

# Programming For Data Science

Part One:

## Introduction

Giulio Rossetti

[giulio.rossetti@isti.cnr.it](mailto:giulio.rossetti@isti.cnr.it)

# Textbooks

## **The Coder's Apprentice**

*Learning Programming with Python 3*

Pieter Spronck

[www.spronck.net/pythonbook](http://www.spronck.net/pythonbook)

## **How to Think Like a Computer Scientist**

Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers

<http://interactivepython.org/runestone/static/thinkcspy/index.html>

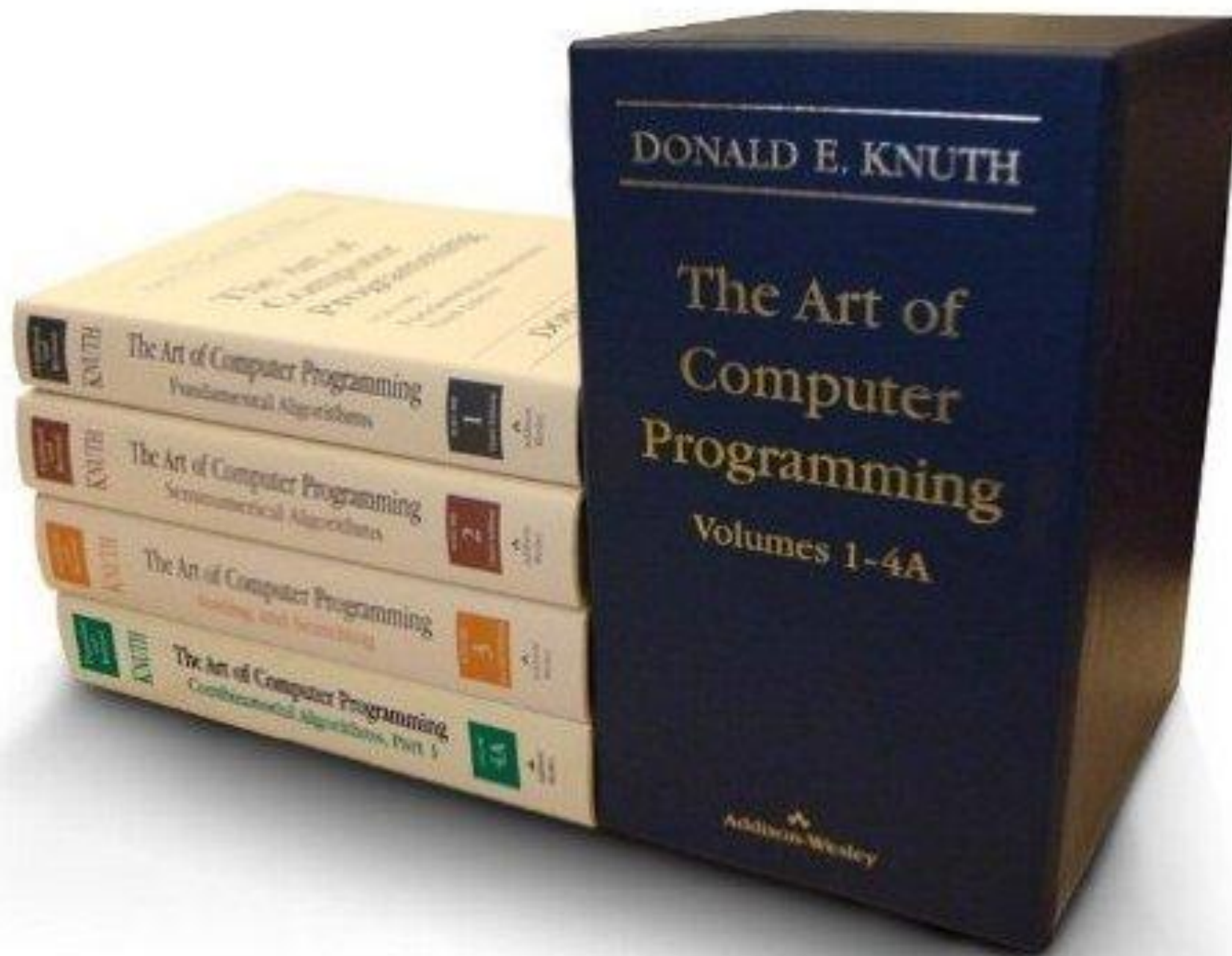
# Computer Science?

The systematic study of **algorithmic** processes that describe and transform **information**: their theory, analysis, design, efficiency, implementation, and application  
*(Association for Computing Machinery)*

# Algorithm

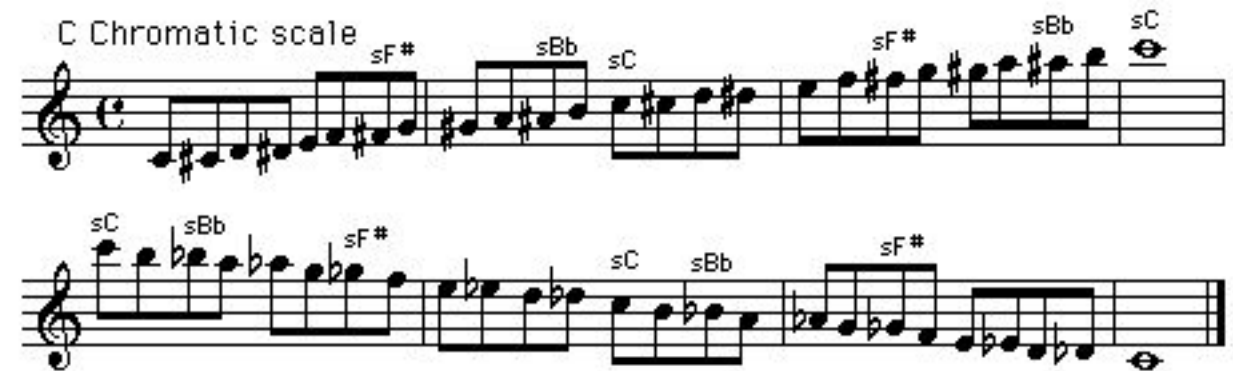
A self-contained sequence of actions to be performed, and can be expressed within a finite amount of time and space

# The art of computer programming

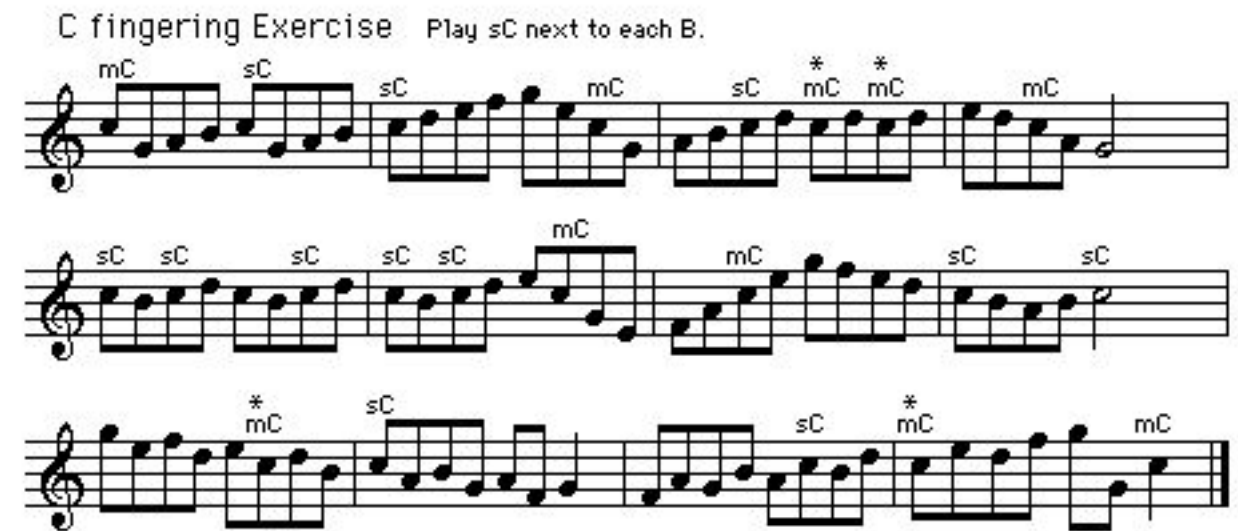


# Thus....

- Practice
- Exercise



Always use side key fingerings for F#, Bb and C in chromatic passages.



(\* = you can keep right hand fingers down.)

# The computer executes

- Computers execute (simple) operations, *quickly* and with high *precision*
- A program is the description of an algorithm, expressed in a programming language that the computer understands
- The computers receives in **input** a **program** (text) and a set of **data** and produces in **output** the results of the execution of the program
- In contrast with other automatic machines (washing machines, small calculators, etc.) computers are programmable: the task depends on the program. *The machine does not change when the task changes.*

# Programming

- Definition of the problem (**specification**)
- Design of the solution (**algorithm**)
- Implement the solution in a specific programming language (**coding**)
- Execution of the program (**run**)



# Specification

- Define and **understand** the task to solve
- The specification can be more or less **formal**, depending on how it is produced
- At an high level, you need to specify the initial state of the problem (**input**) and what you expect as a result (**output**)

# Specification (examples)

- Given two numbers, find the smallest
  - $(-2,56; 78,00) \longrightarrow -2,56$



# Specification (examples)

Given a phonebook and a name, find the number corresponding to the name

- Duplicates? Missing?



# Specification (examples)

Given a map and info on the traffic, find the quickest way to reach point B from A

- The description of the problem does not give hint on the solution



# Specification (examples)

Write all even numbers that are not the sum of two prime numbers

- It is not know how to do it (Goldbach Conjecture)



# Specification (examples)

Decide if for any program A and input D, it terminates when executed on input D

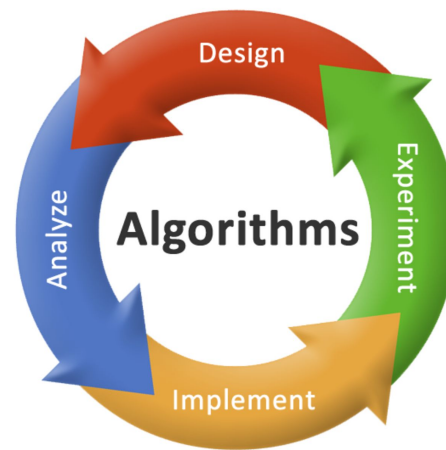
- Terminate(A,D): IF A terminates  $\rightarrow$  **TRUE**, ELSE **FALSE**
- Paradox(A): IF Terminate(A,A)  $\rightarrow$  LOOP(), ELSE **FALSE**
- **What happens on Paradox(Paradox)?**
  - Paradox(Paradox) terminates  $\leftrightarrow$  Paradox(Paradox) does not terminate



Undecidable Problem

# Algorithm

- After the specification, it is possible to **design the algorithm**: a set of actions to perform to achieve the desired results
- The process that the human mind takes to design the solution needs to be *coded* into the **algorithm**
- There are problems that do not have an algorithm to solves them!



# Algorithm's properties

- **Finiteness**
  - Finite sequence of steps
- **Not ambiguous**
  - Each single step cannot be ambiguous (computer is executing!)
- **Executable**
  - The executor can execute each single step in finite time with the available resources



# Coding and Implementing

- Translate the algorithm into a programming language
- Programming languages offer specific tools to help the translation
  - input
  - output
  - information
- Different programmers, using the same programming language, will implement the same algorithm in different ways (remember, programming is an art!)

# Programming For Data Science

Part Two:

## Python: Introduction

Giulio Rossetti

[giulio.rossetti@isti.cnr.it](mailto:giulio.rossetti@isti.cnr.it)

# Python (programming language)

---

From Wikipedia, the free encyclopedia

**Python** is a [widely used high-level programming language](#) for [general-purpose programming](#), created by [Guido van Rossum](#) and first released in 1991. [An interpreted language](#), Python has a design philosophy that emphasizes code [readability](#) (notably using [whitespace](#) indentation to delimit [code blocks](#) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](#) than might be used in languages such as [C++](#) or [Java](#).<sup>[23][24]</sup> The language provides constructs intended to enable writing clear programs on both a small and large scale.<sup>[25]</sup>

Python features a [dynamic type](#) system and automatic [memory management](#) and supports multiple [programming paradigms](#), including [object-oriented](#), [imperative](#), [functional programming](#), and [procedural](#) styles. It has a large and comprehensive [standard library](#).<sup>[26]</sup>

Python interpreters are available for many [operating systems](#), allowing Python code to run on a wide variety of systems. [CPython](#), the [reference implementation](#) of Python, is [open source](#) software<sup>[27]</sup> and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](#).



- Python programs are **text files**
- By convention, the name of those files have the extension **.py**
- You will find a lot of *resources on the web*, including tutorial
  - e.g., <http://learnpython.org/>



- Tools you will need
  - Anaconda with Python 3 (Windows, MacOS, Linux)
    - Jupyter Notebooks
    - Pycharm, Spyder

# Everybody starts like this: Hello, World!

In a text editor:

```
print("Hello, World!")
```

Parameter

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
```

what if we want to print more than one sentence?

The program will be executed one statement after the other

```
print("Hello, World!")  
print("How are you?")
```

# Console Vs. Program

- From the Console:
  - `print(7/4)`
- From the Console:
  - `7/4`

Differences?

# Console Vs. Program

- From the **Console**:

- `print(7/4)`

- From the **Console**:

- `7/4`



No!

- The Console will always display the result of a command
  - $7/4 = 1.75$
  - The result of print is “nothing”; however the **print** command causes the display of whatever is within the parentheses



# Console Vs. Program

Write now a **program** that only contains

`7/4`

What do you expect  
as a result?

Write now a **program** that only contains

`print(7/4)`

What do you expect  
as a result?

# Expressions

Straightforward calculations which you can also do with any simple calculator

$$5+7$$

*What do you need to display the result of an expression?*

# The print function

The syntax is quite easy

```
print("whatever you want to display")
```

# The print function

You multiple things with one **print()** function

```
print("I", "am", "having", "fun", "!")
```

```
print( "I", "am", "having", "fun",  "!" )
```

```
print ("I", "am", "having", "fun", "!" )
```

Spaces superfluous

# Python can even count!

```
print("3 + 5 =", 3 + 5)
print("3 * 5 =", 3 * 5)
print("3 - 5 =", 10 - 5)
```

```
$ python example.py
```

```
3+5 = 8
```

```
3*5 = 15
```

```
3-5 = 5
```

| Operator       | Symbol | Example      |
|----------------|--------|--------------|
| Power          | **     | 5 ** 2 == 25 |
| Multiplication | *      | 2 * 3 == 6   |
| Division       | /      | 14 / 3 == 4  |
| Remainder      | %      | 14 % 3 == 2  |
| Addition       | +      | 1 + 2 == 3   |
| Subtraction    | -      | 4 - 3 == 1   |

# Exercises

1. Write and run a program that displays your first and last name
2. Write and run a program that uses all math operations (pay attention to precedences)

# Programming For Data Science

Part Three:

## Python: Let's get more serious

Giulio Rossetti

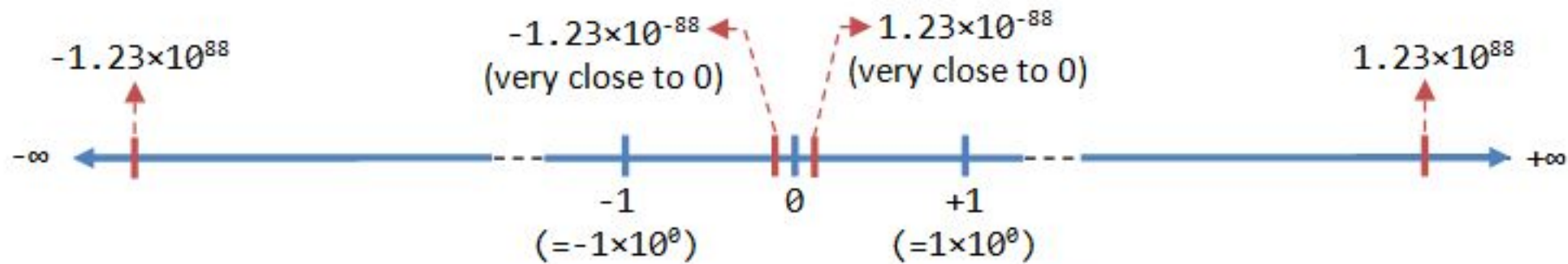
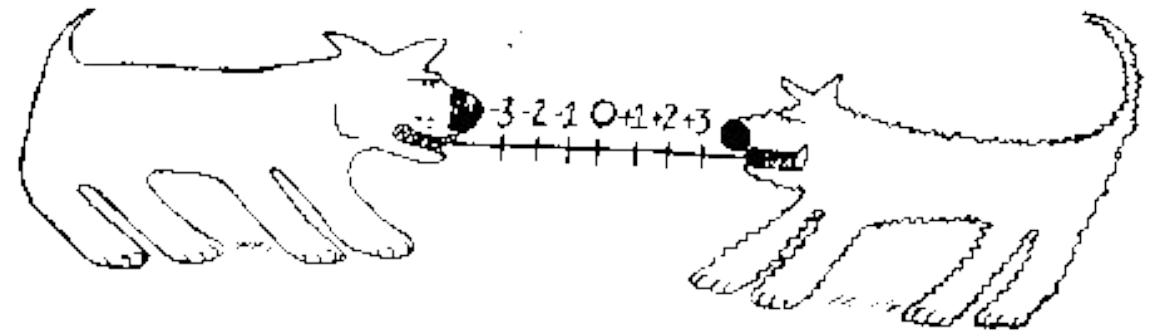
[giulio.rossetti@isti.cnr.it](mailto:giulio.rossetti@isti.cnr.it)

# Data types

Strings

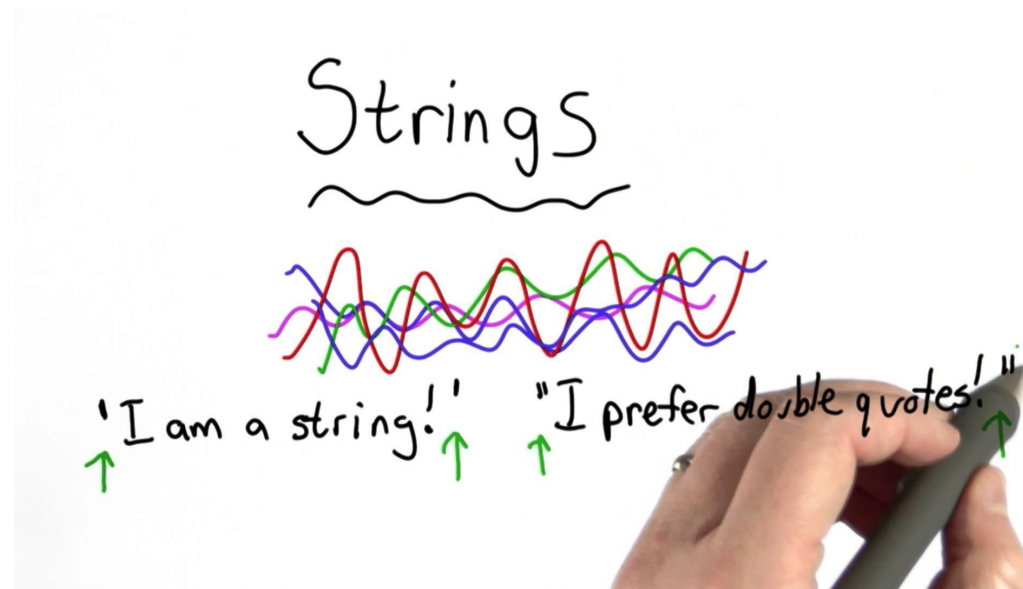
'I am a string!'

"I prefer double quotes!"

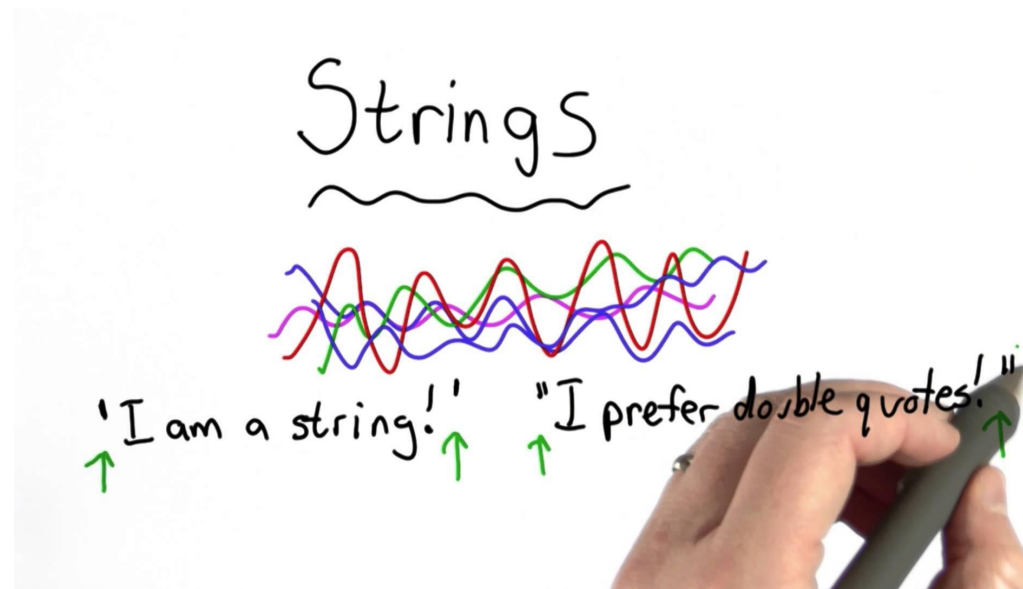


**Floating-point Numbers (Decimal)**

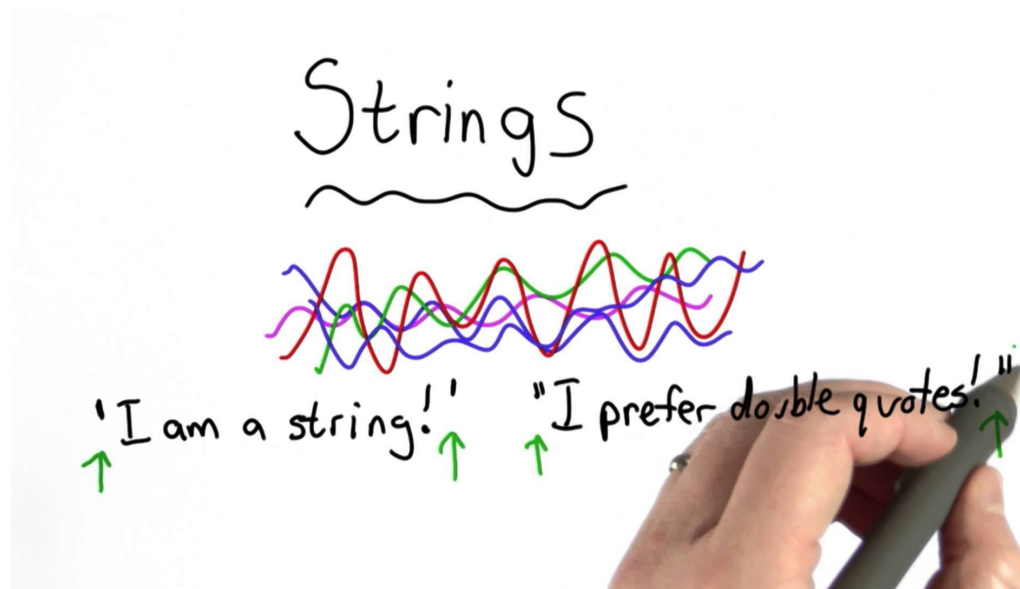




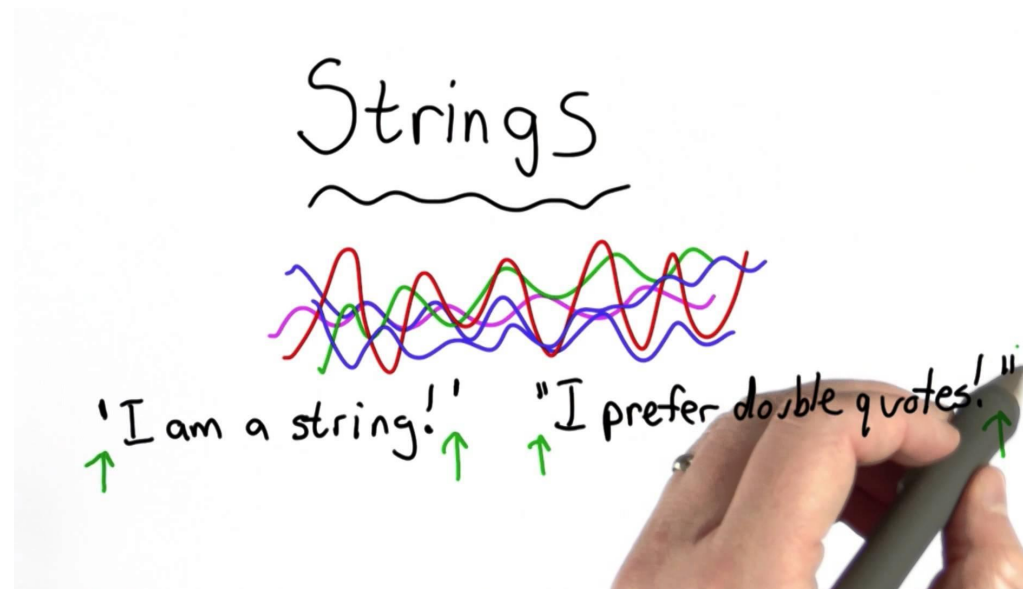
- Is a text
  - $\geq 0$  characters
- Enclosed by “” or “
  - Usually it does not matter...  
*when it might make a difference which one you use?*



- Is a text
  - $\geq 0$  characters
- Enclosed by “” or “”
  - Usually it does not matter...  
*when it might make a difference which one you use?*
  - *What if the strings does contain both “” and “” ?*



- Is a text
  - $\geq 0$  characters
- Enclosed by “” or “
  - Usually it does not matter...  
*when it might make a difference which one you use?*
  - *What if the strings does contain both “” and ‘ ’ ?*
  - Use the escape character \ before each “ or ‘
    - \" and \'

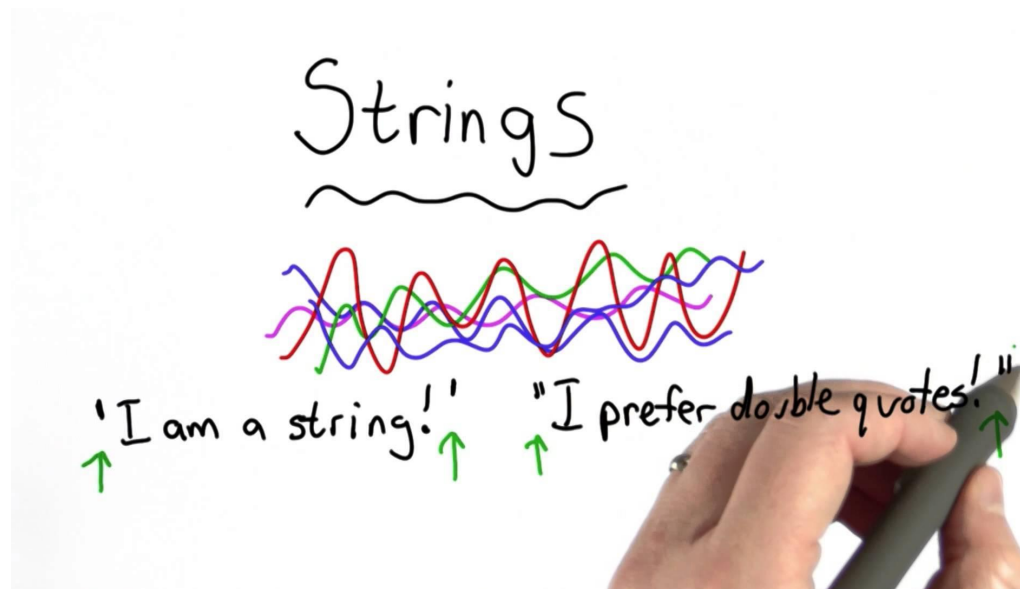


○ And now let's get serious:

■ *What if I want to put an actual backslash in a string?*

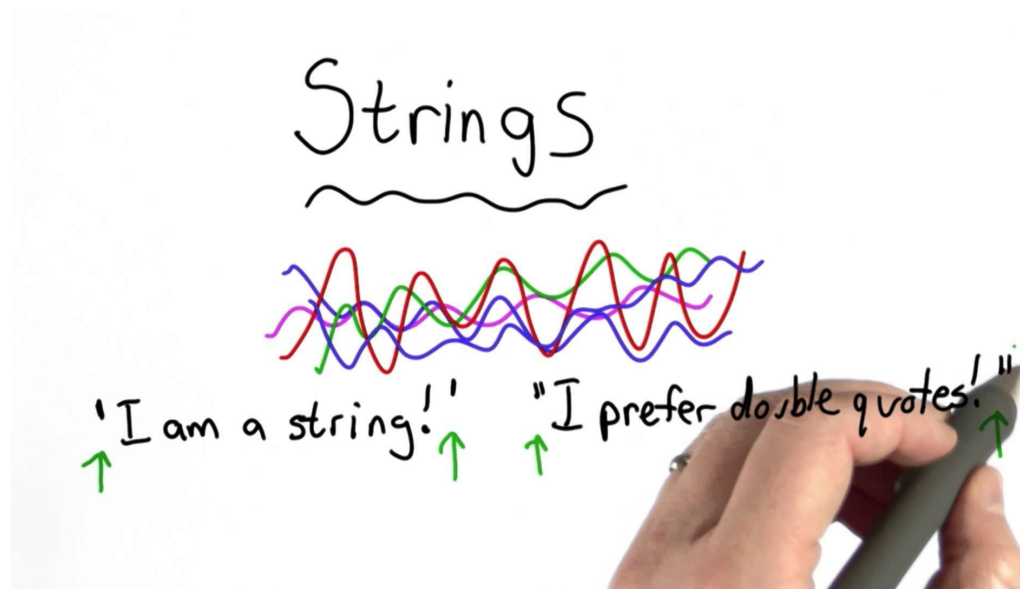
● *No problems: just put it*

```
print(2, "\", 2)
```



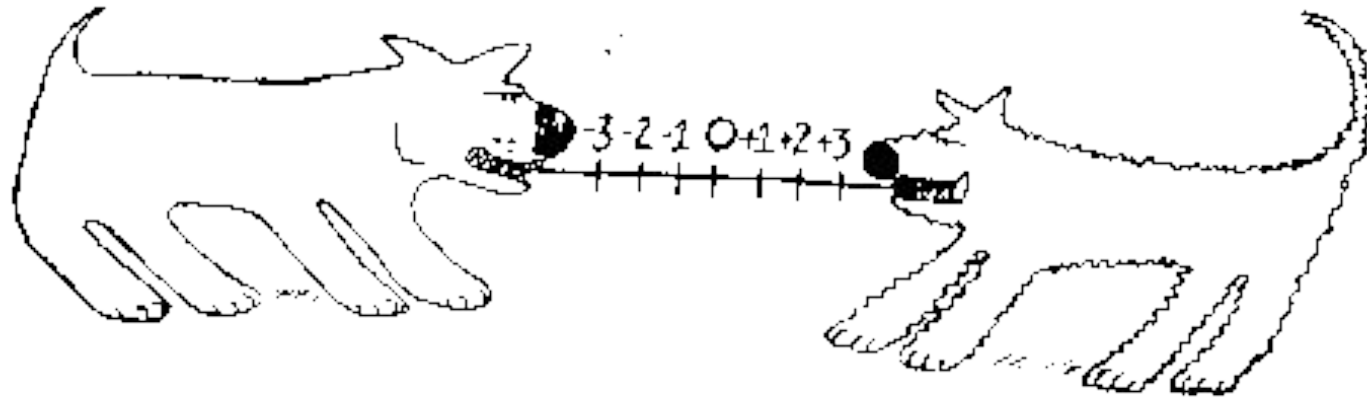
- And now let's get serious:
  - *What if I want to put an actual backslash in a string?*
    - *No problems: just put it*
  - *And if the \ is in front of a single or double quote?*  
*e.g. 'I can\'t stand it'*

```
print('????')
```

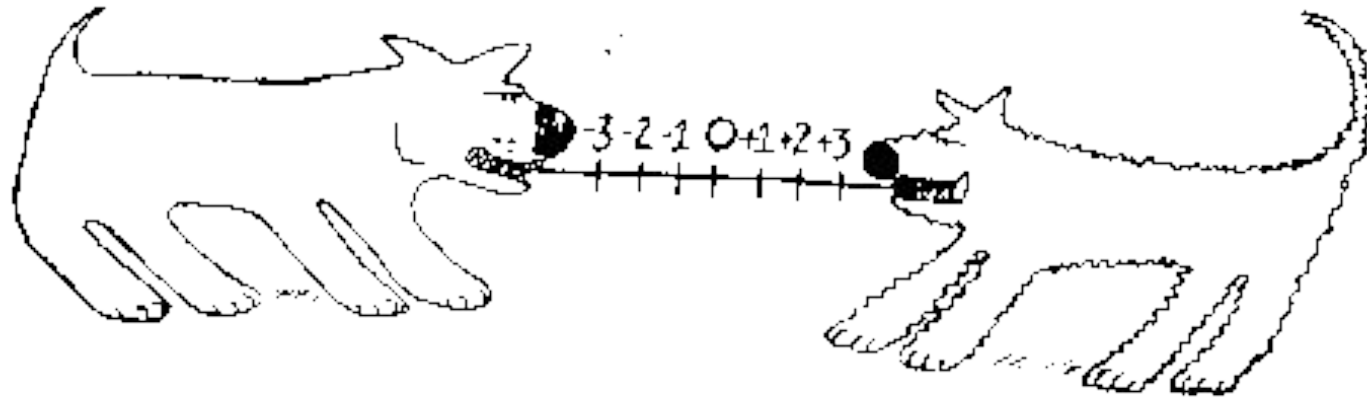


- And now let's get serious:
  - *What if I want to put an actual backslash in a string?*
    - *No problems: just put it*
  - *And if the \ is in front of a single or double quote?*  
*e.g. 'I can\'t stand it'*
  - Just escape it!

```
print('I can\\\'t stand it')
```

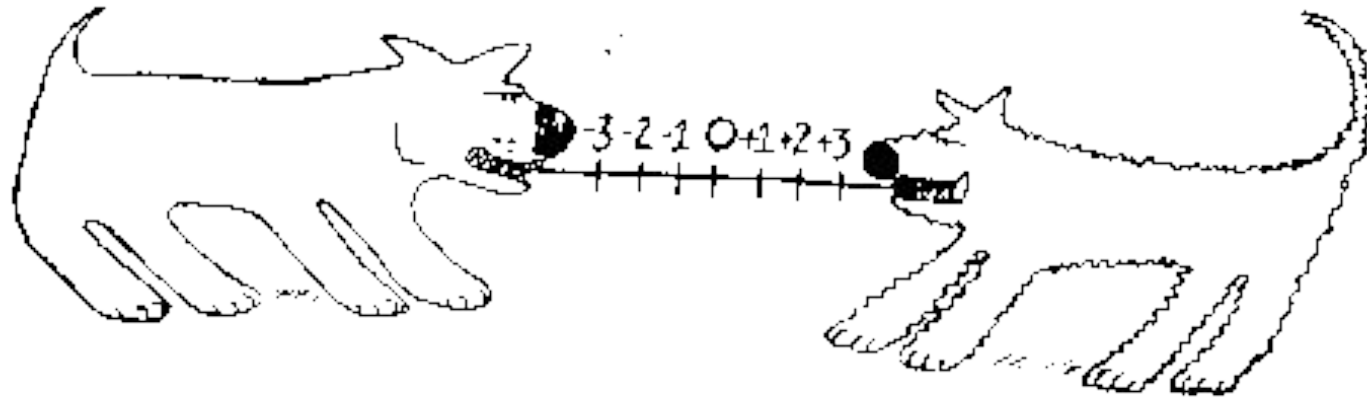


- Are whole numbers,  $< 0$ ,  $0$ ,  $> 0$
- There exists a max size an integer can become
  - It depends on the computer
- Can be displayed using **print()**
  - `print(l)` is the same as `print(+l)`
  - *Would it be the same with the string "1"?*



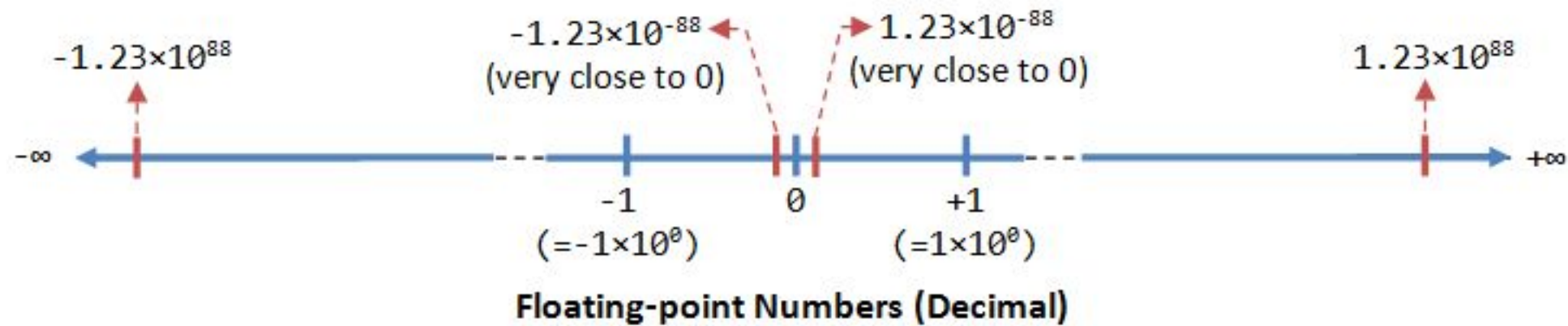
- Are whole numbers,  $< 0$ ,  $0$ ,  $> 0$
- There exists a max size an integer can become
  - It depends on the computer
- Can be displayed using **print()**
  - `print(l)` is the same as `print(+l)`
  - *Would it be the same with the string "1"?*
  - *What* `print(1,000,000,000)` *would display?*
    - i.e. using **thousands separator**





- Are whole numbers,  $< 0$ ,  $0$ ,  $> 0$
- There exists a max size an integer can become
  - It depends on the computer
- Can be displayed using **print()**
  - `print(l)` is the same as `print(+l)`
  - *Would it be the same with the string "1"?*
  - *What* `print(1,000,000,000)` *would display?*
    - i.e. using **thousands separator**

1 0 0 0



- Are **numbers with decimals** (called also “floating-point numbers”)
- Period as a decimal separator
- To use an integer as a float, just put .0 after it
- Like integers, also floats have maximum boundaries
- Note that you cannot expect perfect precision
  - `print( (431 / 100) * 100 )` —> 430.99999999999994

# Expressions

- **A combination of one or more values** (such as strings, integers, floats)  
using operators, which give a new value
- Basic calculations
  - Basic operations
    - `+`, `-`, `*`, `/`
    - `**` (power), `//` (integer division), `%` (modulo)
  - More complex calculations
    - Combine in the same expression more operators

# Expressions

- String expressions

- +: **concatenate**

- “hello” + “world”

- \*: **repetitions**

- 3\* “hello”

**It is not possible to concatenate a string and a number**

# Type casting

When you need to change the data type of a value into a different data type

## Three type casting functions

- **int()**

returns the value between the parentheses as an integer  
(rounding down if necessary)

- **float()**

returns the value between the parentheses as a float  
(adding .0 if necessary)

- **str()**

returns the value between the parentheses as a string

# Type casting -- Examples

- **int()**

```
print( int( 15/4 ) )
```

- **float()**

```
print( float( 15+4 ) )
```

- **str()**

*How can you concatenate a string and an integer?*

# Type casting -- Examples

- **int()**

```
print( int( 15/4 ) )
```

- **float()**

```
print( float( 15+4 ) )
```

- **str()**

*How can you concatenate a string and an integer?*

```
print( "I am " + str( 25 ) + " years old." )
```

# Errors



- When something is wrong, Python rises **errors**
  - Syntax errors
  - Runtime Errors
    - e.g. **ZeroDivisionError**
- For instance, locate the problem in this program (2 statements):

```
print( ((2*3)/4 + (5-6/7)*8 )
```

```
print( ((12*13)/14 + (15-16)/17)*18 )
```



# State

A unique configuration of information in a program or machine

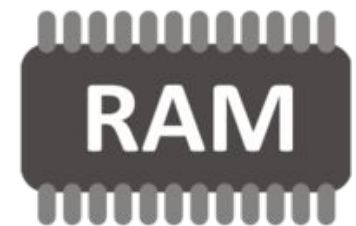


- For instance, the state can represent the position of an elevator, and its direction at a given floor
  - If we are at the second floor, and we have to go to the third, the state might be represented as **(2, up)**
  - The elevator moves, and its state changes to **(3, -)**

# State

A unique configuration of information in a program or machine

- For instance, the state can represent the position of an elevator, and its direction at a given floor
  - If we are at the second floor, and we have to go to the third, the state might be represented as (2, up)
  - The elevator moves, and its state changes to (3, -)
- **Program state** is a snapshot of the measure of various conditions in the system (e.g., memory)



# State

- The specification of a problem consists in the description of an **initial state** (problem's input data) and a **final state** (results)
- We need to find a **representation of the data** we need to manipulate, hence of the state
- An algorithm is a sequence of steps that repeatedly **modify the state** until the final one is reached
- Thus, the actions of executor (e.g., the computer) must be able to **change the state**
- For our purposes, we will consider the Program state (memory's content)
  - Change of state has the effect of changing some position in memory

# Program State

- A state is a set of association between (symbolic) names and values (at a given time instant)
  - In a given state, at each name is associated at most a value
- The name is referred to as **variable**

# Values and variables

*Values* are the main objects handled by programmers

These values have different *types*

Integer

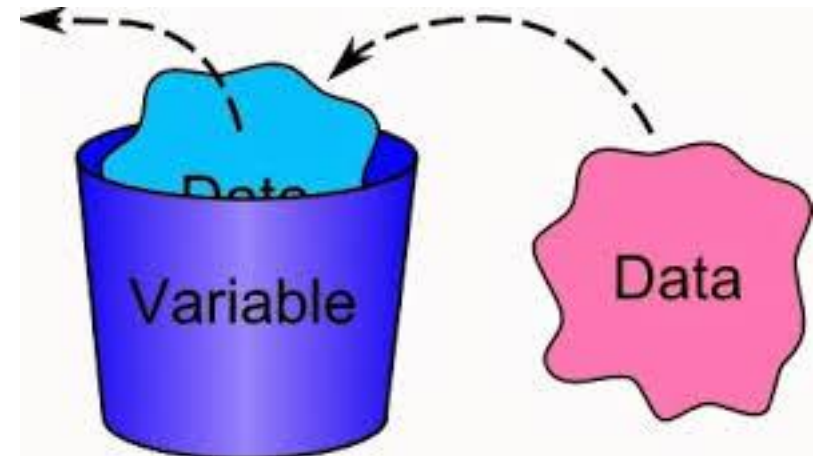
String

Float



# Values and variables

The *assignment* creates new variables and assigns them values



Va Assignment (creates the variable)

```
message = "Hi"  
print(message)  
message = "Am I having fun?"  
print(message)
```

Displays the value of message

# Variables manipulation

*Variables* can be manipulated using expressions

```
x=5
```

```
print( x )
```

5

```
x = 7 * 9 + 13 # overwrite the previous value of x
```

```
print( x )
```

76

```
x = "A nod's as good as a wink to a blind bat."
```

```
print( x )
```

A nod's as good as a wink to a blind bat.

```
x = int( 15 / 4 ) - 27
```

```
print( x )
```

-24


# Variables manipulation

*Variables* can be manipulated using expressions

```
x = 5
```

```
y = x + 7
```

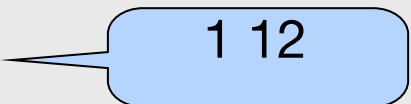
```
print(x, y)
```



5 12

```
x = 1
```

```
print(x, y)
```

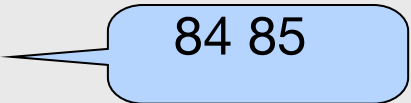


1 12

```
x = 7*y
```

```
y = x + 1
```

```
print(x, y)
```

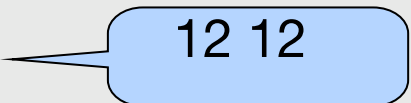


84 85

```
w = x/7
```

```
z = y/7
```

```
print(w, z)
```



12 12



# Variables manipulation

*Variables* can be manipulated using expressions

```
x=2
```

```
y=3
```

```
print( "x =", x )
```

x=2

```
print( "y =", y )
```

y=3

```
print( "x * y =", x * y ) print( "x + y =", x + y )
```

x \* y=6

# Variables manipulation

You may copy the contents from one variable to another

```
x=2
```

```
y=3
```

```
print( "x =", x, "and y =", y )
```

x=2 and y=3

```
# Variable assignment
```

```
z=x
```

```
x=y
```

```
y=z
```

```
print( "x =", x, "and y =", y )
```

x=3 and y=2

Single line comment

# Variables manipulation

You can even use the variable itself on **the right-hand side** of the assignment operator (provided it was created earlier)

```
x=2
```

```
print( x )
```

x=2

```
x=x+3
```

```
print( x )
```

x=5

# Variables manipulation

Spot the error....

```
print( days_in_a_year )  
days_in_a_year = 365
```



# Variable names

You are free to choose the names of your variables as you like them

- A variable name must consist of only letters, digits, and/or underscores (\_)
- A variable name must start with a letter or an underscore
  - Variable names are **cAsE SEnsitiVe**

A variable name should not be a **reserved word**

|        |          |         |          |        |
|--------|----------|---------|----------|--------|
| False  | class    | finally | is       | return |
| None   | continue | for     | lambda   | try    |
| True   | def      | from    | nonlocal | while  |
| and    | del      | global  | not      | with   |
| as     | elif     | if      | or       | yield  |
| assert | else     | import  | pass     |        |
| break  | except   | in      | raise    |        |

# Conventions

## Be reasonable

- *Never* choose variable names that are also the names of functions
- Variable names that are in some way **meaningful**
- Programmers tend to use only lower case letters in variable names
- Multiple words name: either **useful\_variable** or **usefulVariable**
- Programmers **never** choose variable names that start with an underscore

```
a = 3.14159265  
b = 7.5  
c = 8.25  
d=a*b*b*c/3  
print( d )
```



# Conventions



```
pi = 3.14159265  
radius = 7.5  
height = 8.25  
volume_of_cone = pi * radius * radius * height / 3  
print( volume_of_cone )
```

# Using variable for debugging

- Not always things go as you desire
- **Debugging** is the process of finding 'bugs'
- The easiest way of debugging your code is to print the variables content in strategic places of your program



# Data types

- In many programming languages you need to declare the types of your variable. For instance, in C (*hard typing*)
  - `int secs_per_week = 7 * 24 * 60 * 60;`
- Python does not have 'explicit' data typing
  - *Soft typing*

```
a=3
print( type( a ) )
a = 3.0
print( type( a ) )
a = "3.0"
print( type( a ) )
```

# Data types

Because of soft typing you need to **pay great attention** on the values that you store in a given variable over time

```
a=1  
b=4  
c = "|"  
d = "4"  
print( a + b )  
print( c + d )  
print( a + c )
```

Expected results?

# Data types

Because of soft typing you need to **pay great attention** on the values that you store in a given variable over time

```
a=1
```

```
b=4
```

```
c = "|"
```

```
d = "4"
```

```
print( a + b )
```

5

```
print( c + d )
```

14

```
print( a + c )
```

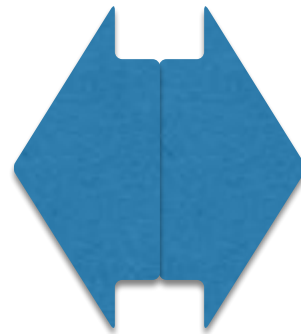
Error

# Alternative operators

```
number = 100
```

```
number = number + 1
```

```
print( number )
```



```
number = 100
```

```
number += 1
```

```
print( number )
```

```
number = 100
```

```
number += 12
```

```
number -= 13
```

```
number *= 19
```

```
number /= number
```

```
print(number )
```

# Comments

Two ways to add comments to your code

- The first is to use a hash mark (#)

`# comment: insert your code here`

- The second is to use **triple double-quotes** or **triple single-quotes** to indicate the start and end of some commentary, which *may be spread over multiple lines*

`"""Another way of commenting on your code is via triple quotes -- these can be distributed over multiple """`

# (Suggested) Exercises

1. Write a program that displays the number of seconds in a week.
2. The cover price of a book is €24.95, but bookstores get a 40 percent discount. Shipping costs €3 for the first copy and 75 cents for each additional copy. Calculate the total wholesale costs for 60 copies
3. Write a program that generates the `ZeroDivisionError`
4. You look at the clock and see that it is currently 14.00h.  
You set an alarm to go off 535 hours later.  
At what time will the alarm go off? Write a program that prints the answer.  
*Hint: for the best solution, you will need the modulo operator*
5. Solve previous exercise 1 by assign the calculation to a variable. Then add a statement to print the contents of the variable.
6. All exercises at the end of Chapter 4 of the book