



PyQB

Monga

Flow of  
control

Selections

Repetitions

# Programming in Python<sup>1</sup>

Mattia Monga

Dip. di Informatica  
Università degli Studi di Milano, Italia

`mattia.monga@unimi.it`

Academic year 2024/25, I semester



PyQB

Monga

Flow of  
control

Selections

Repetitions

## Lecture III: Control flow



# Basic types

`bool` `False`, `True` Logical operations

`int` 1, -33, 1\_000\_000\_000 ... Arithmetic operations, no upper or lower limit

`float` 1.0, .1, 1.2e34 ... Arithmetic operations, limited but you have `float('infinity')` (and `float('nan')`) (learn how write `literals!`)

```
sys.float_info(max=1.7976931348623157e+308,  
→ max_exp=1024, max_10_exp=308,  
→ min=2.2250738585072014e-308,  
→ min_exp=-1021, min_10_exp=-307, dig=15,  
→ mant_dig=53,  
→ epsilon=2.220446049250313e-16, radix=2,  
→ rounds=1)
```

`str` `'aaaa\nthis is on a new line'`,  
`"bbb'b\"b"` ... Concatenation, alphabetical ordering, replication, ...

PyQB

Monga

Flow of control

Selections

Repetitions



# Sequence of operations

```
1 x = 1 + 2 * 3  
2 x = x + 1
```

The 2 lines of code translate to at least 5 “logical” instructions (maybe more, for example adding two big numbers require multiple instructions):

- 1  $2 * 3$
- 2  $1 + 6$
- 3  $x = 7$
- 4  $7 + 1$
- 5  $x = 8$

PyQB

Monga

Flow of  
control

Selections

Repetitions



# Flow of control

It is normally not very useful to write programs that do just one single computation. You wouldn't teach a kid how to multiply  $32 \times 43$ , but the **general algorithm** of multiplication (the level of generality can vary).

To write programs that address a family of problems we need to be able to **select** instructions to execute according to **conditions**.

```
if x < 0:
    x = -x
y = 2 * x

if x == -1:
    x = x + 1
else:
    x = 3 * x
y = 2 * x
```

In Python the indentation is part of the syntax and it is **mandatory**.

PyQB

Monga

Flow of  
control

Selections

Repetitions



# Input (special command needed)

- A special command to ask to the operating system (same as `print`)
- `input()` or `input("Prompt the user:")`
- The operating system (or the operating environment as in `cscircle`) collect the input data (from keyboard/console or the network in `cscircles`) and returns them to Python as a `str`.
  - `s = input()` *## read a string*
  - `i = int(input())` *## read a string, convert to int*
- Input on `cscircles` seems strange, but when one understands the need of the mediation, the machinery is rather straightforward

PyQB

Monga

Flow of control

Selections

Repetitions



# Repetitions

It is also useful to be able to **repeat** instructions: it is very convenient, but it also opens a deep Pandora's box. . .

There are two ways of **looping** in Python:

Repeat by iterating on the elements of a collection (similar to math notation

$$\sum_{i \in \{a,b,c\}} f(i)$$

```
for i in range(0, 5):  
    # 0 1 2 3 4  
    print(i)
```

Repeat while a (variable) condition is true

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

PyQB

Monga

Flow of control

Selections

Repetitions



PyQB

Monga

Flow of  
control

Selections

Repetitions

- Create an account on <https://github.com/> (if you don't have one) and send me the name.