



PyQB

Monga

Flow of control
Selections
Repetitions

Programming in Python¹

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Academic year 2022/23, I semester

¹ © 2022 M. Monga. Creative Commons Attribution — Condividi allo stesso modo 4.0 Internazionale. <http://creativecommons.org/licenses/by-sa/4.0/deed.it>



PyQB

Monga

Flow of control
Selections
Repetitions

Lecture III: Control flow



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 32×43 , 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**.

Repetitions



PyQB

Monga

Flow of control
Selections
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
```

Loops can be difficult to understand



PyQB

Monga

Flow of control
Selections
Repetitions

When you have loops, understanding the code can be a difficult task and the only general strategy is to track the execution.

This is known as Collatz's procedure

```
n = ...
while n > 1:
    if n % 2 == 0:
        # if the remainder of division by 2 is 0, i.e. n
        # is even
        n = n / 2
    else:
        n = 3*n + 1
```

We know (by empirical evidence) that it ends for all $n < 2^{68} \approx 10^{20}$, nobody is able to predict the number of iterations given any n .

With loops it is also hard to exploit parallel execution.

Learn to write loops can be hard



PyQB

Monga

Flow of control
Selections
Repetitions

When you write a loop, you should have in mind two related goals:

- 1 **the loop must terminate:** this is normally easy with `for` loops (when the finite collection ends, the loop ends also), but it can be tricky with `while`s (remember to change something in the condition);
- 2 **the loop repeats something:** the programmer should be able to write the "repeating thing" in a way that makes it equal in its form (but probably different in what it does).

The second part (technically known as loop invariant) is the hardest to learn, since it requires experience, creativity, and ingenuity.

Homework



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.