Programming in Python[1]

Mattia Monga

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

Academic year 2020/21, II semester

Lecture VII: Procedural abstraction

Make a program readable

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

Example


```python
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`.

Procedural abstraction

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
```

Another "sum"

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
```

Another "sum"

This is also very similar...

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

(Leibniz: $\frac{1}{1\cdot3} + \frac{1}{5\cdot7} + \frac{1}{9\cdot11} + \cdots = \frac{\pi}{8}$)

Another “sum”

This is also very similar. . .

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

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

```
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
```

Can we abstract the similarity?


```python
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
```

Homework

- https://classroom.github.com/a/51BJ-wgC