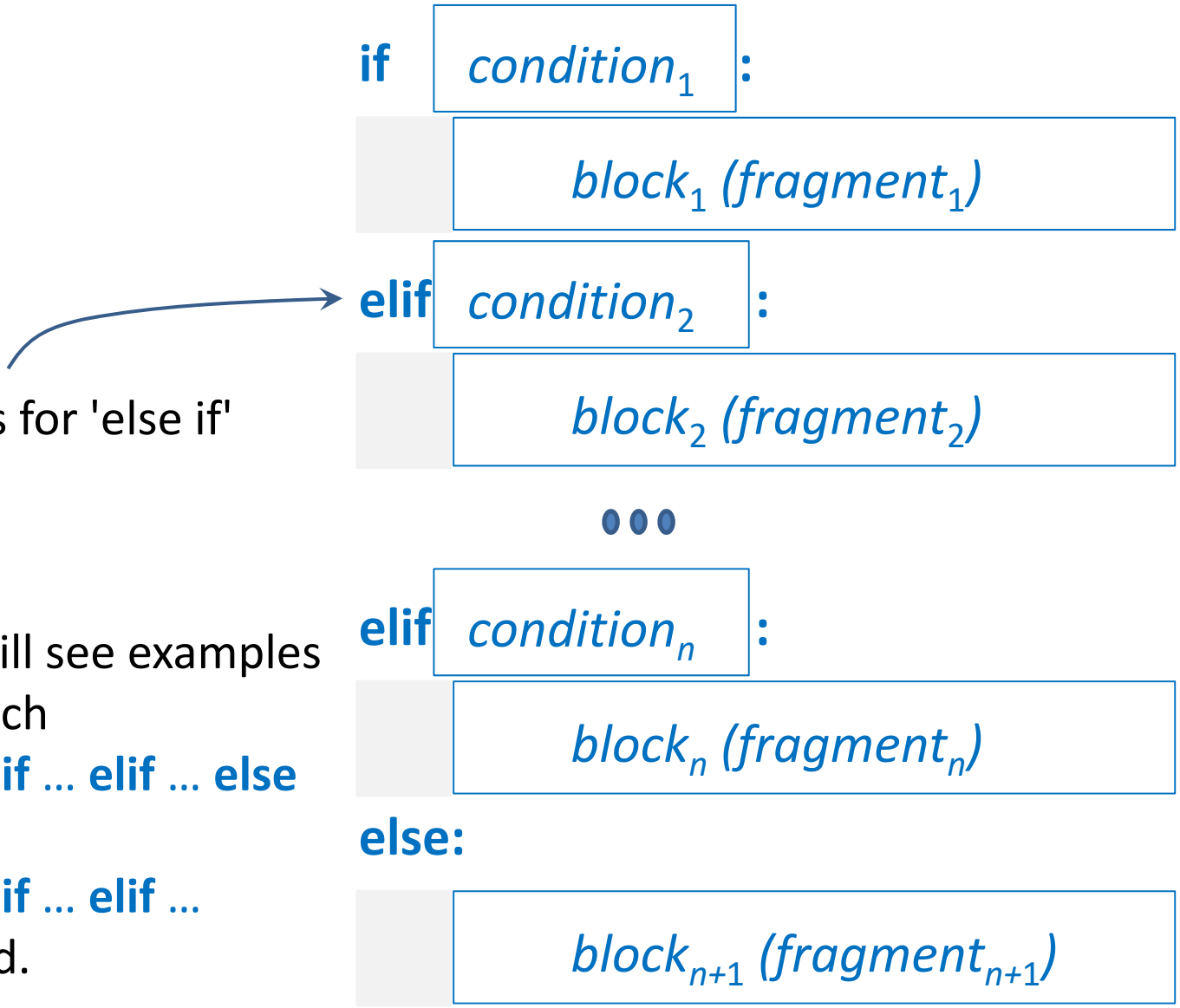
Let *a* & *b* be integers.
a//*b* returns an integer (the quotient of the division of *a* by *b*).
a/*b* returns a floating-point number.

if (conditional) statements

```
v0 = 20000000000000000
for i in range(v0):
    if i * i > v0:
        v1 = i - 1
        break
print('sr(', v0, ') = ', v1)
```

```
v0 = 20000000000000000
v1 = 0
v2 = v0
while v1 != v2:
  if (v2-v1)%2 == 0:
    v3 = v1+(v2-v1)//2
  else:
    v3 = v1+(v2-v1)//2+1
  if v3*v3 > v0:
    v2 = v3-1
  else:
    v1 = v3
print('sr(', v0, ') = ', v1)
```

if (conditional) statements



Stands for 'else if'

You will see examples in which
if … elif … elif … else
or
if … elif … elif …
is used.