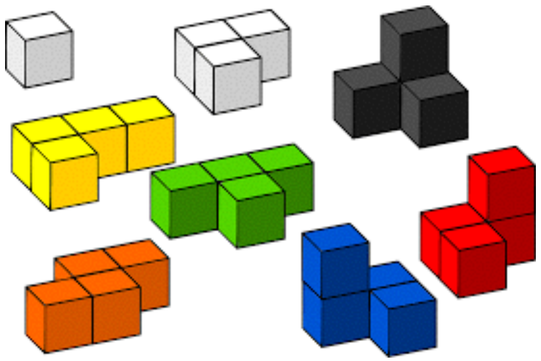
Class 15: Simulation and program design

Instructor: Michael Szell
Oct 23, 2019

Today you will learn about good program design

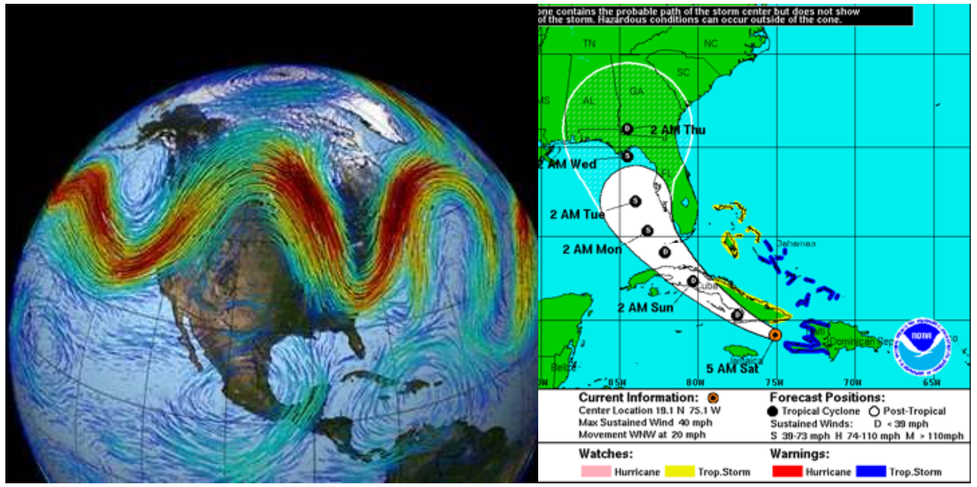
How to use random numbers

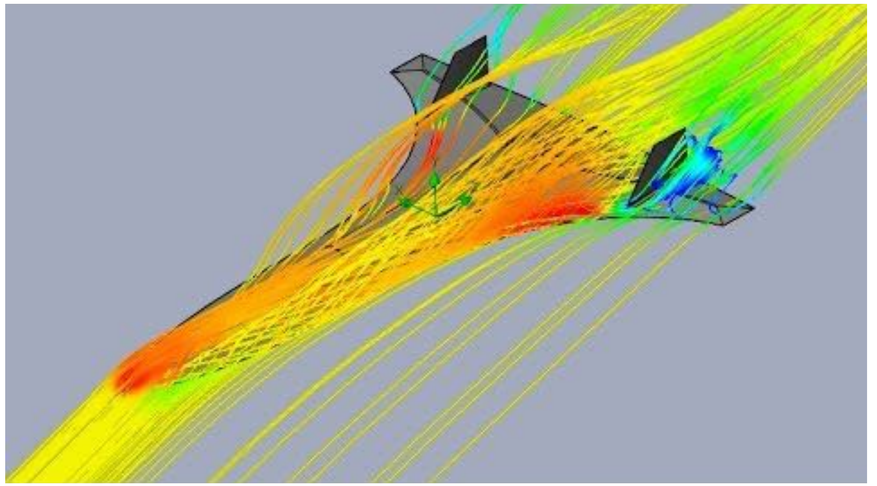
Unit testing



Top down design of raquetball simulation

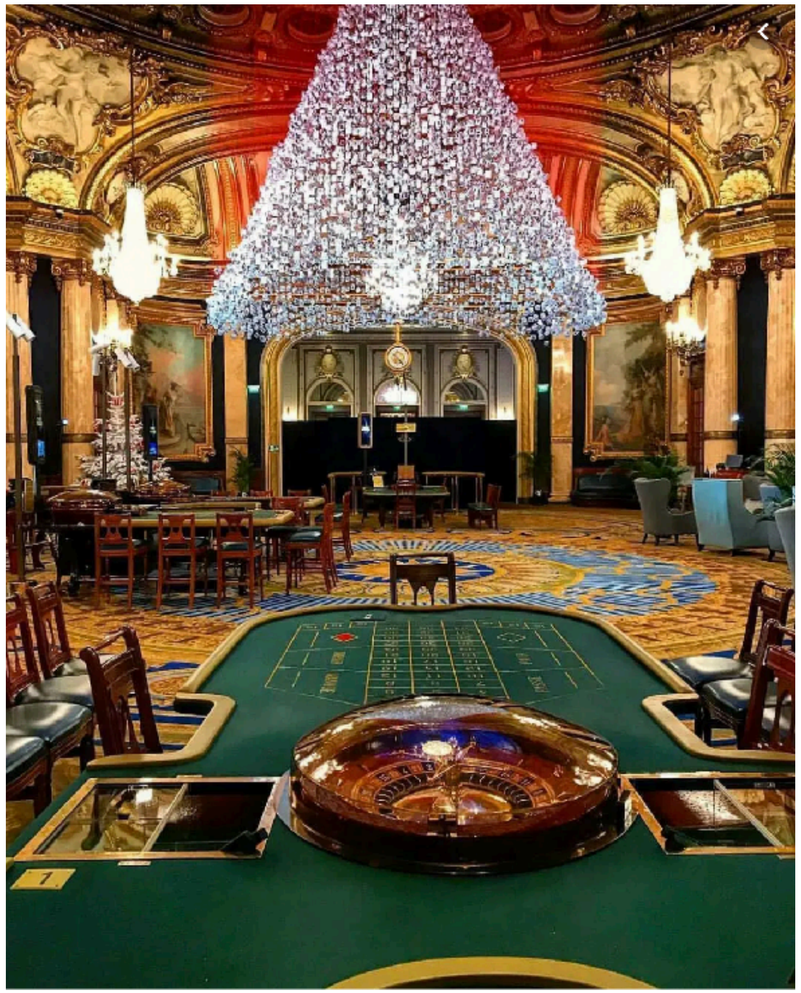
Simulation can give you an otherwise unobtainable understanding of real-world problems






E-FORCE
RENOVART
ICODER
GEARBOX
geep
COSTA RICA
NUTRI SNACKS
Rumba
OLYMPIC CHANNEL
LONG 4
MART 0

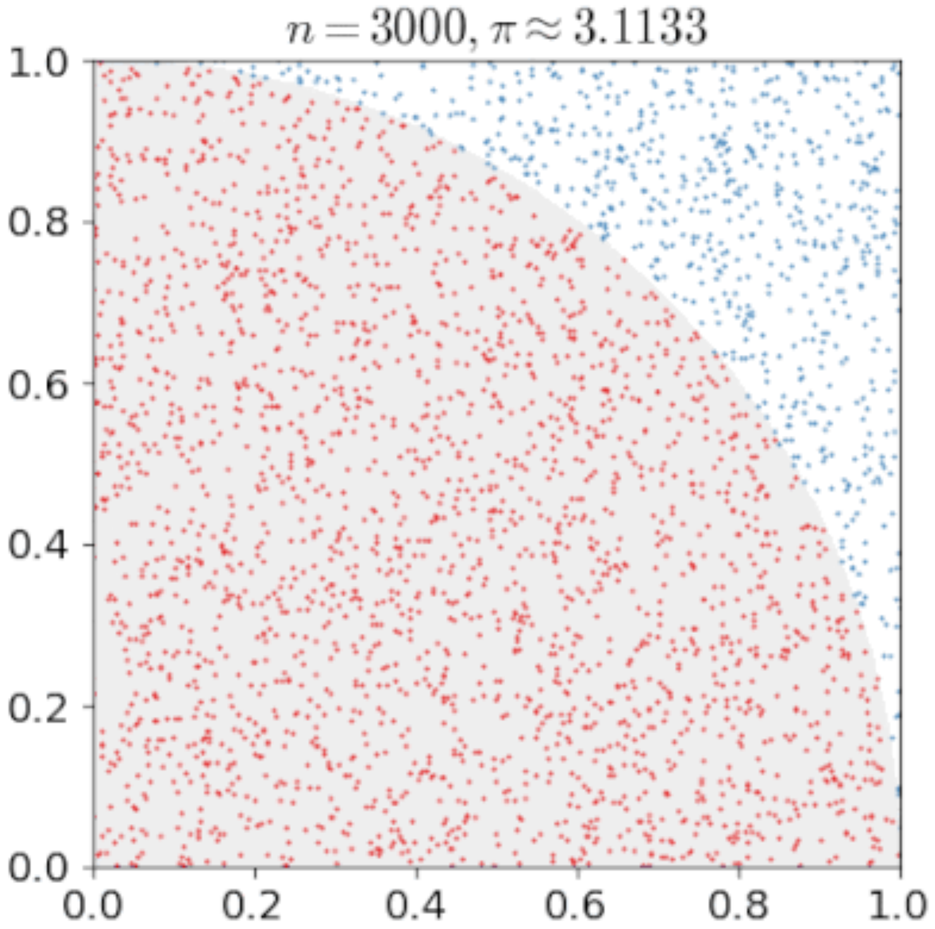
To simulate uncertain events, we generate random numbers



Monte Carlo methods are computational algorithms that rely on repeated random sampling to obtain numerical results.

https://en.wikipedia.org/wiki/Monte_Carlo_method

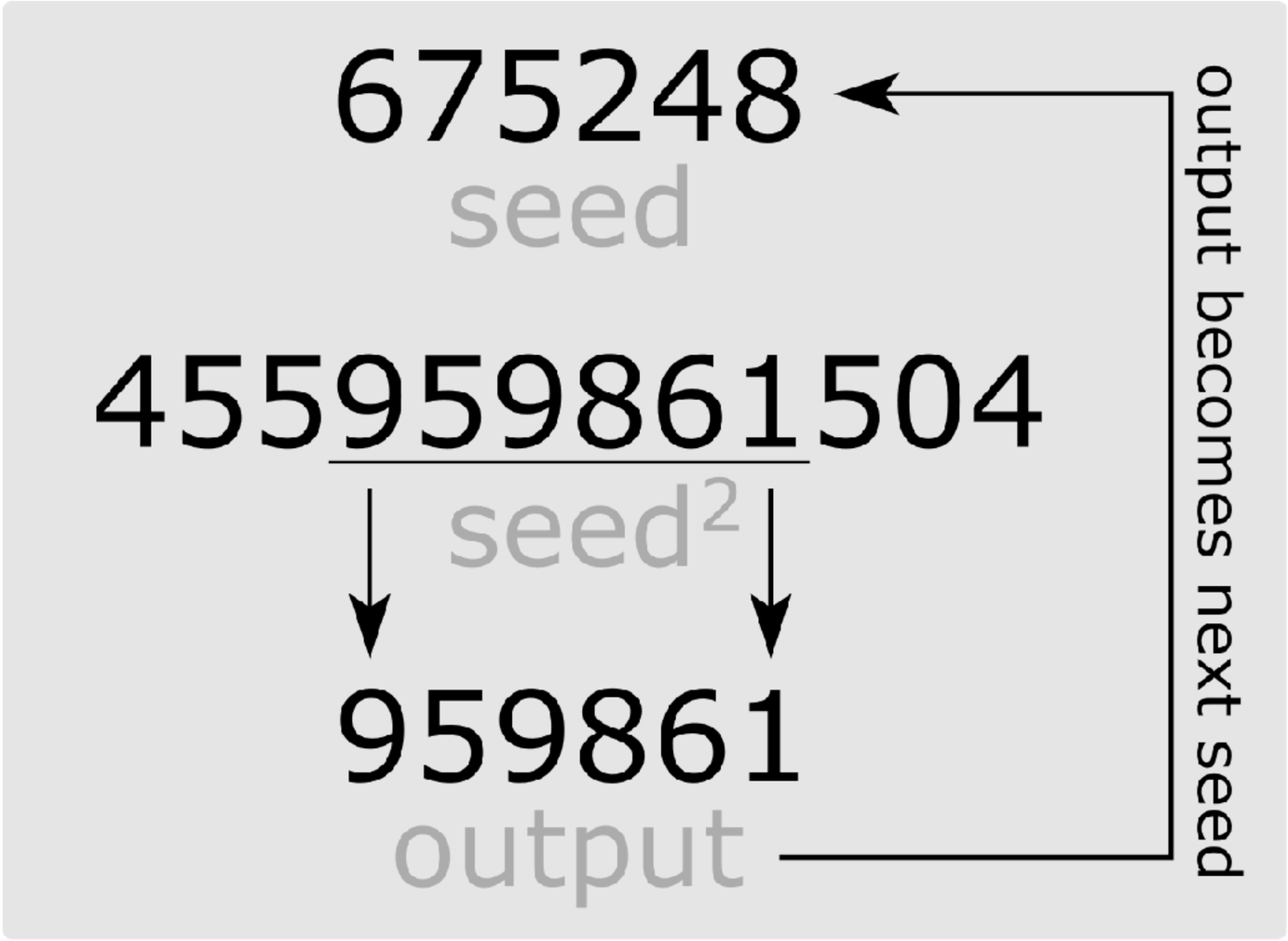
To simulate uncertain events, we generate random numbers



Monte Carlo methods are computational algorithms that rely on repeated random sampling to obtain numerical results.

https://en.wikipedia.org/wiki/Monte_Carlo_method

Pseudorandom numbers are often good enough



https://en.wikipedia.org/wiki/Middle-square_method

If you need better randomness, measure nature

Lava lamps

https://www.cloudflare.com/learning/ssl/lava-lamp-encryption/

Atmospheric noise

random.org

Dice-O-Matic

https://www.youtube.com/watch?v=7n8LNxGbZbs

Use random to generate random numbers in Python

```
>>> from random import randrange
>>> randrange(1,6)
A number from [1,2,3,4,5]
```

random.randrange for integers

Use random to generate random numbers in Python

```
>>> from random import random
>>> random()
A number from [0,1)
```

random.random for floats

Use random() to decide with a given probability

```
prob = 0.7

if random() < prob:
    # 70% this happens
else:
    # 30% this happens
```

Use choice() to select a random element

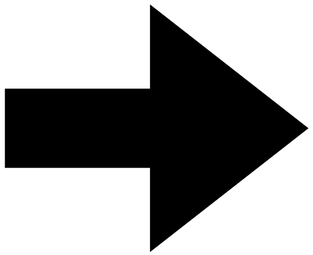
```
>>> from random import choice
>>> list = [100,200,300,400,500,600]

>>> random_item = choice(list)
A random item from the list
```

The seed makes your simulation reproducible

```
from random import *
seed(10)
print(random())
print(random())
print(random())

seed(10)
print(random())
print(random())
print(random())
```



```
0.5714025946899135
0.4288890546751146
0.5780913011344704
0.5714025946899135
0.4288890546751146
0.5780913011344704
```

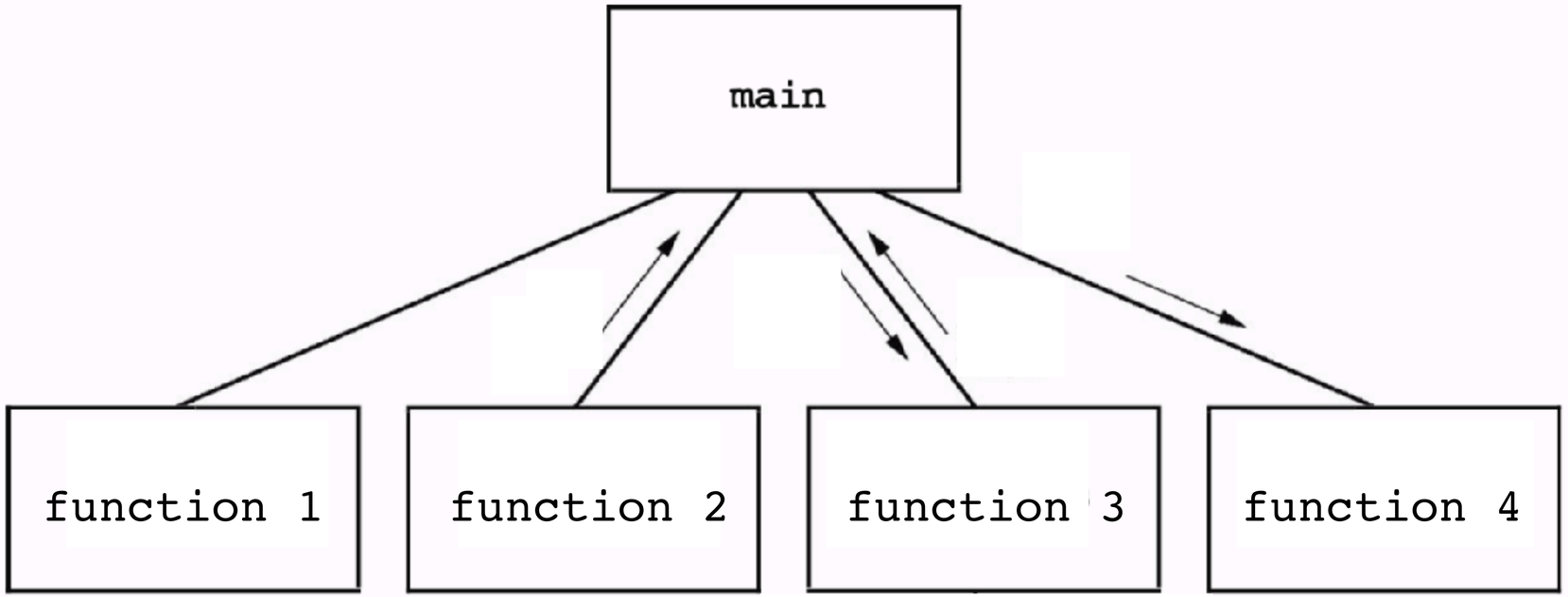
Many more: shuffle, sample, etc.

https://pynative.com/python-random-module/

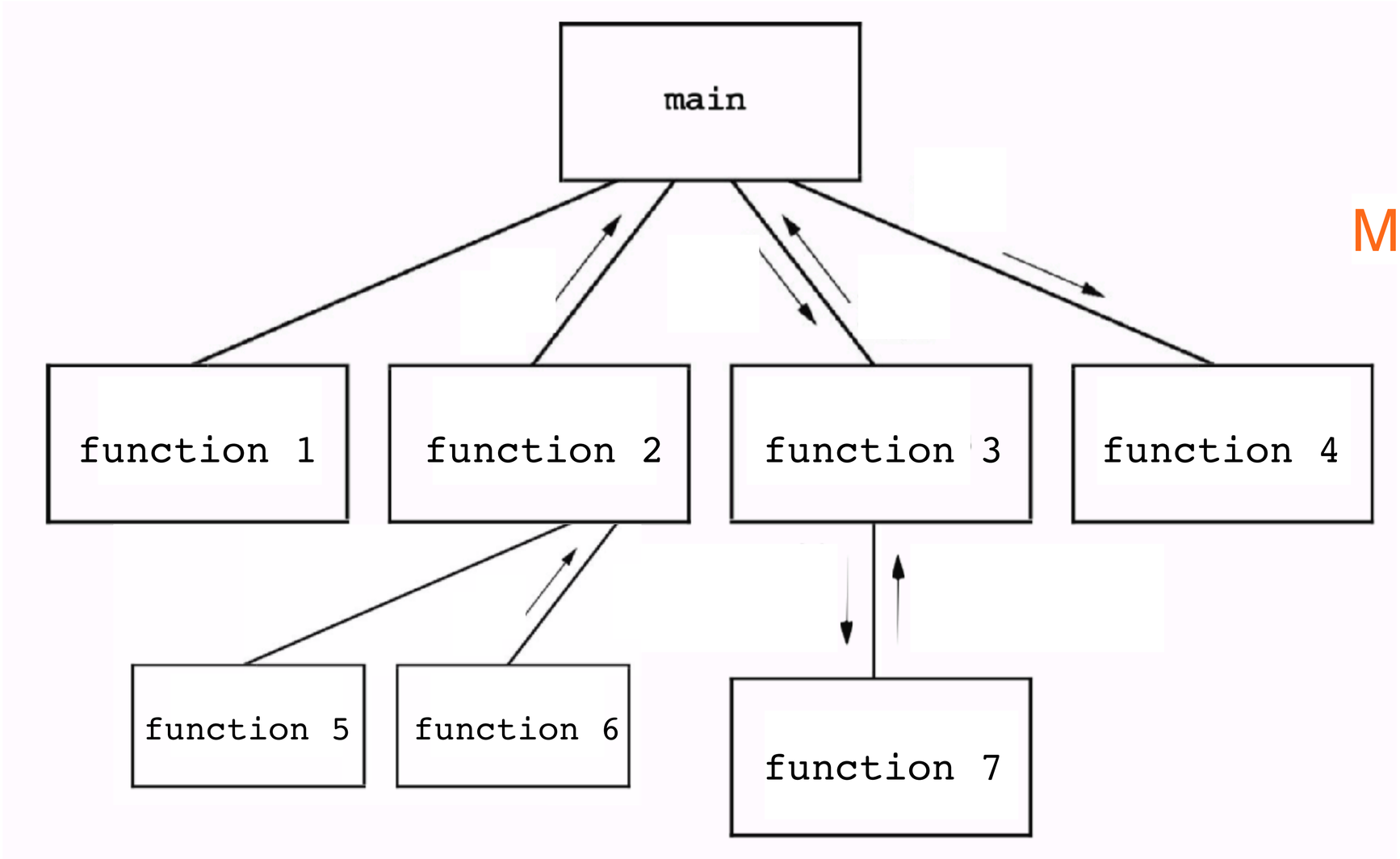

E-FORCE
RENOVART
ICODER
GEARBOX
geep
COSTA RICA
NUTRI SNACKS
Rumba
OLYMPIC CHANNEL
LONG 4
MART 0

Top-down development

Top-down is a programming style that starts with the big picture



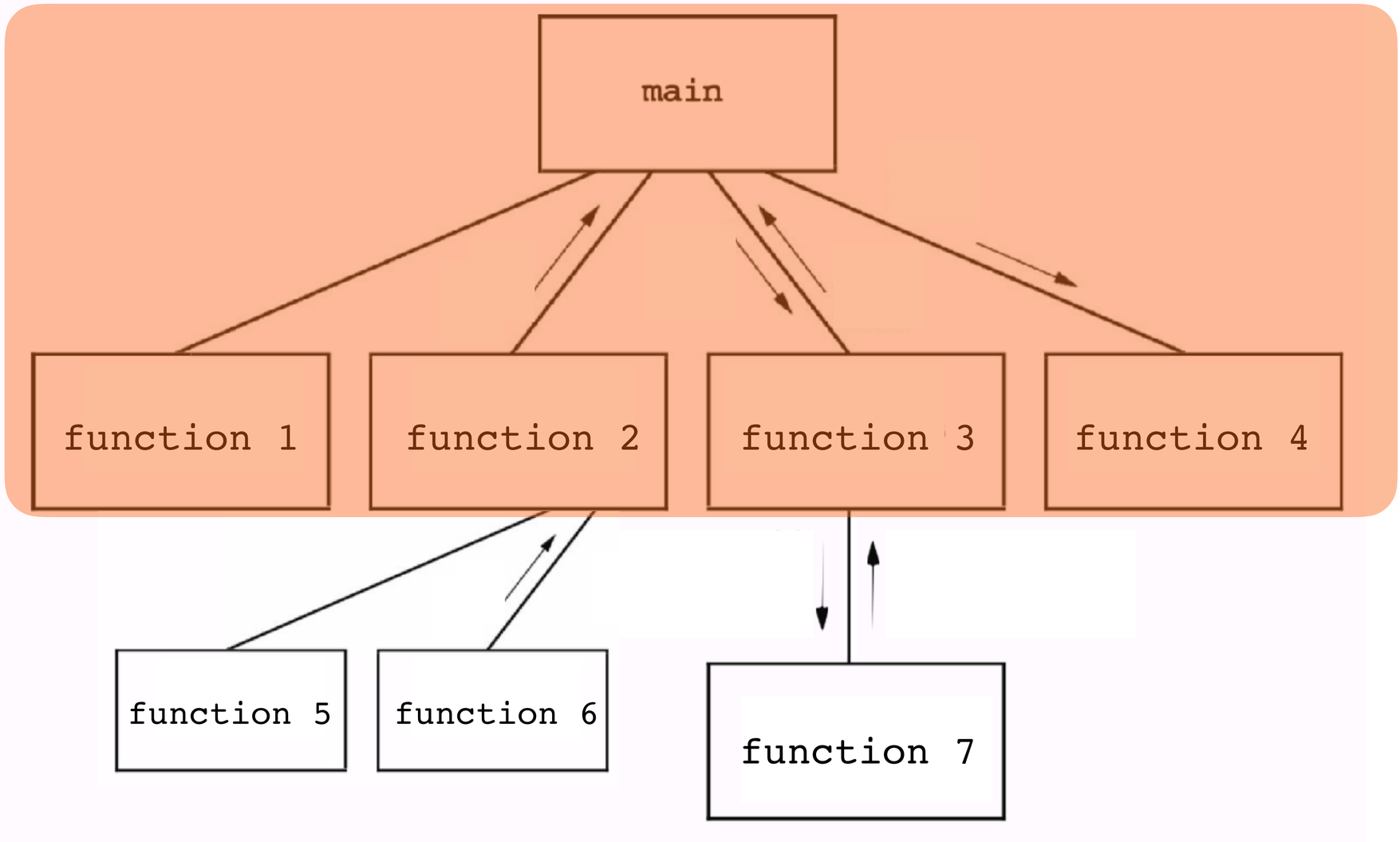
Top-down is a programming style that starts with the big picture, then divides the system into smaller pieces



Structure chart

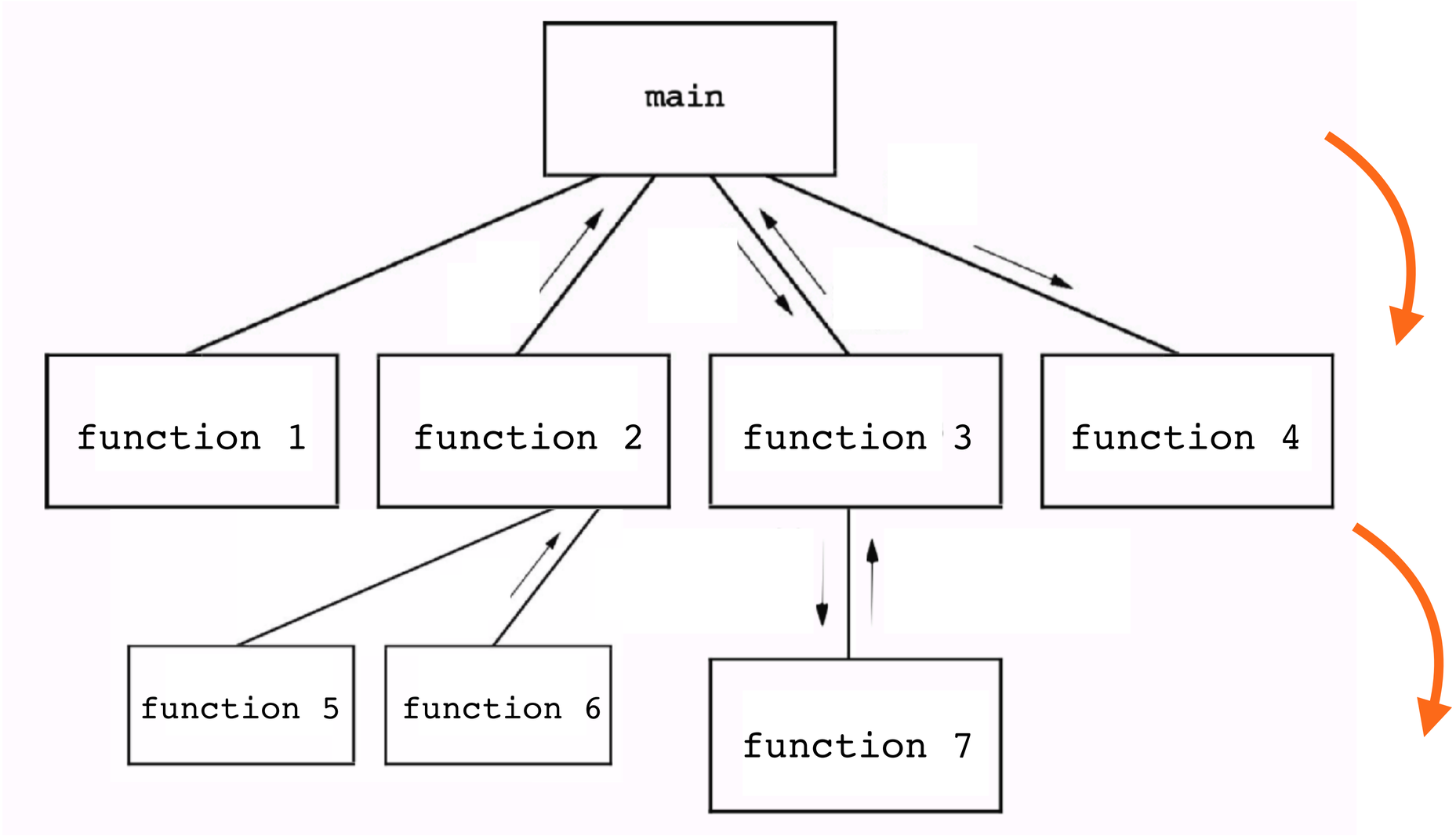
Module hierarchy chart

Top-down is a programming style that starts with the big picture, then divides the system into smaller pieces



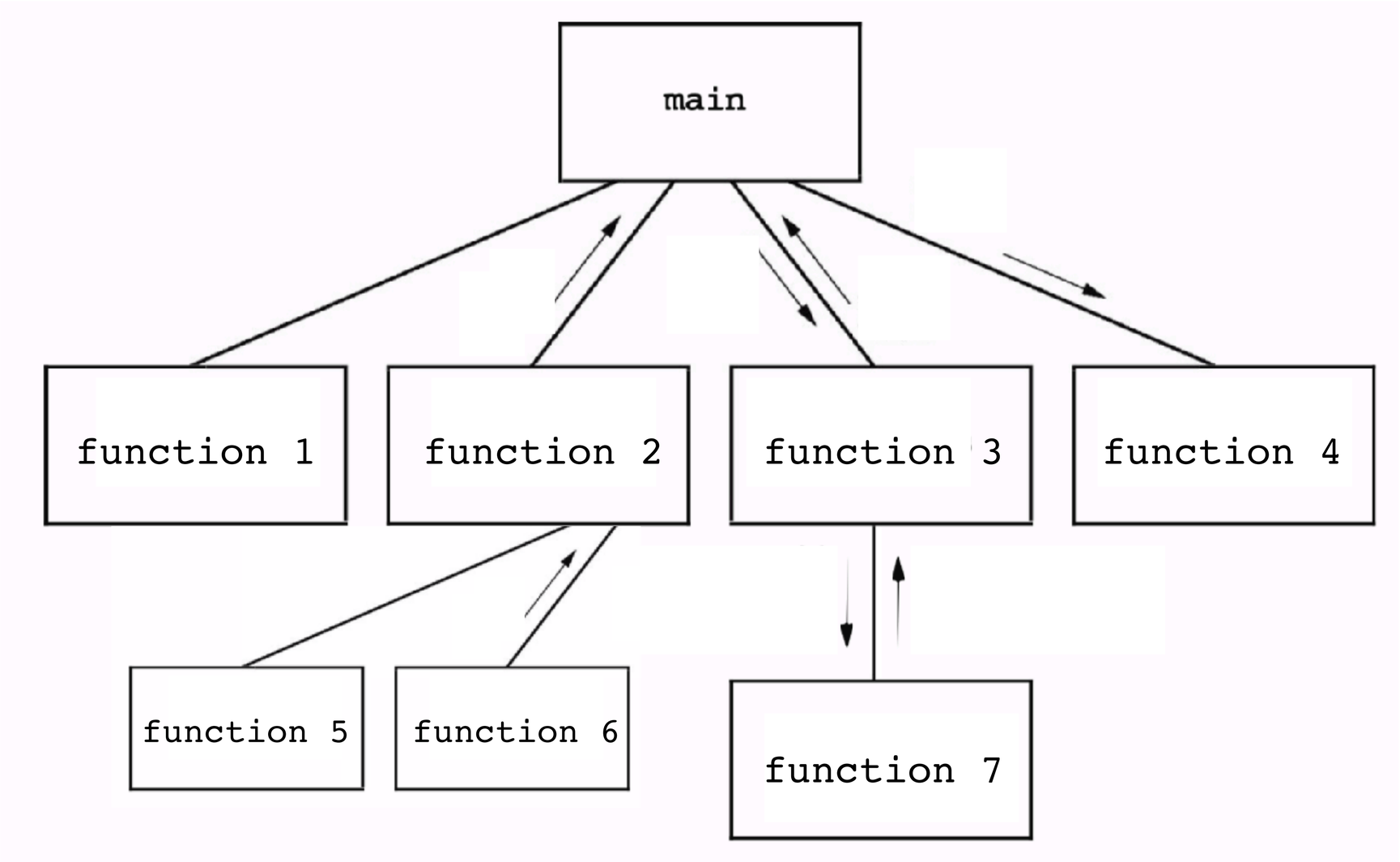
Problem decomposition

Top-down is a programming style that starts with the big picture, then divides the system into smaller pieces

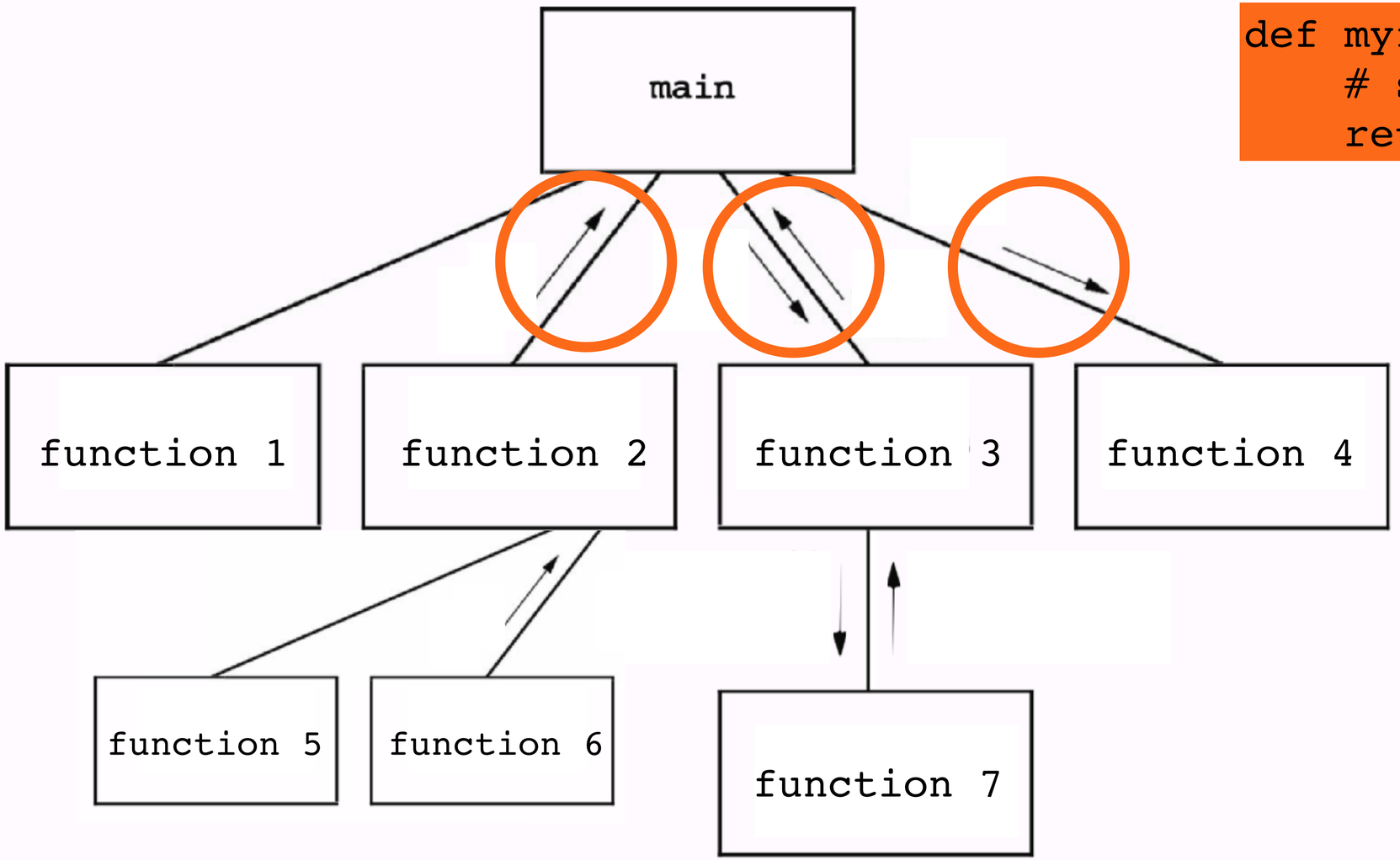


Stepwise refinement

Identifying the main characteristics of modules while ignoring other details is called abstraction

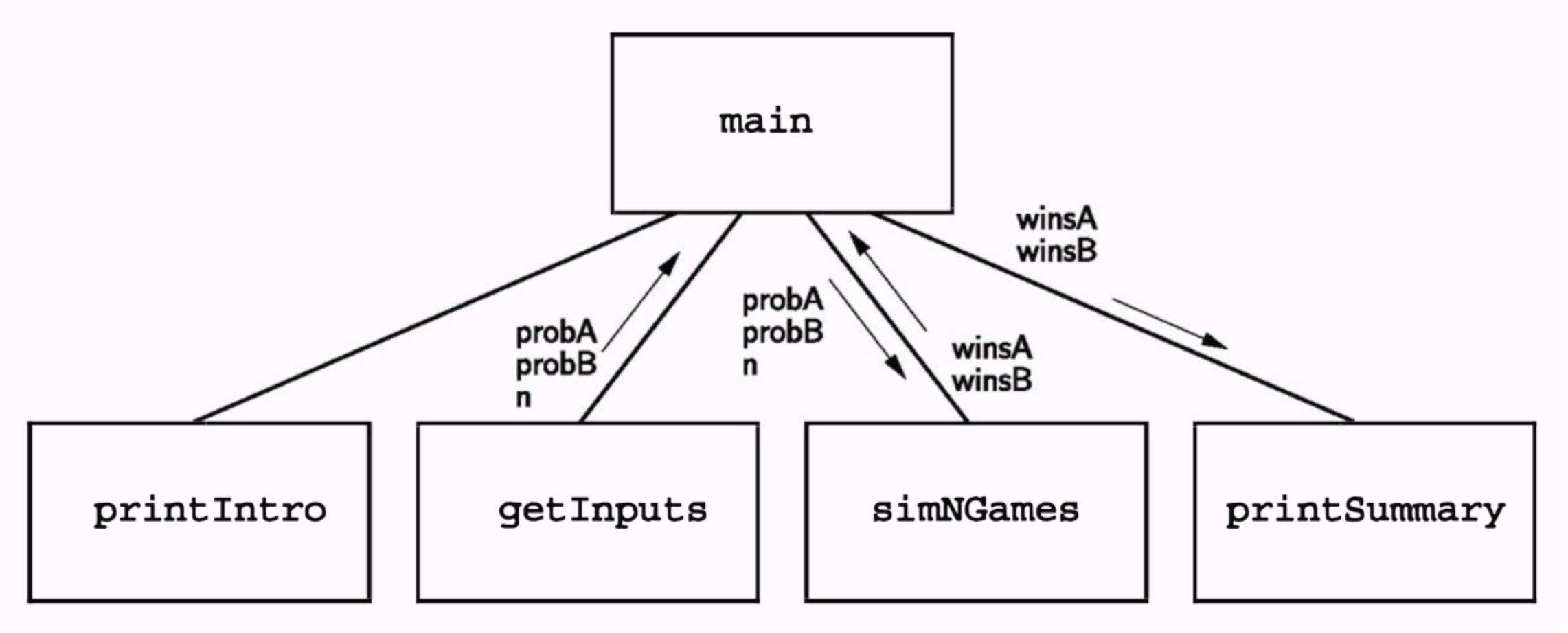


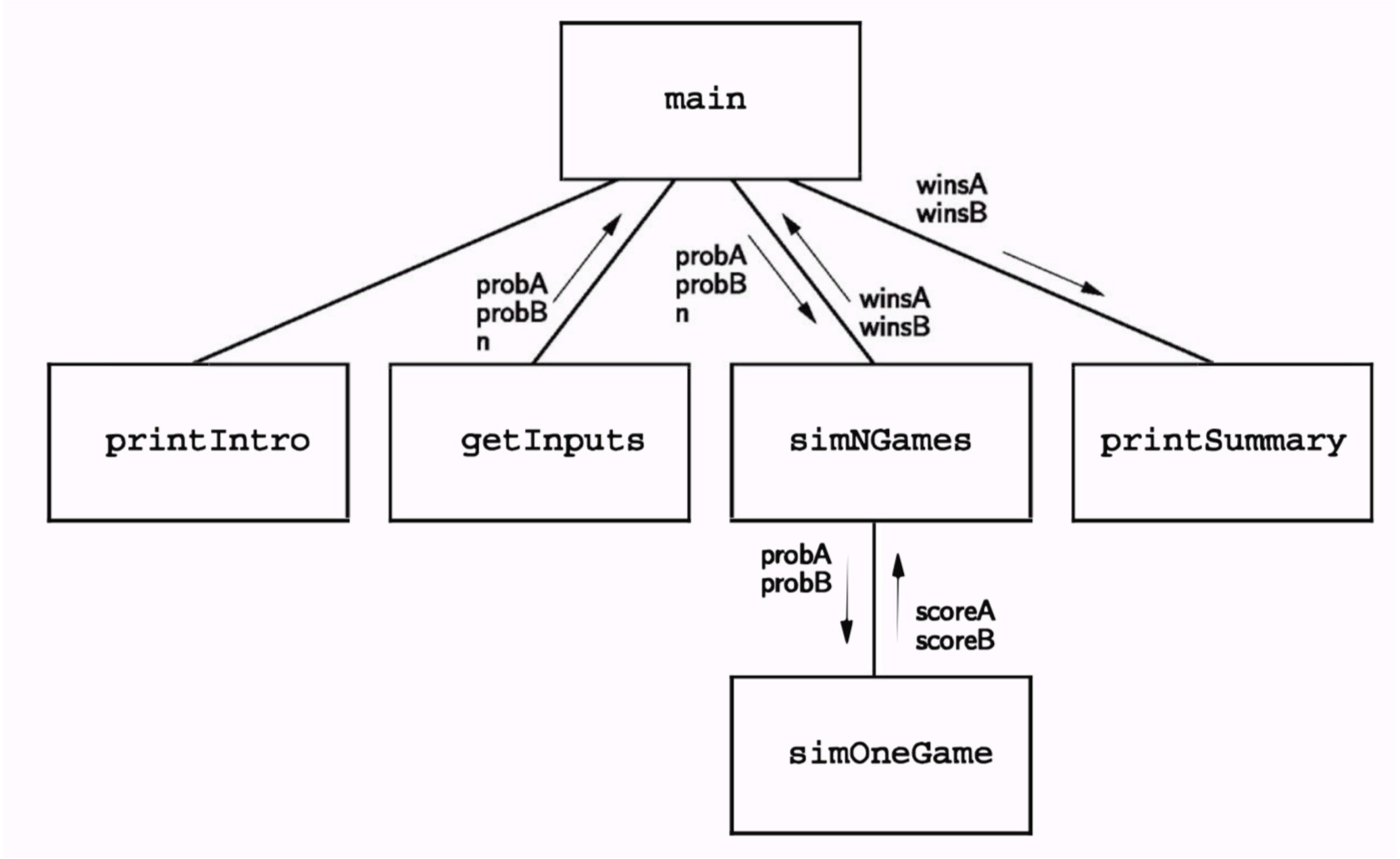
Top-down is a programming style that starts with the big picture, then divides the system into smaller pieces

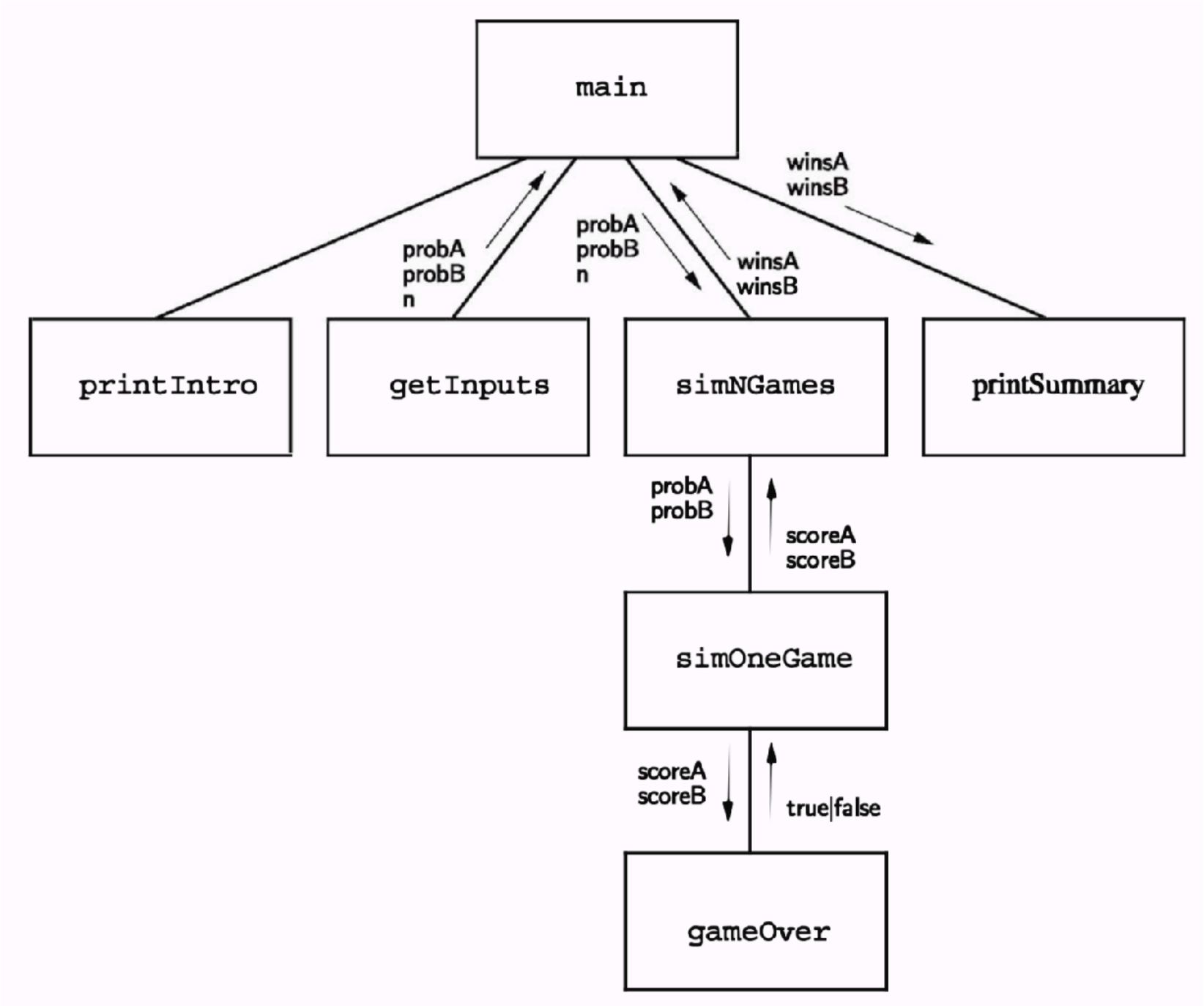



```
def myfunc(input1, input2):
    # stuff happening
    return output1, output2
```

Interface


main
printIntro
getInputs
simNGames
printSummary
probA
probB
n
probA
probB
n
winsA
winsB
winsA
winsB


main
printIntro
getInputs
simNGames
printSummary
simOneGame
probA
probB
n
probA
probB
n
winsA
winsB
winsA
winsB
probA
probB
scoreA
scoreB

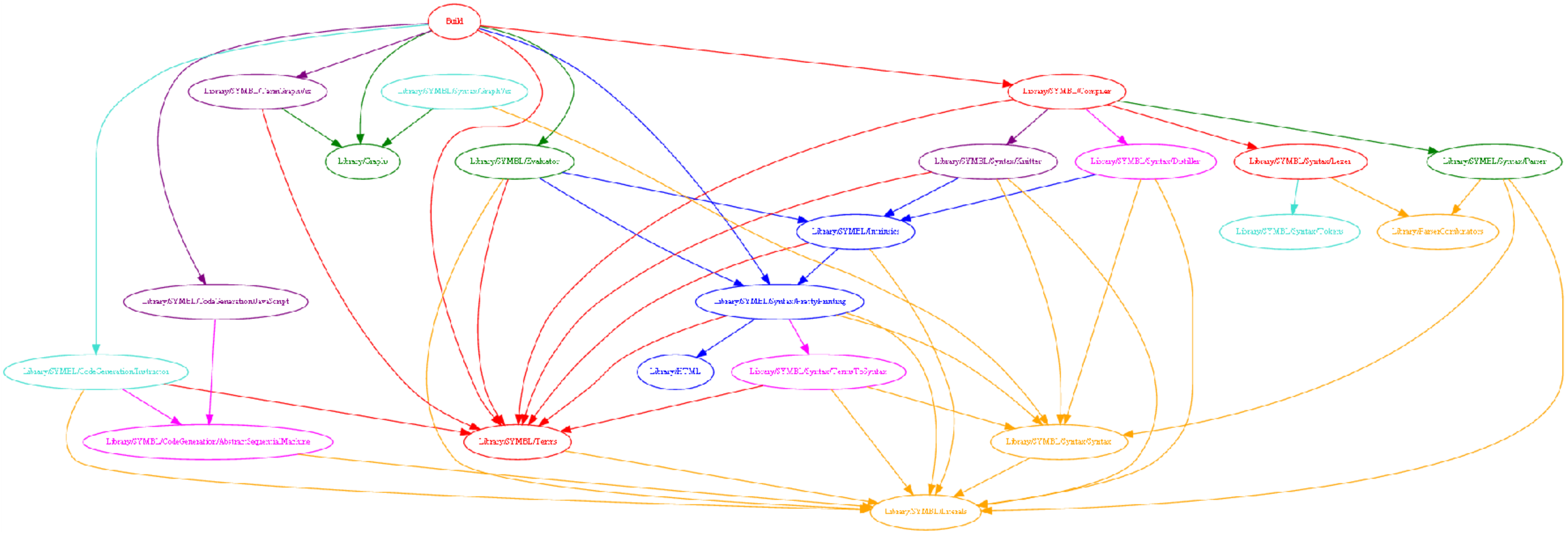

main
printIntro
getInputs
simNGames
printSummary
simOneGame
gameOver
probA
probB
n
probA
probB
n
winsA
winsB
winsA
winsB
probA
probB
scoreA
scoreB
scoreA
scoreB
true|false

Top-down development encourages planning and understanding of the whole system

Top-down development encourages planning and understanding of the whole system

Top-down development delays testing of the functional units of a system until significant design is complete

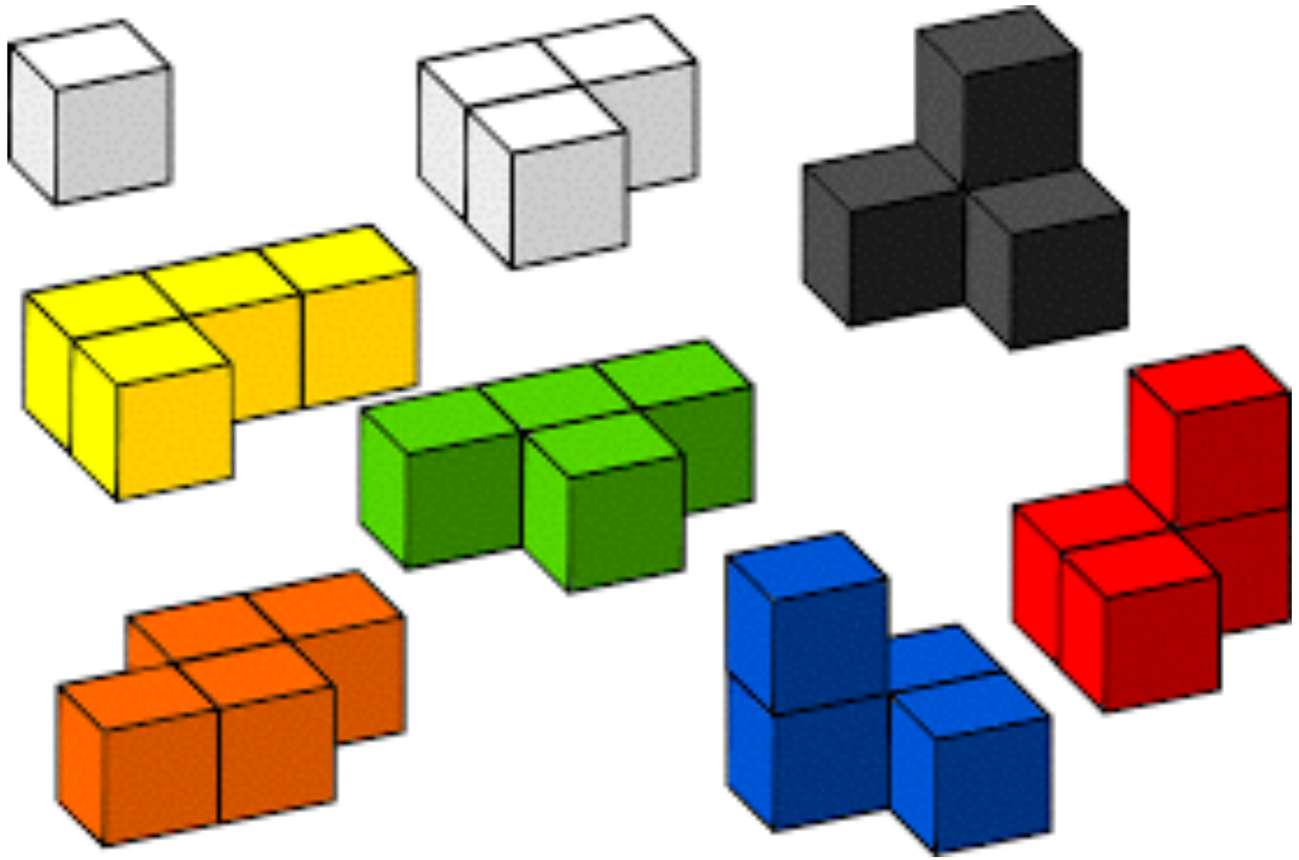
Overzealous separation into modules can make navigation through the code more difficult



Overzealous separation into modules can make navigation through the code more difficult

Bottom-up development

Bottom-up emphasizes early coding and unit testing



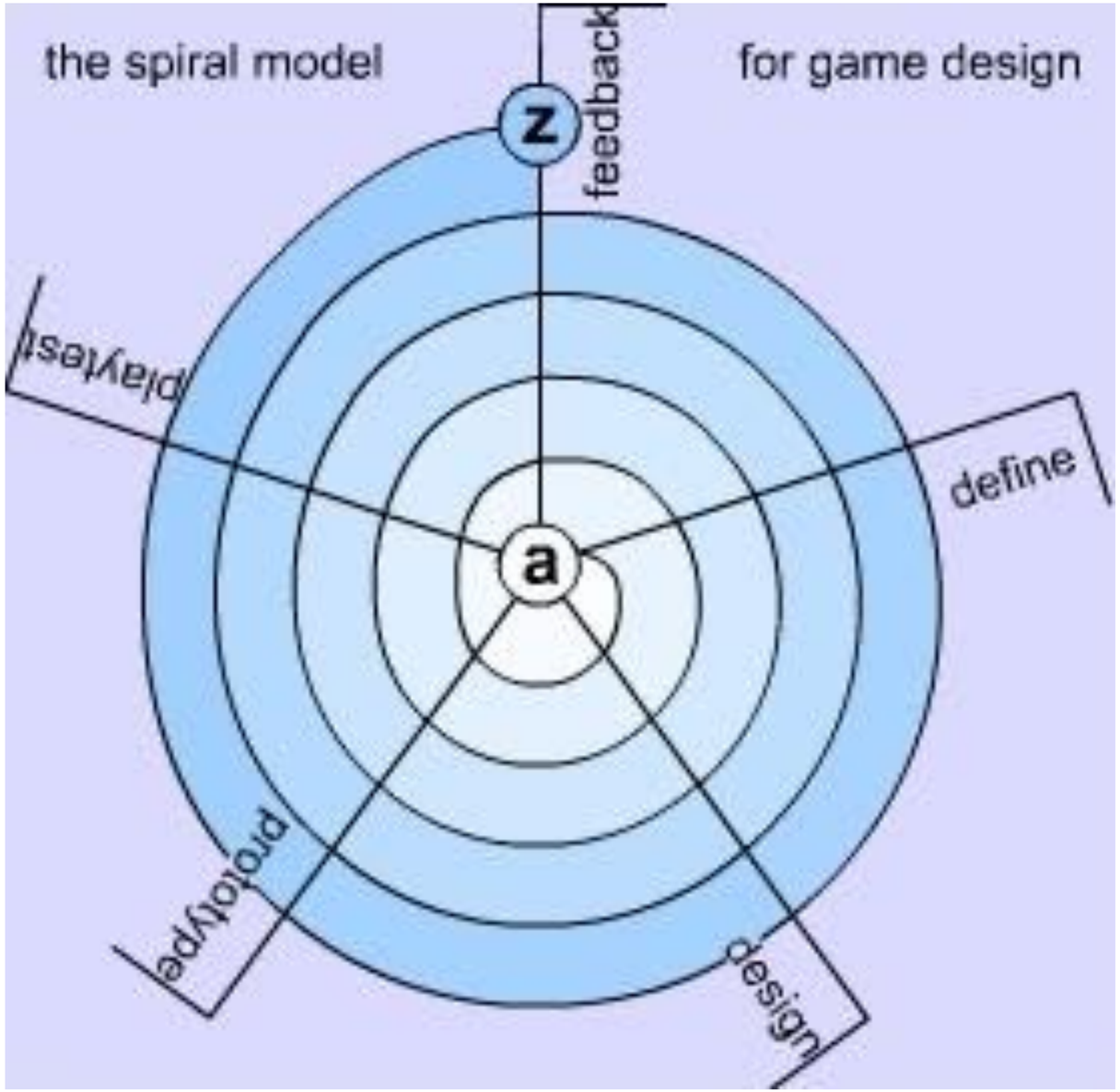
Unit testing is a level of software testing where individual units of a software are tested.

Bottom-up can lead easier to Spaghetti code

Protoyping starts with a simple version, then adds features

Useful if you get stuck, if you need early feedback, or if the specification is too complicated or unclear

Protoyping often follows a spiral development



Jupyter

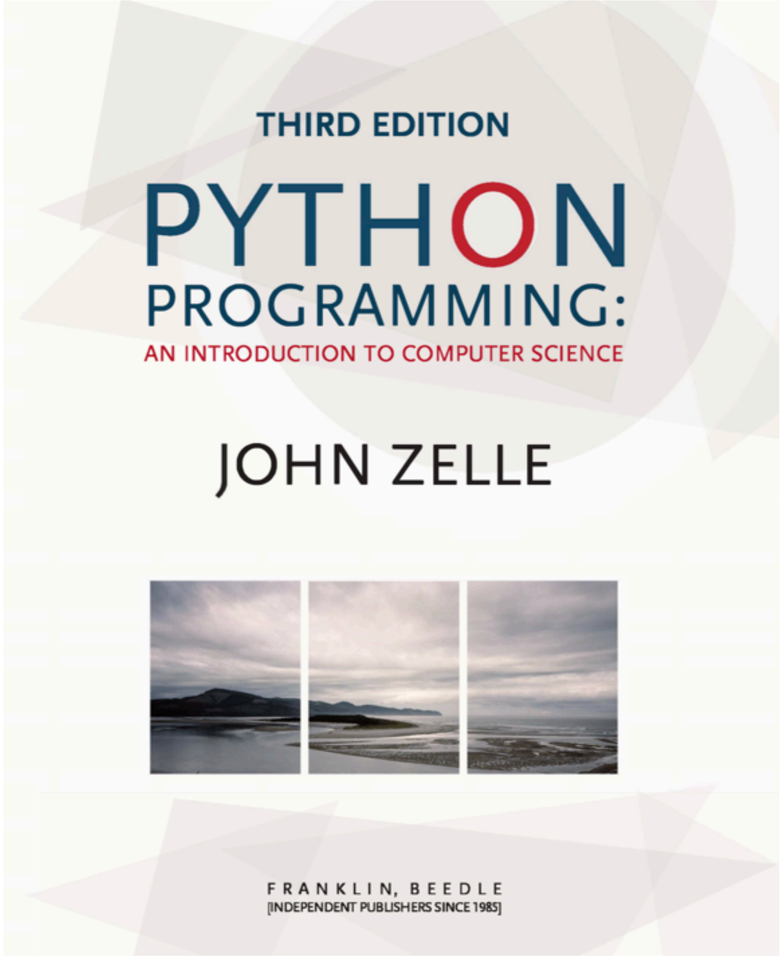
Unit testing in Python

https://www.youtube.com/watch?v=1Lfv5tUGsn8

Validating the output against a known response is called assertion.

https://realpython.com/python-testing/

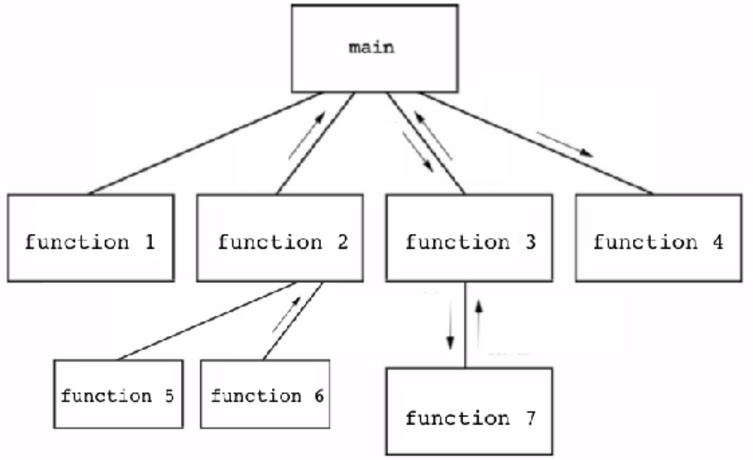
Sources and further materials for today's class



Chapter 9

What you learned today

Top-down programming



Using random numbers for simulation and program flow

Unit testing