



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

Programming in Python¹

Mattia Monga

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

Academic year 2021/22, II 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

Files

Types,
docstrings,
doctests

Abstracting
similarities

Lecture VIII: Files



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

Files

A file is an abstraction the operating system uses to preserve data among the execution of programs. Data must be accessed **sequentially**. (Italian reading people might enjoy this)

- We need commands to ask to the OS to give access to a file (`open`).
- It is easy to read or write data **sequentially**, otherwise you need special commands (`seek`) to move the file “cursor”
- The number of open files is limited (\approx thousands), thus it is better to `close` files when they are not in use

Files contain bits (normally considered by group of bytes, 8 bits), the interpretation (“format”) is given by the programs which manipulate them. However, “lines of printable characters” (plain text) is a rather universal/predefined interpretation, normally the easiest to program.



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

File read access

```
f = open('filename.txt', 'r') # read only

# iterating on a file reads (all) the lines
for i in f:
    print(i)

# End of file already reached, result is ''
f.readline()

f.close()

# File closed, error!
f.readline()

To avoid remembering to close explicitly, Python provides the
context manager syntax.
with open('filename.txt', 'r') as f:
    for i in f:
        print(i)
```

Make a program readable



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

You never write a program only for a machine! You, others, tools will *read* the program for different purposes. Every minute spent in making a program more understandable pays off hours saved later.

- Type hinting makes clear what a function needs to work properly, and what it produces
- Documentation helps understanding without the need to read implementation details
- Examples of use make easy to remember how to use a function and can be used for verification

50

Example



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

```
from typing import Union

Num = Union[int, float]

def cube(x: Num) -> Num:
    """Return the cube of x.

    >>> cube(-3)
    -27

    >>> abs(cube(0.2) - 0.008) < 10e-5
    True
    """
    return x * x * x
```

Examples can be tested by:
`python -m doctest filename.py.`

51

Procedural abstraction



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

Procedural abstraction is key for our thinking process (remember the power of recursion, for example): giving a name to a procedure/function enhances our problem solving skills.

```
def sum_int(a: int, b: int) -> int:
    """Sum integers from a through b.
```

```
>>> sum_int(1, 4)
10
```

```
>>> sum_int(3, 3)
3
"""
```

```
assert b >= a
result = 0
for i in range(a, b+1):
    result = result + i
return result
```

52

Another “sum”



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

This is very similar...

```
def sum_cubes(a: int, b: int) -> int:
    """Sum the cubes of the integers from a through b.

    >>> sum_cubes(1, 3)
    36

    >>> sum_cubes(-2, 2)
    0

    """
    assert b >= a
    result = 0
    for i in range(a, b+1):
        result = result + int(cube(i))
    return result
```

53

Another “sum”



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

This is also very similar...

$$\frac{1}{a(a+2)} + \frac{1}{(a+4)(a+6)} + \frac{1}{(a+8)(a+10)} + \dots + \frac{1}{(b-2)(b)}$$

(Leibniz: $\frac{1}{1 \cdot 3} + \frac{1}{5 \cdot 7} + \frac{1}{9 \cdot 11} + \dots = \frac{\pi}{8}$)

```
def pi_sum(a: int, b: int) -> float:
    """Sum  $\frac{1}{a(a+2)}$  terms until  $(a+2) > b$ .
```

```
>>> from math import pi
>>> abs(8*pi_sum(1, 1001) - pi) < 10e-3
True
```

```
"""
assert b >= a
result = 0.0
for i in range(a, b+1, 4):
    result = result + (1 / (i * (i + 2)))
return result
```

54

Can we abstract the similarity?



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

```
from typing import Callable

def gen_sum(a: int, b: int, fun: Callable[[int], Num], step: int = 1) -> Num:
    """Sum terms from a through b, incrementing by step.

    >>> gen_sum(1, 4, lambda x: x)
    10

    >>> gen_sum(1, 3, cube)
    36

    >>> from math import pi
    >>> abs(8*gen_sum(1, 1000, lambda x: 1 / (x * (x + 2)), 4) - pi) < 10e-3
    True

    """

    assert b >= a
    result = 0.0
    for i in range(a, b+1, step):
        result = result + fun(i)
    if result.is_integer():
        return int(result)
    return result
```

55

Homework



PyQB

Monga

Files

Types,
docstrings,
doctests

Abstracting
similarities

- <https://classroom.github.com/a/5W5A587v>

56