COMP122/22 - Data Structures and Algorithms

01 Introduction

Instructor : Ke Wei 〖柯韋〗

➽ A319 ✆ Ext. 6452 ✉ wke@ipm.edu.mo

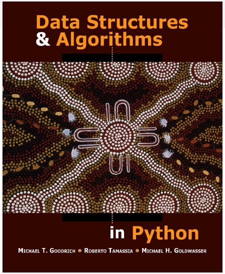
https://canvas.ipm.edu.mo/courses/4091/

Bachelor of Science in Computing, School of Applied Sciences, Macao Polytechnic Institute

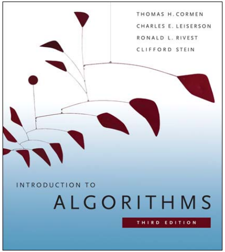
January 6, 2022

Textbooks and References

Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser.
Data Structures and Algorithms in Python, 1st Edition.
Wiley, 2013.
ISBN-13 978-1-118-29027-9
Textbook.


Data Structures
& Algorithms
in Python
Michael T. Goodrich • Roberto Tamassia • Michael H. Goldwasser

Thomas H. Cormen., Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.
Introduction to Algorithms, International Edition (3rd Edition).
MIT Press, 2009.
ISBN-13 978-0-262-03384-8
Reference book.



Outline

1. Textbooks and References

2. Learning Module Overview

3. Data Structures and Algorithms

4. Python Programming

Learning Module Overview

- This learning module provides an introduction to data structures and algorithms, including their design, analysis, and implementation.
- *Python* is the programming language for the implementation.
- The course is divided into the following sections:
    - Python programming fundamentals,
    - linear structures — arrays and linked lists,
    - abstract data types — stacks, queues, double-ended queues (deques), priority queues and associative arrays,
    - fundamental algorithm analysis — the Big-$\mathcal{O}$ notation,
    - recursion and mathematical induction,
    - trees, binary trees and applications — heaps and search trees,
    - hash tables,
    - sorting algorithms, and finally
    - some advanced algorithms on graphs.

Data Structures

A data structure is a precise way to organize related data in order to solve a problem or provide a function.

	2		3	8				7
			9					5
		6				3		
9				1				2
2			5	3	8			9
	5		2				7	3
		5				1		
7	3							
				4				

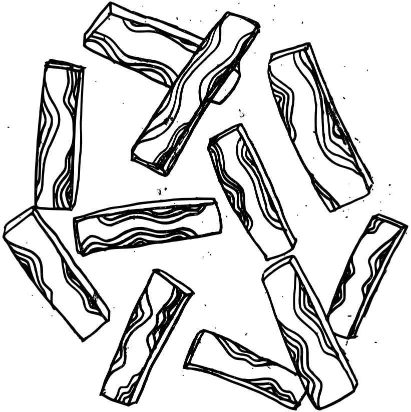
How to organize these numbers in computer memory, so that your program knows the two circled numbers are on the same row?

Angel	5124891	98.5
Maya	5033887	80.0
Adi	5122321	90.5
Ivan	5098980	68.0
Leo	5021747	71.0
Luca	5544787	99.0
Nico	5169327	89.5
Filip	5291871	77.0
Tim	5533982	89.5
Olivia	5098980	95.0
Lily	5419019	59.5

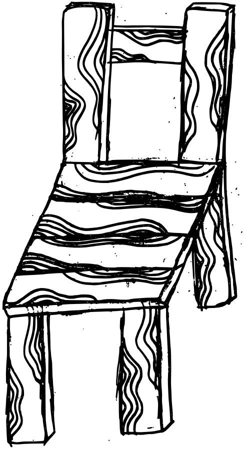
How to maintain the table, so that the highest mark can be easily returned, or a new entry can be efficiently inserted?

What Is a Structure?

A mess:



A structure:



A Data Structure

We store *names*, *contacts* and *marks* respectively in 3 arrays, items with the same index are related.

89.5	59.5	5098980	Lily
Maya	5291871	Luca	5098980
	5033887	Tim	98.5
Leo	Adi	99.0	95.0
68.0	5122321		5533982
5124891	Nico	80.0	77.0
Angel	71.0	5419019	5021747
90.5	Olivia	5169327	Filip
Ivan	89.5	5544787	

Unstructured data

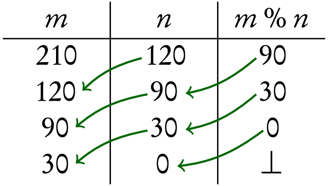
names	*contacts*	*marks*
Angel	5124891	98.5
Maya	5033887	80.0
Adi	5122321	90.5
Ivan	5098980	68.0
Leo	5021747	71.0
Luca	5544787	99.0
Nico	5169327	89.5
Filip	5291871	77.0
Tim	5533982	89.5
Olivia	5098980	95.0
Lily	5419019	59.5

Structured data

Algorithms

- An algorithm is the precise steps for solving a problem. It is similar to a program, but more abstract.
- The Greatest Common Divisor: $gcd(m, n)$ is the greatest integer that divides both m and n, provided $m > 0$ and $n \geqslant 0$.
- Euclid's Algorithm

```
def gcd(m, n):
    while n != 0:
        m, n = n, m % n
    return m
```



m	n	$m \% n$
210	120	90
120	90	30
90	30	0
30	0	$\perp$

Relation between Algorithms and Data Structures

- Usually, an algorithm requires some data structures to help store and retrieve information.
- On the other hand, to maintain the integrity of a data structure requires some specific (often complex) steps — algorithms. For example,

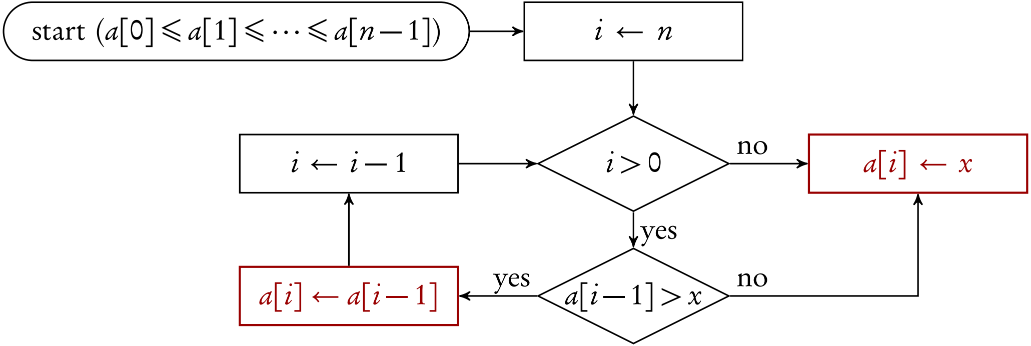
  how do we add a record to the arrays of names, contacts and marks?

- The algorithms in this course are mainly to maintain structures of data *collections* — how to

  add *get* *remove*

  items to and from the collections.

Insertion Algorithm of Ordered Arrays



Fundamental Python Programming Concepts

In order to implement the data structures and algorithms in this course, you need to understand the main structures of a Python program including:

- Variables and expressions
- Functions
- Objects and classes
- Lists and mutable sequences
- Tuples, strings and immutable sequences
- Assignments and unpacking
- Decision structures (if-then-else)
- Iteration structures (while, for)

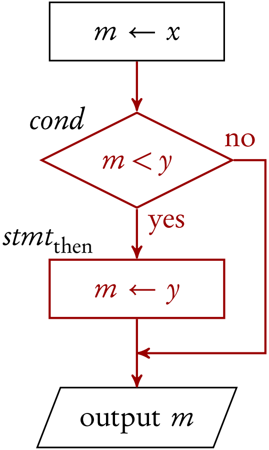
Installing the Python Programming Environment

- The current Python 3.10 interpreter and documents can be found at `https://www.python.org/`.
- The Windows installer
  for x86-64: `https://www.python.org/ftp/python/3.10.1/python-3.10.1-amd64.exe`, and
  for x86-32: `https://www.python.org/ftp/python/3.10.1/python-3.10.1.exe`.
- After the installation, we can use the IDLE (Python's Integrated DeveLopment Environment) to interactively write and run Python statements; load, edit and run Python source programs.
- We can also use Eclipse as the environment, with the PyDev plugin at `http://www.pydev.org/`.
- The update site of PyDev for Eclipse: `http://www.pydev.org/updates`.

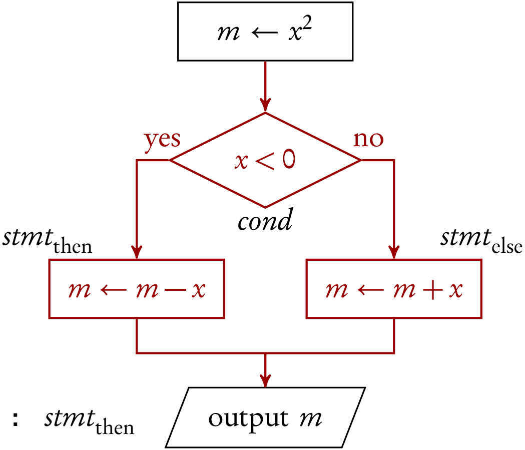
A Python Program

```
class Student:    # a class def
    def __str__(self):    # a method def
        return 'Name:␣'+self.name+',␣mark:␣'+str(self.mark)

def input_students(n):    # a function def
    ls = []    # a list
    for i in range(1, n+1):    # a for-each loop
        print('Student␣{}.'.format(i))
        s = Student()
        s.name, s.mark = input('␣␣Name:␣'), float(input('␣␣Mark:␣'))
        ls.append(s)
    return ls    # the result of the function

if __name__ == '__main__':    # the main program
    print([str(s) for s in input_students(3)])    # prints a list comprehension
```

Control Flow Statements (if-then, if-then-else)

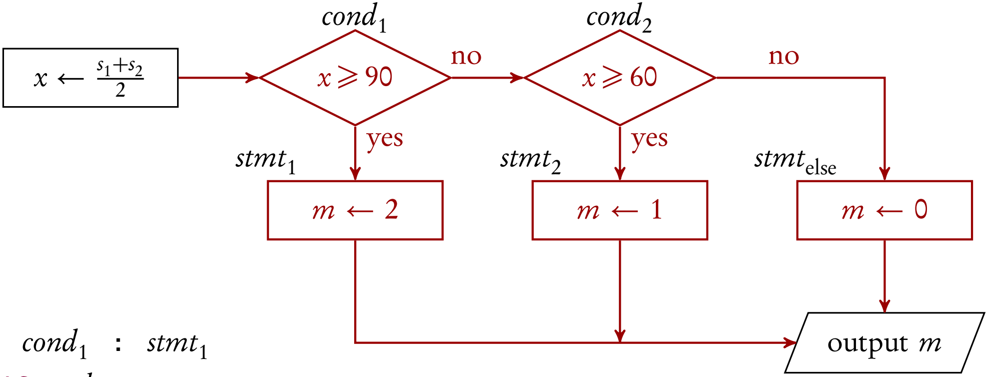


`if` $cond$: $stmt_{\text{then}}$



`if` $cond$: $stmt_{\text{then}}$
`else` : $stmt_{\text{else}}$

Control Flow Statements (if-then-else if-else)



if $cond_1$: $stmt_1$

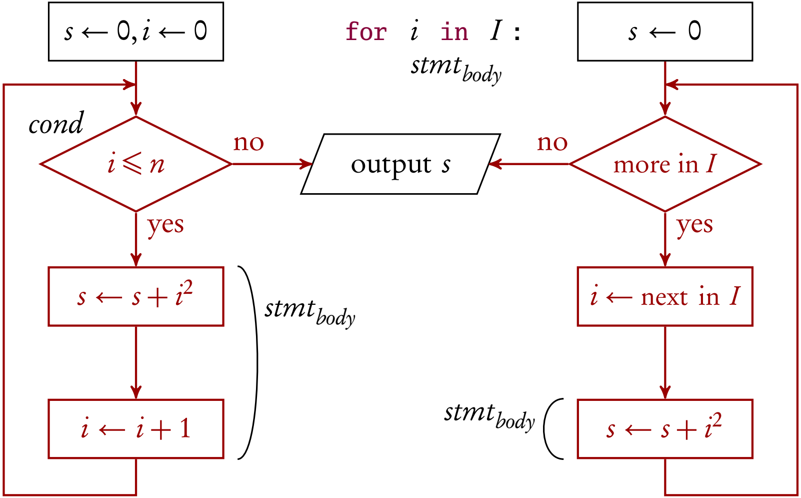
elif $cond_2$: $stmt_2$

else : $stmt_{\text{else}}$

Control Flow Statements (while, for)

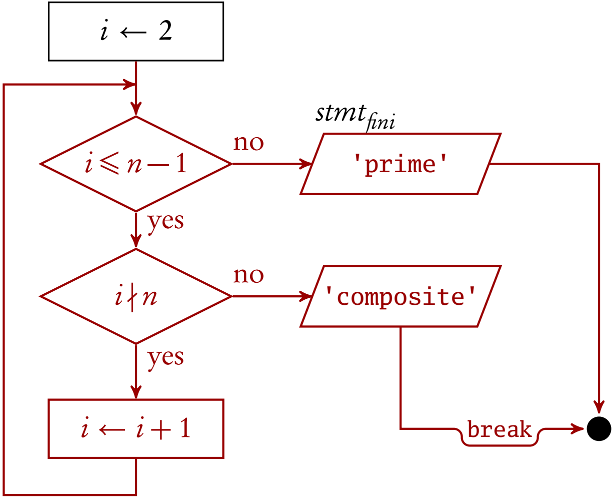
```
while  cond :
    stmt_body
```

```
for  i  in  I :
    stmt_body
```



The else Clause for Loop Statements

```
while  cond :
     stmt_body
else:
     stmt_fini
```



```
i = 2
while i <= n-1:
    if n%i == 0:
        print('composite')
        break
    i = i+1
else:
    print('prime')
```