# Sum Program from last week rewritten

In [1]:

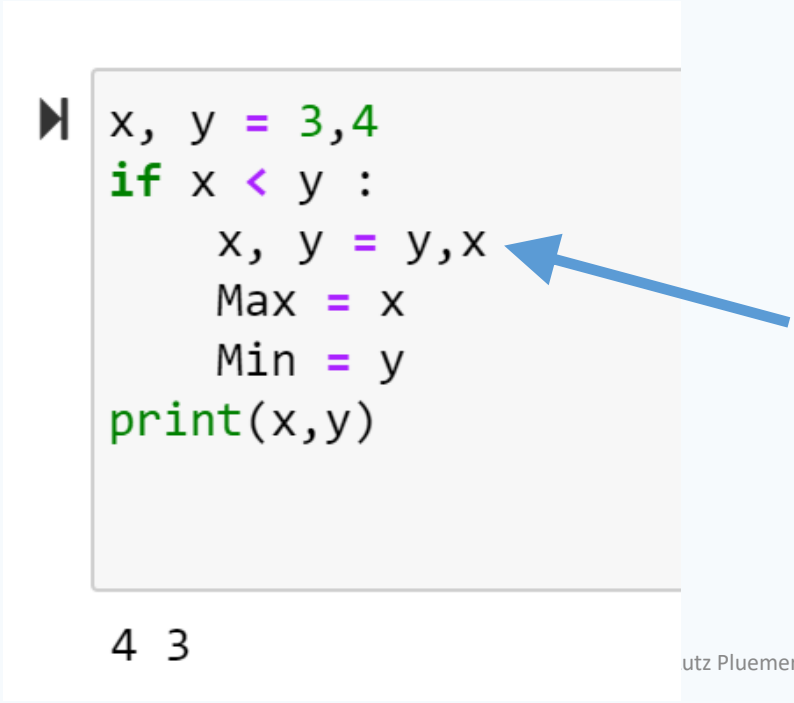
```
i = 0
result = 0
n = 5
while i <= n :
    result += i
    i += 1
print(result)
```

```
15
```

In [ ]:

# Multiple Assignments in One Line

- You can assign several variables in one line, such as
- a, b, c = 100
- **x, y = y, x**
- The latter swaps two variables
- so you have a shorter way to find the maximum or minimum



# The GCD Algorithm in Jupyter

In [2]:

```
x = 1071
y = 462
print("Greatest Common Divisor of " + str(x) + " and " + str(y) + "  is:   ")

# The GCD Algorithm

while y > 0  :
    x, y = y, x % y

# This was already the GCD algorithm

print(x)
```

```
Greatest Common Divisor of 1071 and 462  is:
21
```

Try to **understand why** it works and **how** it works

For that you can for instance insert a **print** command

# This is how it works

In [4]:

```
x = 1071
y = 462


# The GCD Algorithm

while y > 0  :
    x, y = y, x % y
    print(x,y)

# This was already the GCD algorithm

print(x)
```

```
462 147
147 21
21 0
21
```

# More about Strings

- Use quotation marks to define a string:
  welcome = “Good Morning to everybody”
- You can concatenate strings with **+**
- str1 = “Good Morning ”
- str2 = “to everybody”
- str3 = str1 + str2
- You can even multiply, try **3 * str1**
- **You** can use 漢字:
- And you can insert
  **line breaks**

```
GoodMorning = "早上好"
print(GoodMorning)
```

```
早上好
```

```
WelcomeToMyLecture = "欢迎来到我的讲座"
print(WelcomeToMyLecture)
```

```
欢迎来到我的讲座
```

```
print("this is a \n\rcarriage return")
```

```
this is a
carriage return
```

# Lists

- Lists are sequences of **arbitrary length** with elements of **arbitrary** type
- Lists are defined by square brackets
- list = [1, 2, 3, 4, 6]

```
list = [1, 2, 3, 4, 6]
print(list)
```

```
[1, 2, 3, 4, 6]
```

- As for strings, you can **concatenate** lists with **+** and **multiply** lists as well , such as **3 * list**

```
list = [1, 2, 3.0, 4, 6]
print(list)
```

```
[1, 2, 3.0, 4, 6]
```

- You can access single elements by list[3]

```
list = [1, 2, 3.0, 4, "good morning"]
print(list)
```

```
[1, 2, 3.0, 4, 'good morning']
```

# Tuples

- Tuples are very similar to Lists, with the following differences:
- Tuples are defined by parentheses, such as
  a = (1, 2, 3)
- You can access single elements, but you cannot change their value
- The nice thing with tuples is **packing** and **unpacking**
- **date** = 2023, April, 2
- (year,  month, day) = **date**
- **Swap** variable values easily

```
date = 2023, 4, 2
(year,  month, day) = date
```

```
print(year)
```

```
2023
```

```
a, b = 10 , 20
a, b = b, a
print(a,b)
```

```
20 10
```

# The range Generator

- range generates an ordered sequence of numbers, in a given range
- Range(10) generates 0, 1, 2 … 9
- Range(5,10) generates 5, 6, 7, 8, 9
- Range(2,10,2) generates 2, 4, 6, 8, 10
- Format: *range(**start**, **stop**, **stepsize**)*
  you can **omit** ***start*** and ***stepsize***
- Note that it stops **before stop**, so best write range(**1,n+1**) if you want to get the numbers between 1 and n
- The most popular application for range is the for - loop

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

```
0
1
2
3
4
```

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

```
1
3
5
7
9
```

# The for-loop in Python

We restrict to the most simple variant, there are others

For **<var>** in <**iterable**> **:**

**<statement(s)>**

**var** is the name of a variable

**Iterable** can be range,
a list or a tuple

**Statement(s)** is a sequence
of indented statements

Syntax: don't forget the colon,
don't forget the indentation

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

```
0
1
2
3
4
```

```
for i in [1, 2, 3] :
    print(i)
```

```
1
2
3
```

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

```
1
2
3
```

# The Sum formula as a for-loop

- Note that **`range(1,n+1,1)`** counts from **`1 to n`** with **`stepsize 1`**
- You need to initialize **`sum`** before the loop, otherwise you cannot write **`sum = sum + 1`** the right side would be undefined

```
n = 5
sum = 0
for i in range(1,n + 1, 1) :
    sum = sum + i
print(sum)
```

```
15
```

# Classroom Exercises 5

1. Go to the Classroom Exercises Slides and run them in your Jupyter
2. Generate a List of the even numbers up to 32 and print them with a for-loop
3. Generate a List of the numbers divisible by 3 up to 33 and print them with a for loop
4. Write a program for factorial using the for-loop .

# Homework Exercises 5

- Write a program with for-lop for factorial, using the += operator
- Write a program which sum of those numbers between (and including) 1 and 60 which are divisible by 3
- Try to understand the GCD algorithm better, why and how it works. For that, include one or two print statements in the while-loop
- Instead of range, you can also use a list in a for-loop. Having this in mind:
  Calculate the products of these two lists with for-loops:
  [2, 3, 5, 7, 11, 13, 17, 19, 23] and
  [13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67]
- Star Exercise: Design a formula and a program which uses the GCD to calculate the sum of two fractions $\frac{a}{b} + \frac{c}{d}$