Ghent University
Faculty of Sciences

Python Programming

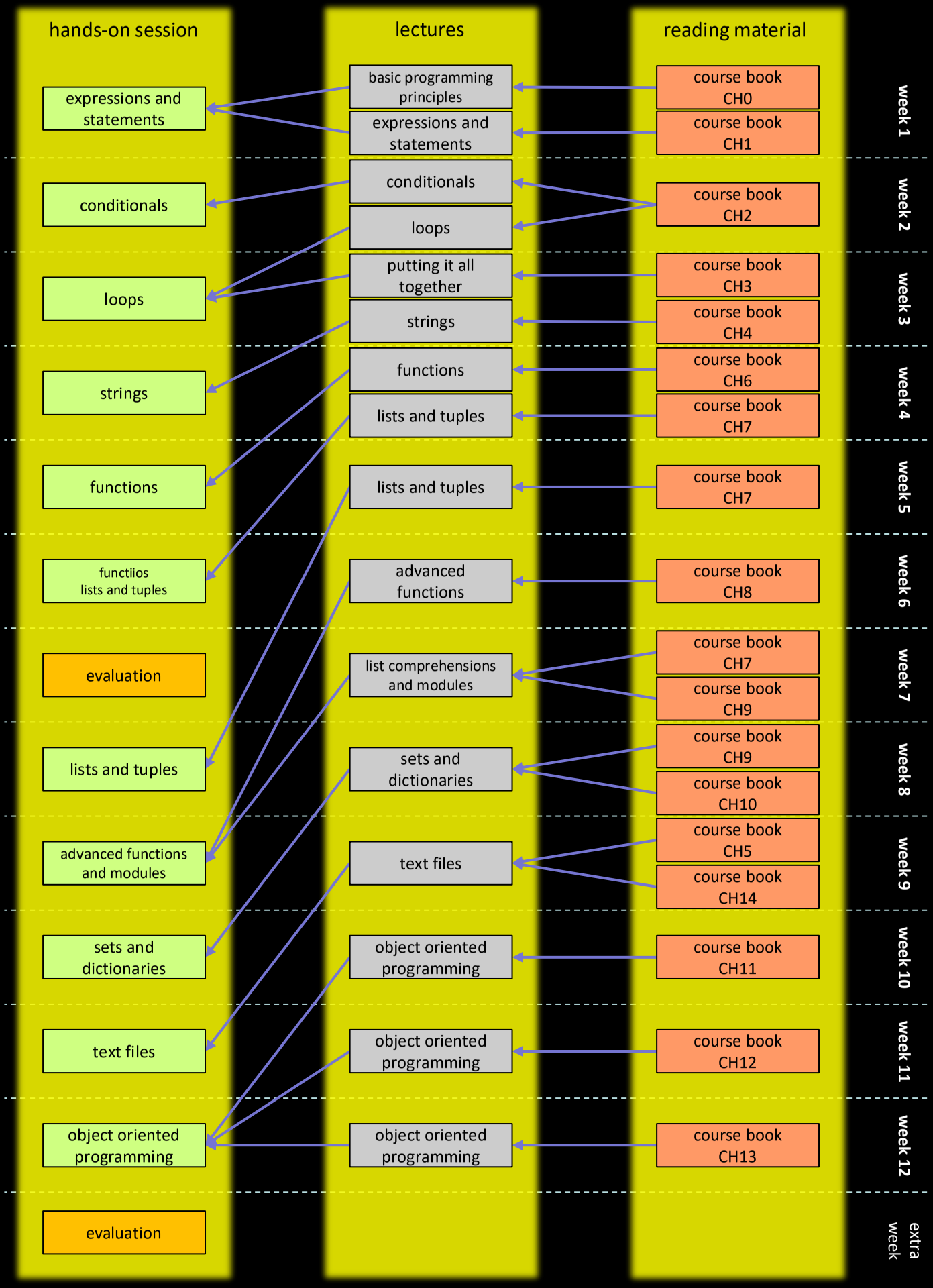
putting it all together

Prof. Dr. Peter Dawyndt

peter.dawyndt@ugent.be

@dawyndt

UNIVERSITEIT GENT


hands-on session
lectures
reading material
week 1
week 2
week 3
week 4
week 5
week 6
week 7
week 8
week 9
week 10
week 11
week 12
extra week
expressions and statements
conditionals
loops
strings
functions
functiios lists and tuples
evaluation
lists and tuples
advanced functions and modules
sets and dictionaries
text files
object oriented programming
evaluation
basic programming principles
expressions and statements
conditionals
loops
putting it all together
strings
functions
lists and tuples
lists and tuples
advanced functions
list comprehensions and modules
sets and dictionaries
text files
object oriented programming
object oriented programming
object oriented programming
course book CH0
course book CH1
course book CH2
course book CH3
course book CH4
course book CH6
course book CH7
course book CH7
course book CH8
course book CH7
course book CH9
course book CH9
course book CH10
course book CH5
course book CH14
course book CH11
course book CH12
course book CH13


DODONA

What day is your birthday?

socrative

socrative.com

room: 407122

format: DD/MM

= two digits for day and month

= no year of birth

Example

I was born on December 5, 1975.

answer: 05/12

socrative

socrative.com

room: 407122

What is the probability that in this group of XX students, at least two students have their birthdays on the same day?

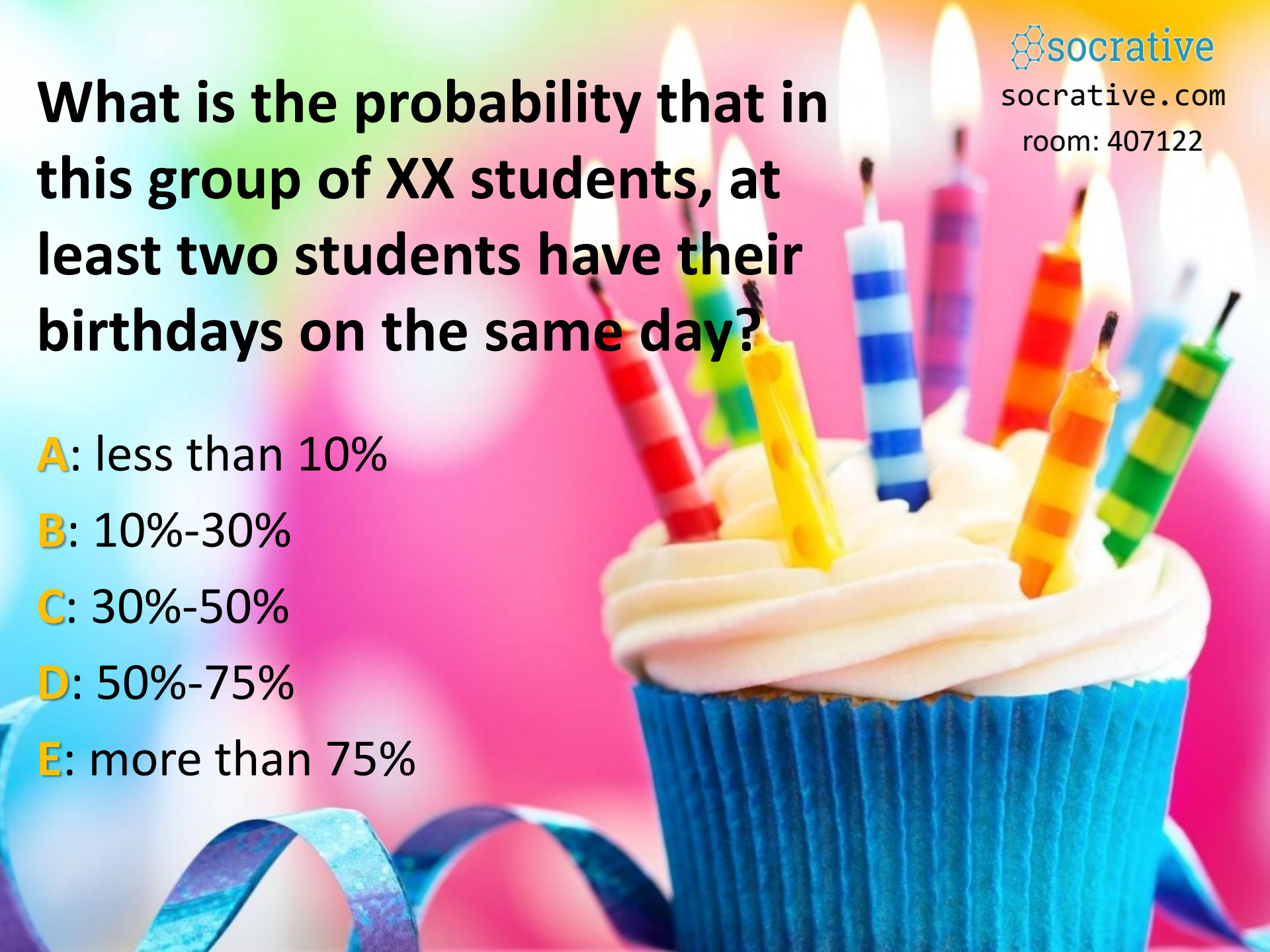
A: less than 10%

B: 10%-30%

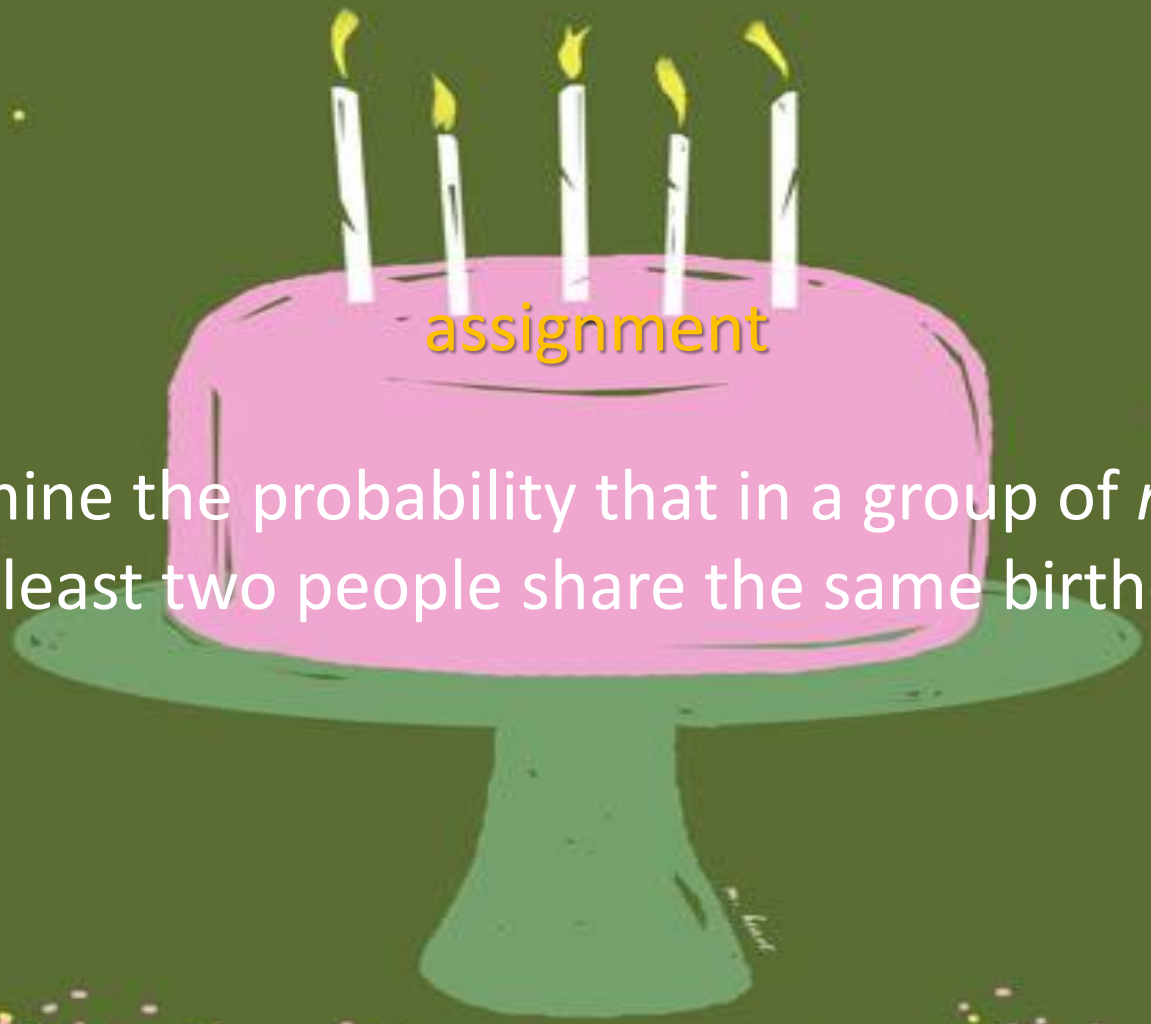
C: 30%-50%

D: 50%-75%

E: more than 75%



the birthday paradox



assignment

Determine the probability that in a group of *n* people at least two people share the same birthday.

Birthday paradox

$$q_n = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365 - n + 1}{365}$$

paradox1.py

```
for n in range(5, 76, 5):
    # compute probability
    qn = 1.0
    for m in range(n):
        qn *= (365 - m) / 365       # floating point division

    # print probability
    print(n, 1 - qn)
```

UNIVERSITEIT GENT

Birthday paradox

$$q_n = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365 - n + 1}{365}$$

paradox2.py

```
qn = 1.0
for n in range(2, 76):
    # compute probability
    qn *= (365 - n + 1) / 365       # floating point division

    # print probability
    if not n % 5:
        print(n, 1 - qn)
```

UNIVERSITEIT GENT

Birthday paradox

$$q_n = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365 - n + 1}{365}$$

paradox_graphical.py

```
import pylab
# construct graphical representation
x, y, qn = [], [], 1.0
for n in range(365):
    # compute probability
    qn *= (365 - n) / 365          # floating point division

    # add sample point
    x.append(n + 1)                # x-value of sample point
    y.append(1 - qn)               # y-value of sample point

# show graphical representation
pylab.plot(x, y, 'b-')             # plot trend
pylab.xlim([0, 365])               # fix limits of x-axis
pylab.show()                       # show plot
```

UNIVERSITEIT GENT

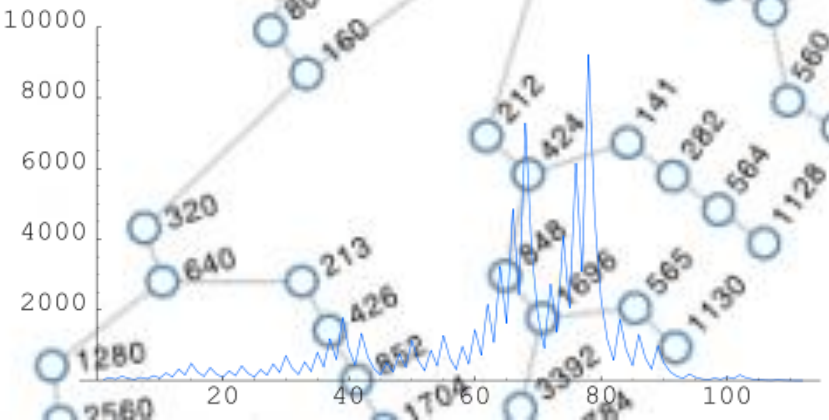
Collatz Conjecture

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ 3n+1 & \text{if } n \text{ is odd} \end{cases}$$

hailstone series:

$$a_i = \begin{cases} n & \text{voor } i = 0 \\ f(a_{i-1}) & \text{voor } i > 0 \end{cases}$$

27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 890, 445, 1336, 668, 334, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, **9232**, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, **1**



Collatz Conjecture

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ 3n+1 & \text{if } n \text{ is odd} \end{cases}$$

hailstone series:

$$a_i = \begin{cases} n & \text{voor } i = 0 \\ f(a_{i-1}) & \text{voor } i > 0 \end{cases}$$

collatz.py

```
cases = int(input())
for case in range(cases):
    # read start value and initialize cycle length
    n, cyclelength = int(input()), 1
    # determine length of hailstone series
    while n != 1:
        if n % 2:                # n odd
            n = 3 * n + 1
        else:                    # n even
            n //= 2
        cyclelength += 1
    # print length of hailstone series
    print(cyclelength)
```

UNIVERSITEIT GENT

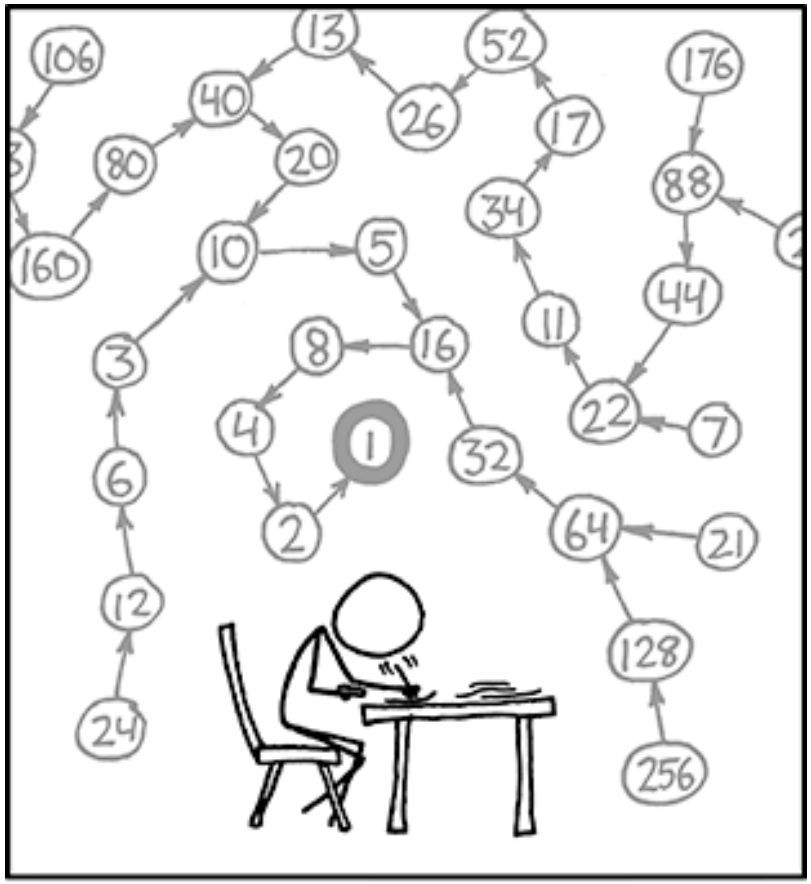
Collatz Conjecture

collatz_graphical.py


```
m = int(input('Enter a value for n: '))

# determine hailstone series
n = m
hailstone = [n]
while n != 1:
    if n % 2:                # n odd
        n = 3 * n + 1
    else:                    # n even
        n //= 2
    hailstone.append(n)

# graphical representation of hailstone series
import pylab
pylab.plot(range(1, len(hailstone) + 1), hailstone)
pylab.title(f'hailstone series for $n = {m}$')
pylab.xlim(1, len(hailstone))
pylab.xlabel('$i$')
pylab.ylabel('$a_i$')
pylab.show()
```

UNIVERSITEIT GENT

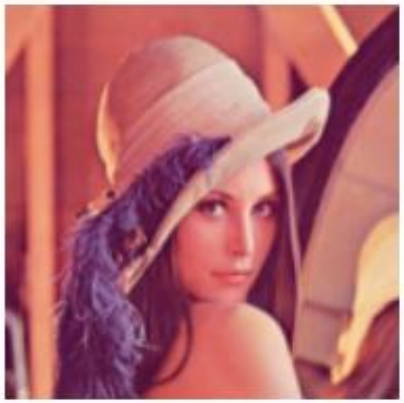
What's the use ?



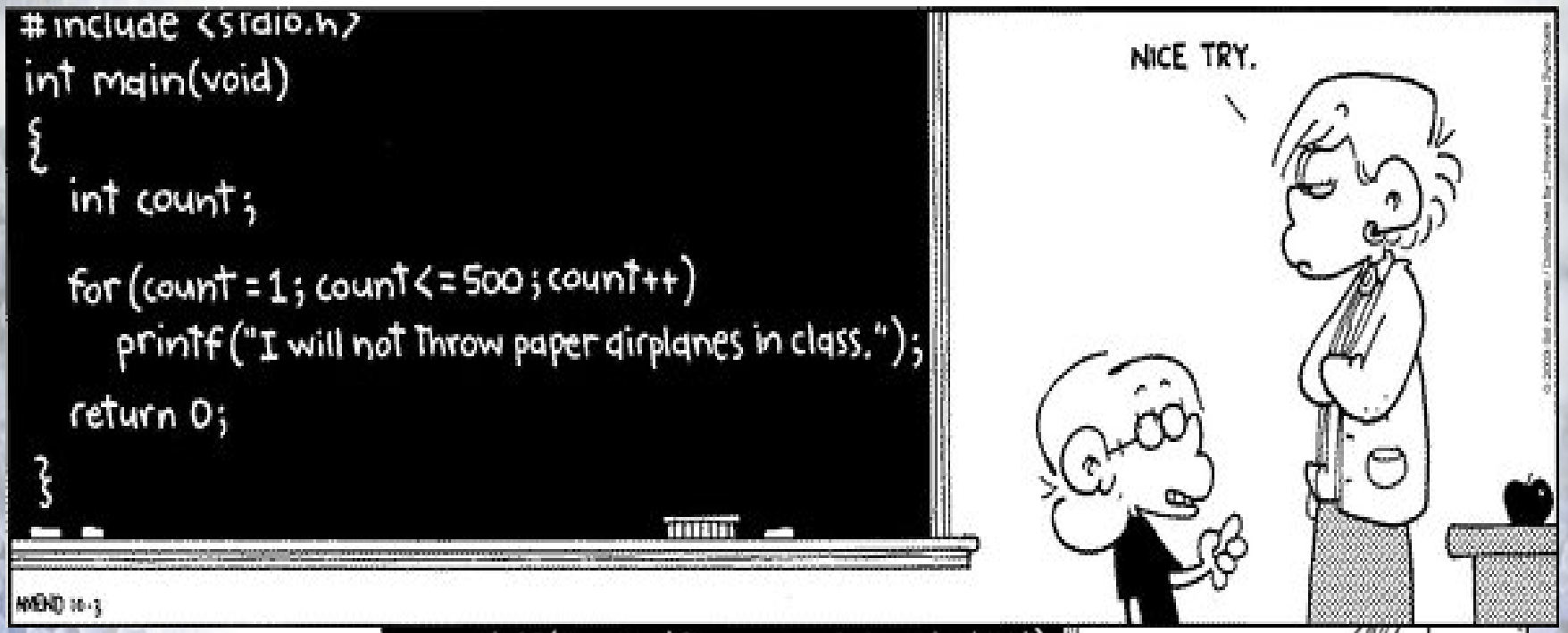
THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

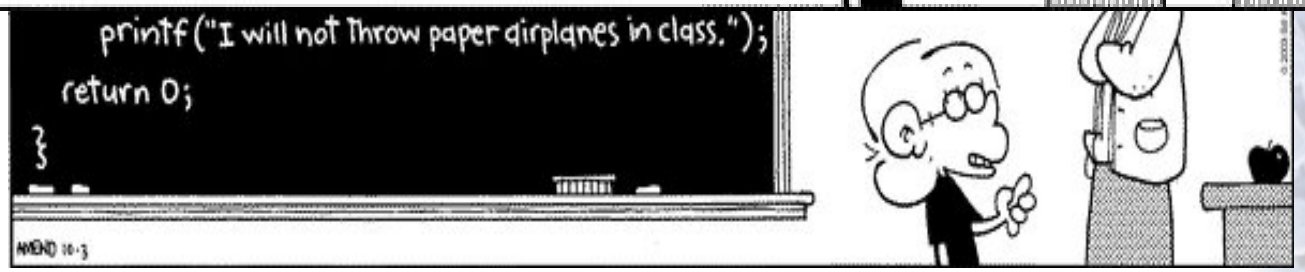
UNIVERSITEIT GENT

Matplotlib gallery



For or while ?

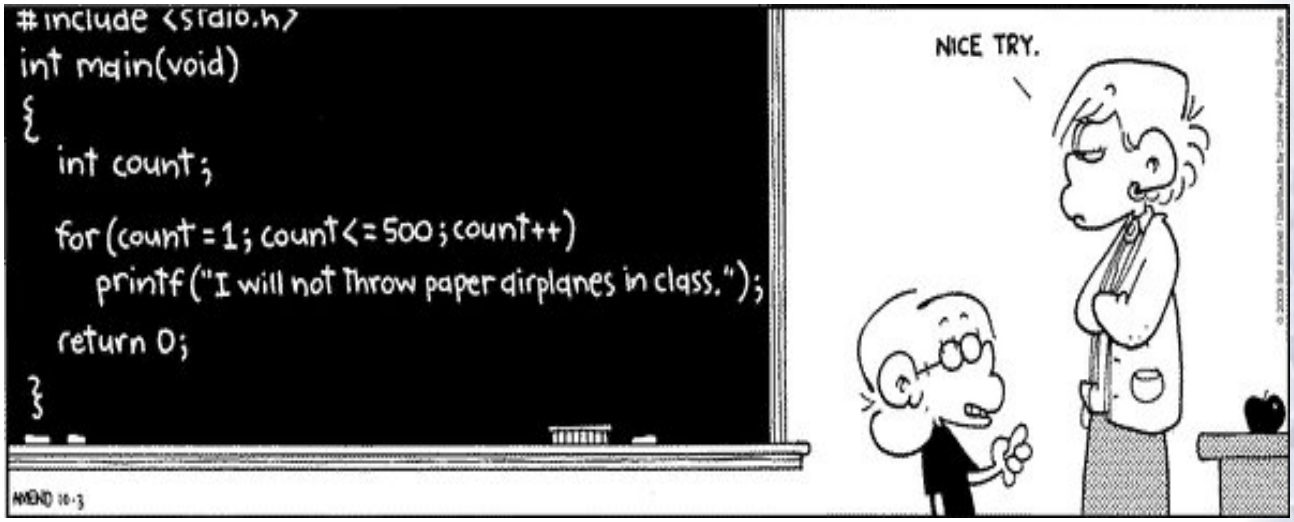




UNIVERSITEIT GENT

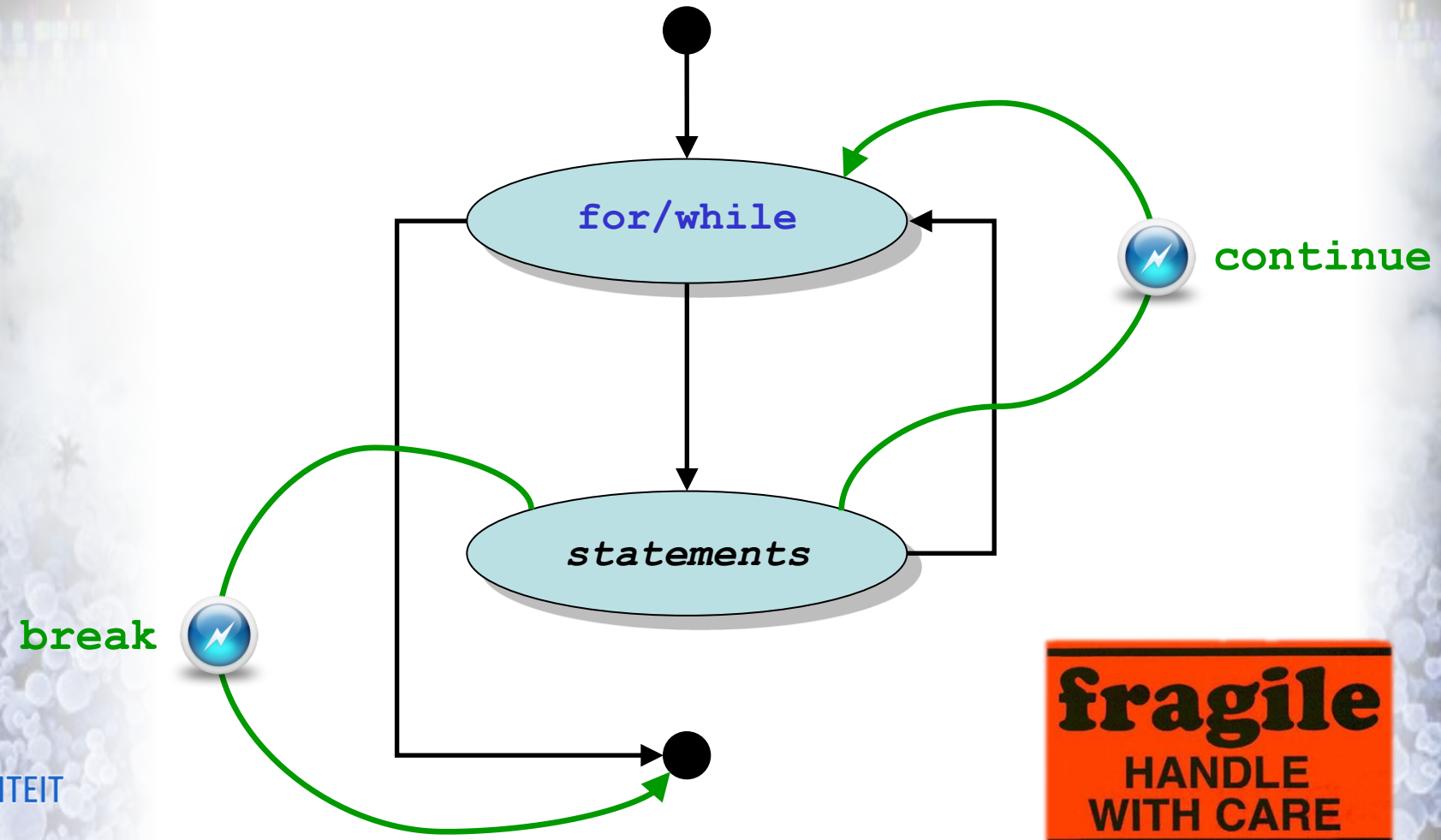
For or while ?

- for loops
  - number of iterations is **fixed** at the start of the loop
  - traversing elements of collection type using an iterator
- while loops
  - each iteration has an associated **condition** that determines whether or not a new iteration should be started



UNIVERSITEIT GENT

Short circuit



UNIVERSITEIT GENT

Guess the number

guess1.py

```
# choose random number between 1 and 10 (limits included)
import random
number = random.randint(1, 10)

# give player three turns to guess the number
guessed = False
for attempts in range(3, 0, -1):

    # ask player to make a guess
    guess = int(input(f'Make a guess ({attempts} attempts left): '))

    # test if player has guessed the correct number
    if guess == number:
        print(f'Number guessed in {3 - attempts + 1} attempts!')
        guessed = True
        break
    else:
        print('Incorrect!')

# print correct number if not guessed by player
if not guessed:
    print(f'The number was {number}.')
```

UNIVERSITEIT GENT

Guess the number

guess1.py

```
# choose random number between 1 and 10 (limits included)
import random
number = random.randint(1, 10)

# give player three turns to guess the number
guessed = False
for attempts in range(3, 0, -1):

    # ask player to make a guess
    guess = int(input(f'Make a guess ({attempts} attempts left): '))

    # test if player has guessed the correct number
    if guess == number:
        print(f'Number guessed in {3 - attempts + 1} attempts!')
        guessed = True
        break
    else:
        print('Incorrect!')

# print correct number if not guessed by player
if not guessed:
    print(f'The number was {number}.')
```

UNIVERSITEIT GENT

Guess the number

guess2.py

```
# choose random number between 1 and 10 (limits included)
import random
number = random.randint(1, 10)

# give player three turns to guess the number
attempts, guessed = 3, False
while attempts and not guessed:

    # ask player to make a guess
    guess = int(input(f'Make a guess ({attempts} attempts left): '))
    attempts -= 1

    # test if player has guessed the correct number
    if guess == number:
        print(f'Number guessed in {3 - attempts} turns!')
        guessed = True
    else:
        print('Incorrect!')

# print correct number if not guessed by player
if not guessed:
    print('The number was {number}.')
```

UNIVERSITEIT GENT

Guess the number

guess3.py

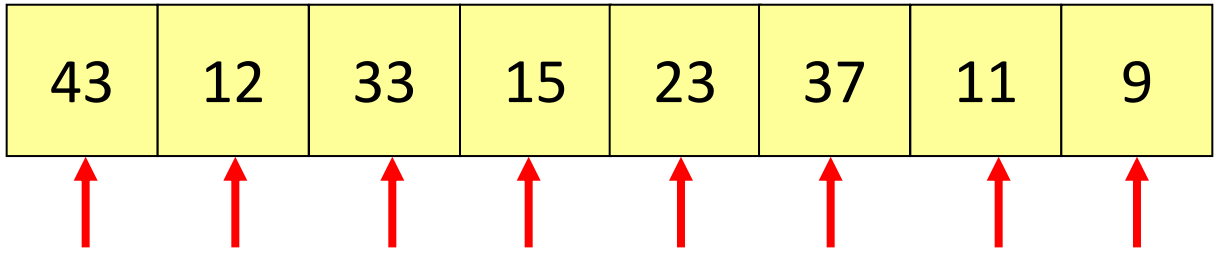
```
# count number of attemtps to guess a number between 1 and 100

# choose random number between 1 and 10 (limits included)
import random
number = random.randint(1, 100)

# allow player to guess the number until it is found
attempts, guess = 0, 0
while guess != number:

    # ask player to make a guess
    attempts += 1
    guess = int(input('What is your next guess? '))

    # hint about the number to be guessed
    if guess < number:
        print('Higher!!')
    elif guess > number:
        print('Lower!!')

# print number of attempts needed to guess the number
print(f'Number was guessed in {attempts} attempts.')
```

UNIVERSITEIT GENT

Series minimum and average

Write a program that computes the minimum and the average of a series of *n* integers.

```
$ python average.py < input.txt
minimum: 9
average: 22.875
$
```



min = 43 min = 12 min = 12 min = 12 min = 12 min = 11 min = 9

sum = 43 sum = 55 sum = 88 sum = 103 sum = 126 sum = 163 sum = 174 = 183

UNIVERSITEIT GENT

Series minimum and average

average.py

```
# determine minimum and average for a series of integers

# read length of series
n = int(input('Enter number of integers in series: '))

# initialize running variables
myMin = int(input('Enter first integer: '))
mySum = myMin

# traverse series
for i in range(n - 1):
    number = int(input('Enter next integer: '))
    mySum += number
    if number < myMin:
        myMin = number

# print minimum and average
print('minimum:', myMin)
print('average:', float(mySum) / n)
```

UNIVERSITEIT GENT

Closed series

Write a program that computes the sum of squares of a series of *n* integers.

$$\sum_{i=1}^{n} i^2$$

1	4	9	16	25	36	49	64

UNIVERSITEIT GENT

Closed series

Write a program that computes the sum of squares of a series of *n* integers.

$$\sum_{i=1}^{n} i^2$$

sum_closed_series.py

```
# read length of series
n = int(input('Enter number of integers in series: '))

# compute sum of squares
mySum = 0
for i in range(1, n + 1):
    mySum += i ** 2

# print sum of squares
print(f'The sum of the first {n} squares is {mySum}.')
```

UNIVERSITEIT GENT

Open series

Write a program that computes the sum of squares of a series of integers that ends with an empty input.

117	44	12	87	48	94	38	

UNIVERSITEIT GENT

Open series

Write a program that computes the sum of squares of a series of integers that ends with an empty input.

sum_open_series.py

```
# compute sum of open series
mySum = 0
myInput = input('Enter next number (press [enter] to stop): ')
while myInput:
    mySum += int(myInput)
    myInput = input('Enter next number (press [enter] to stop): ')

# print sum of series
print(f'The sum of the numbers is {mySum}.')
```

UNIVERSITEIT GENT

Nested loops

Write a program that computes the result of the following double sum of series.

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \frac{1}{i^2 + j^2}$$

UNIVERSITEIT GENT

Nested loops

Write a program that computes the result of the following double sum of series.

double_sum.py

```
# read limits
m = int(input('Enter value for m: '))
n = int(input('Enter value for n: '))

# compute double sum of series
mySum = 0.0
for i in range(1, m + 1):
    for j in range(1, n + 1):
        mySum += 1 / (i ** 2 + j ** 2)

# print sum
print(f'The double sum of the series is {mySum:.6f}.')
```

UNIVERSITEIT GENT

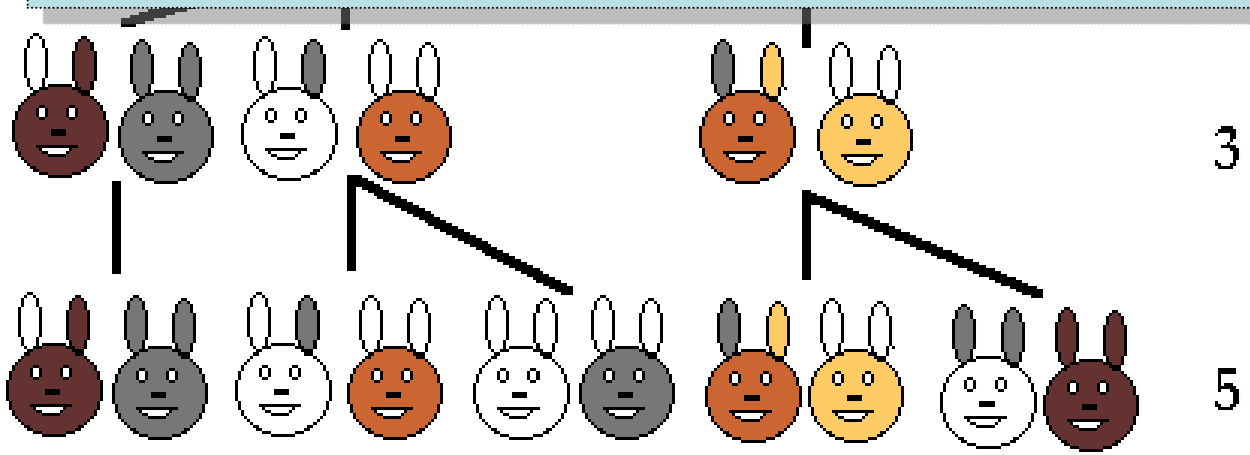
Fibonacci

 # rabbit

Suppose a newly-born pair of rabbits, one male, one female, are put in a field. Rabbits are able to mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits.

Suppose that our rabbits **never die** and that the female **always** produces one new pair (one male, one female) **every month** from the second month on. The puzzle that Fibonacci posed was...

How many pairs will there be in one year?



UNIVERSITEIT GENT

Fibonacci

Write a program that prints the first *n* numbers in the Fibonacci series.

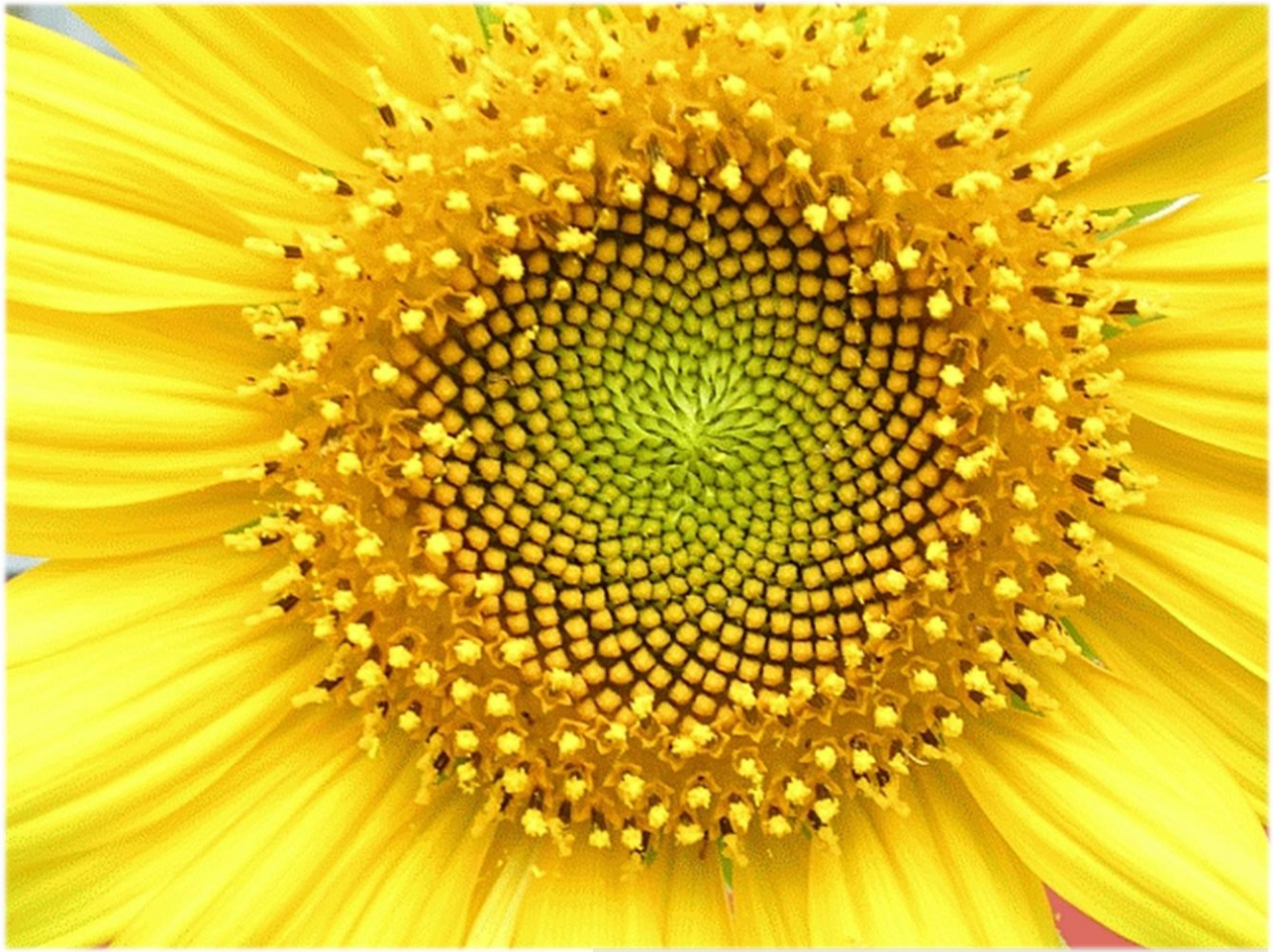
$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n > 1) \end{cases}$$

fibonacci.py

```
# read value of n
n = int(input('Enter value of n: '))

# initialize and print first numbers
f0, f1 = 0, 1
print(f1)

# compute and print next number
for i in range(n - 1):
    f0, f1 = f1, f0 + f1
    print(f1)
```

UNIVERSITEIT GENT

Fibonacci in nature



Head displaying florets in spirals of 34 and 55 around the outside

UNIVERSITEIT GENT

Homework (next lecture)

- course book
  - read chapter 4 (strings)
  - read chapter 6 (functions)
- read problem description of demo exercises
  - Caesar cipher
  - Letter soup
  - Toothpicks
  - Emirp
  - Sanger sequencing
  - Mobile synonyms

Homework (hands-on sessions)

- mandatory exercises of series 03 (control loops) (deadline Tuesday, October 14, 2025, 22:00)
  - ISBN (demo)
  - Chaos
  - German tanks
  - Monkeys and coconuts
  - Pythagorean triples
  - Hitchhikers problem

Questions or remarks ?

UNIVERSITEIT GENT

The sky is the limit …

"If you don't know where you are going,
any road will get you there."

— Lewis Carroll

UNIVERSITEIT GENT