Introduction to Programming with
Scientific Applications

Gerth Stølting Brodal
Department of Computer Science
Aarhus University

Course evaluation 2018
"Den første forelæsning var meget skræmmende og overvældende"

Course description – kursuskatalog.au.dk/en/course/123832/

Introduction to Programming with Scientific Applications

Description of qualifications

After the course the participants will have knowledge of principles and techniques for systematic **construction** of **programs**.

At the end of the course, the participants will be able to:

- apply constructions of a common programming language,
- develop **well-structured** programs and perform **testing** and **debugging** of these,
- explain fundamental programming concepts and basic algorithmic techniques,
- apply standard **tools for scientific applications**,
- use the documentation for a programming language and available software packages.

Contents

The course gives an introduction to programming with scientific applications. Programming concepts and techniques are introduced using the **Python** programming language. The programming concepts are **illustrated in other programming languages**. The following content is included.

Basic programming constructs: Data types, operators, variables, flow of control, conditionals, loops, functions, recursion, scope, exceptions. *Object orientation*: Abstract data types, classes, inheritance, encapsulation. *Basic algorithmic techniques*: Sorting, binary search, dynamic programming. *Systematic development of programs*: Testing and debugging. File-based input/output, numerical analysis, functional programming. Scientific computing using standard packages for Python.

ECTS   10

Hours - weeks - periods
   Lectures 2 x 2 hours/week
   TA sessions 1 x 3 hours/week
   Study café 3 x 1 hour/week

Language of instruction
   Danish

Instructor
   Gerth Stølting Brodal

Academic prerequisites
   (Some) Linear algebra

Exam

   5 hour programming
   Aid: Computer and Internet, headphones, 7-point grading scale

Prerequisites for examination participation
   Submission and approval of 10 mandatory assignments and submission of **1 implementation project**

Notes   Grade reflects an overall assessment of implementation project and written examination. Project counts 20% and written exam counts 80%

Lecturer

Name	Gerth Stølting Brodal
Research	Algorithms and Data Structures (Computer Science)
Teaching	
2018 -	BSc course on Introduction to Programming with Scientific Applications
2003 -	BSc course on Introduction to Algorithms and Data Structures
1999 - 17	MSc courses on Computational Geometry, Algorithm Engineering, Advanced Data Structures, External Memory Algorithms and Data Structures
Python	Advanced Beginner

Question – Primary Education?

a) Mathematics
b) Mathematics-Economics
c) Data Science
d) Chemestry
e) Physics
f) Other Science-Technology
g) Other

Question – Programming languages you know?

Question – Programming experience?

For the programming language you know best (if any) please state you proficiency level within the language.

a) None
b) Fundamental awareness (basic knowledge)
c) Novice (limited experience)
d) Intermediate (practical application)
e) Advanced (applied theory)
f) Expert (recognized authority)

Some course practicalities
Primary lecture material = slides

Week	Monday	Tuesday	Wednesday	Thursday	Friday
5	F1	no TA class	F2		
6	F3	TØ1	TØ1 / F4	TØ1	
7	F5	TØ2	TØ2 / F6	TØ2	
8	F7	TØ3	TØ3 / F8	TØ3	
9	F9	TØ4	TØ4 / F10	TØ4	
10	F11	TØ5	TØ5 / F12	TØ5	
11	F13	TØ6	TØ6 / F14	TØ6	
12	F15	TØ7	TØ7 / F16	TØ7	
13	Easter break				
14		-	- / F17	TØ8	
15	F18	TØ8	TØ8 / F19	TØ9	
16	F20	TØ9	TØ9 / F21	TØ10	
17	F22	TØ10	TØ10 / F23	TØ11	Kapsejlads?
18	F24	TØ11	TØ11 / F25	TØ12	
19	F26	TØ12	TØ12 / -	Ascesion Day	
20	F27	TØ13	TØ13 / -	TØ13	

PBA Aud

Aud E

	Monday	Tuesday	Wednesday	Thursday	Friday
8:15-9:00		TA meeting			
9:15-10:00	Studiecafé		Studiecafé		Studiecafé
10:15-11:00	Lecture		Lecture		
11:15-12:00	Lecture		Lecture		
12:15-13:00			DV		
13:15-14:00			KE		
14:15-15:00		MA1		MA2	
15:15-16:00		MØ		Mix	
16:15-17:00		FY			
17:15-18:00					

Course page on Brightspace and GitHub

Course text book – optional

John V. Guttag: **Introduction to Computation and Programming Using Python, Third Edition With Application to Computational Modeling and Understanding Data**. Third Edition. 664 pages. MIT Press, 2021.

- *[Guttag, 2nd Ed., page 8] "The reader should be forewarned that this book is by no means a comprehensive introduction to Python". 3rd Ed. added about 80 pages on introduction to Python.*

- *Covers all basic features of Python enabling you to deal with data in Chapters 1-10 (212 pages) - remaining chapters are applications*

- *Other resources: Google, stackoverflow, Python.org, YouTube, ...*
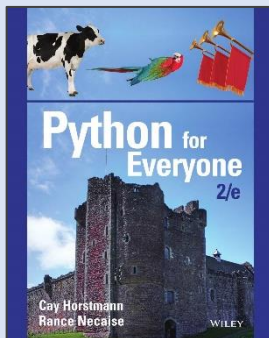
Comparison to a standard text book on the *programming language* Python by Cay Horstmann and Rance Necaise:

Topic **recursion** is covered by Guttag on page 123 (2nd edition on page 50), Horstmann and Necaise do it on page 611

Some other books on Python
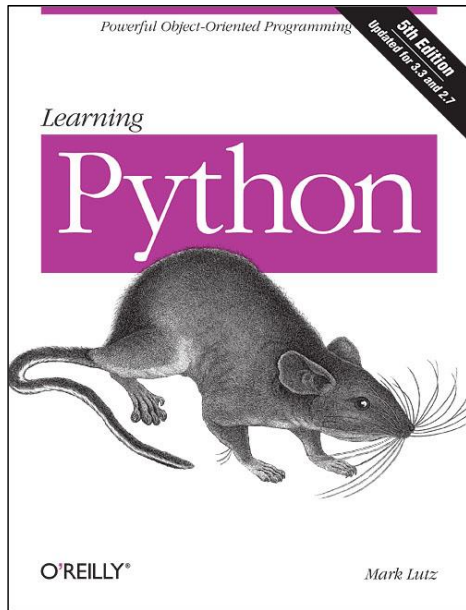
| O'Reilly, 2013 1684 pages | Wiley, 2016 752 pages | Addison-Wesley, 2015 794 pages | Wiley, 2013 580 pages | Franklin & Beedle, 2016 552 pages |
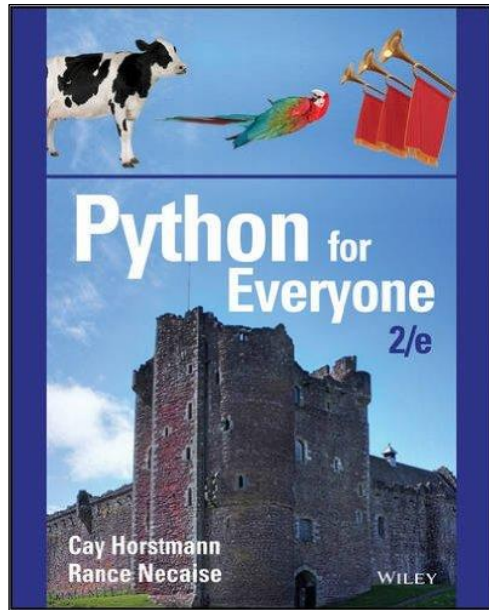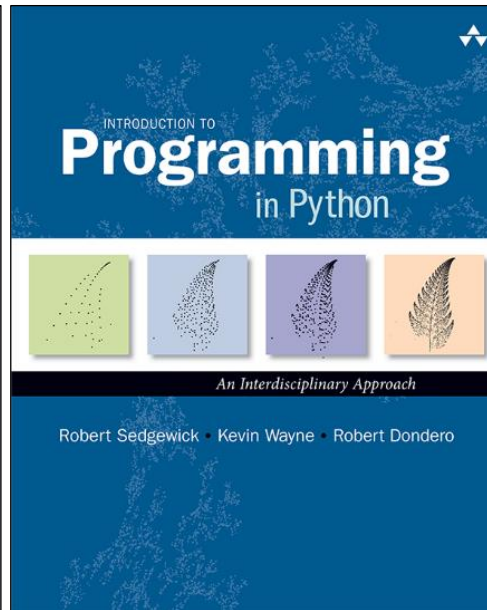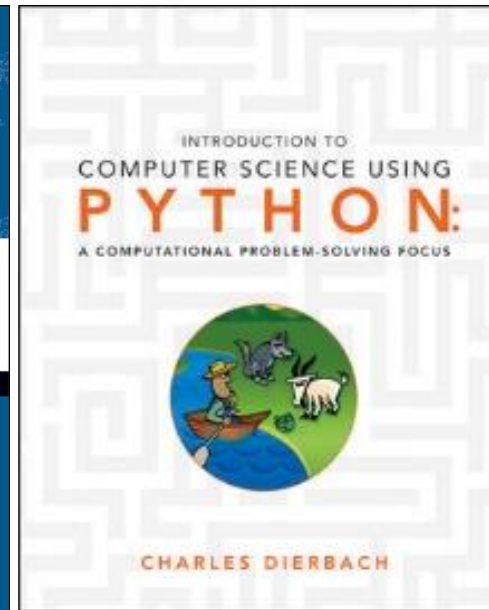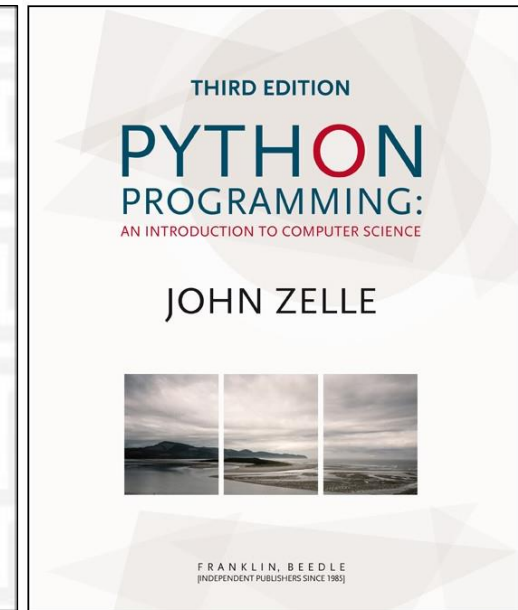
... numerous online introduction texts/courses/videos on Python

Two Python programs

A Python program

Python shell

```
> x = 7
> print(x * x)
| 49
```

Memory

x | 7

- 7 is an *integer literal* – in Python denoted an "`int`"

- x is the name of a *variable* that can hold some value

- $=$ is assigning a value to a variable

- $*$ denotes multiplication

- `print` is the name of a built-in *function*,
  here we call `print` to print the result of 7 * 7

- A program consists of a sequence of *statements*, executed sequentially

Question – What is the result of this program?

```
Python shell
> x = 3
> y = 5
> x = 2
> print(x * y)
```

x assigned new value

Memory

a) 10

b) 15

c) 25

d) [15, 10]

e) Error

f) Don't know

old x   3

y   5

new x   2
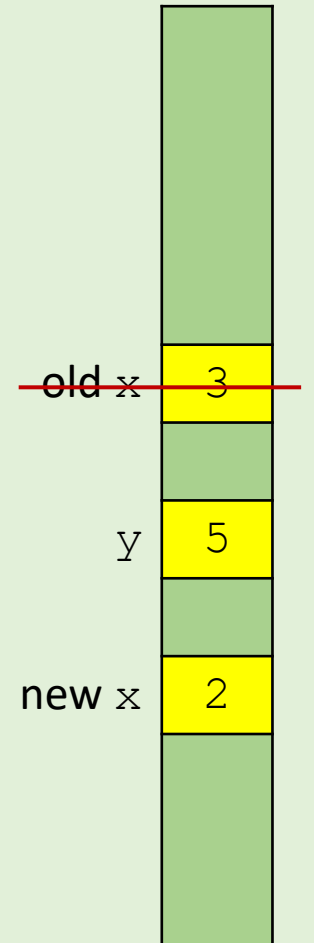
Another Python program using lists

Memory

```
Python shell
> a = [13, 27, 7, 42]
> print(a)
| [13, 27, 7, 42]
> print(a[2])
| 7
```

13	a[0]
27	a[1]
7	a[2]
42	a[3]

a

- `[13, 27, 7, 42]` is a *list* containing four integers
- `a[2]` refers to the entry in the list with *index* 2
  (the first element has index 0, i.e. `a[2]` is the 3rd element of the list)
- Note that `print` also can print a list
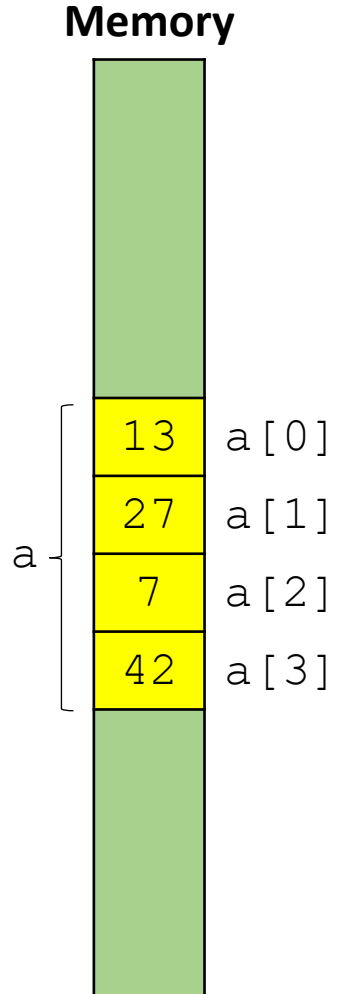
Question – What is the result of this program?

Python shell

```
> a = [3, 5, 7]
> print(a[1] + a[2])
```

a) 8

b) 10

c) 12

d) 15

e) Don't know

Memory

3	a[0]
5	a[1]
7	a[2]

a
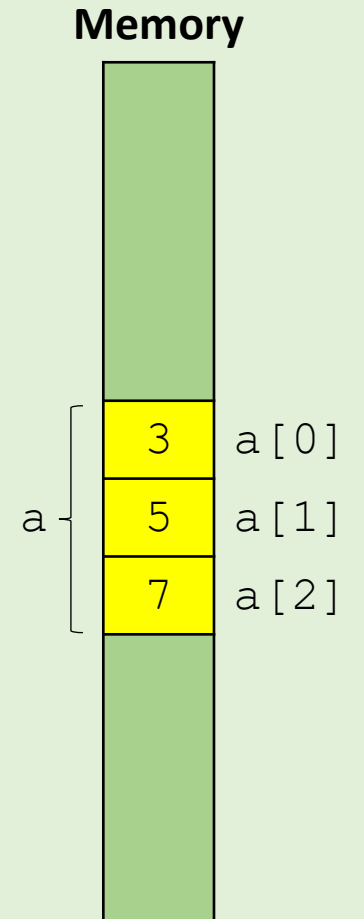
Why Python ?

⚠ the next slides will be technical

TIOBE Index January 2024

Jan 2024	Jan 2023	Change		Programming Language	Ratings	Change
1	1			Python	13.97%	-2.39%
2	2			C	11.44%	-4.81%
3	3			C++	9.96%	-2.95%
4	4			Java	7.87%	-4.34%
5	5			C#	7.16%	+1.43%
6	7	^		JavaScript	2.77%	-0.11%
7	10	^		PHP	1.79%	+0.40%
8	6	v		Visual Basic	1.60%	-3.04%
9	8	v		SQL	1.46%	-1.04%
10	20	⌃		Scratch	1.44%	+0.86%

The TIOBE Programming Community index is an indicator of the *popularity of programming languages*. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.　　　　　www.tiobe.com

Popularity of programming languages
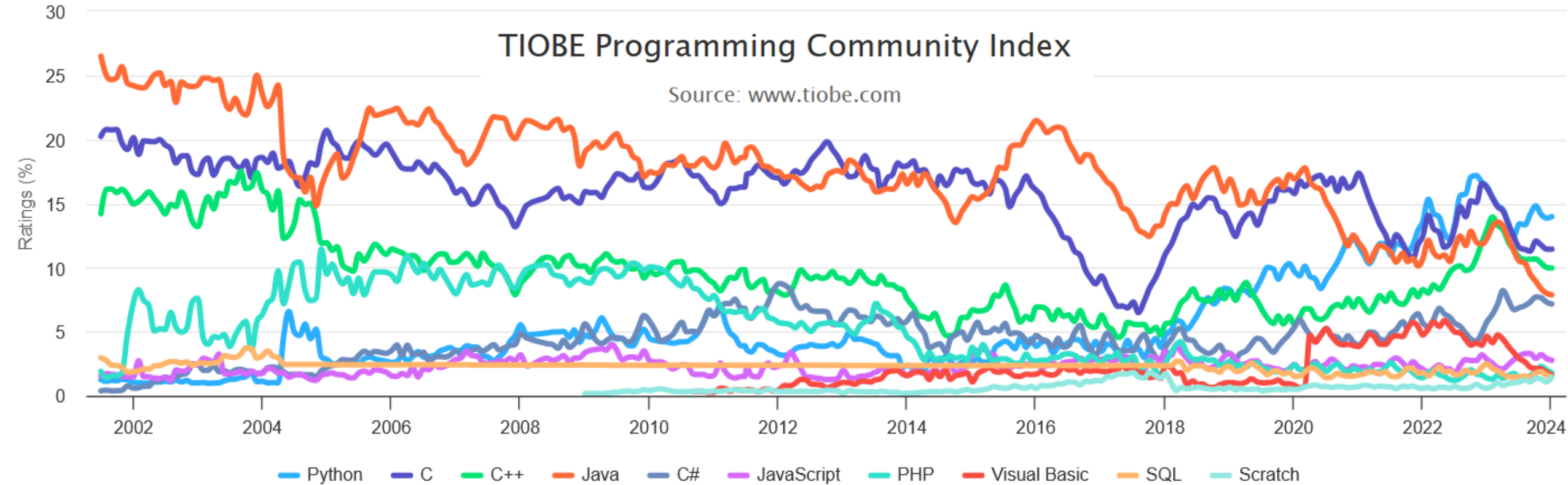
"Hello World"

- In Java, C, C++ a lot of "{", "}" and ";" are needed

- Java tends to have a lot of "public..." details that need to be spelled out

- Python is concise

Java
```java
public class HelloWorld {
    public static void main( String[] args ) {
        System.out.println( "Hello World!" );
        System.exit( 0 );
    }
}
```

C
```c
#include <stdio.h>

int main(int argc, char **argv) {
    printf("Hello World");
    return 0;
}
```

C++
```cpp
#include <iostream>
using namespace std;

int main(int argc, char** argv) {
    cout << "Hello, World!";
    return 0;
}
```

Python 2
```python
print "Hello world"
```

Python 3
```python
print("Hello world")
```

Why Python ?

- **Short concise code**

C index out of bounds

Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system.

en.wikipedia.org/wiki/Debugging

Memory

array A

x

2   A[0]

3   A[1]

~~1~~42
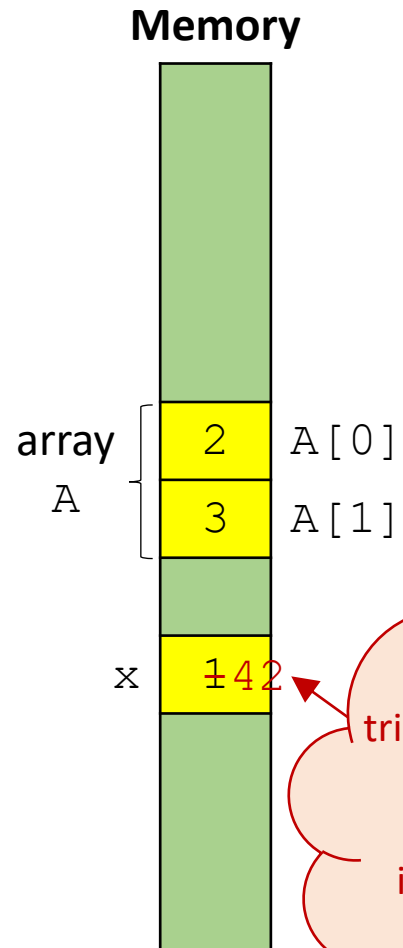
"A" only has size 2, but tries to update the 4th entry. No warning is giving. Something unexpected is overridden in memory. **Have fun debugging!**

indexing.c

```c
#include <stdio.h>
int main() {
    int x = 1;
    int A[2] = {2, 3}; // A[0] = 2, A[1] = 3
    printf("x = %d, A = {%d, %d}\n", x, A[0], A[1]);
    A[3] = 42; // index A[3] out of bounds
    printf("x = %d, A = {%d, %d}\n", x, A[0], A[1]);
    return 0;
}
```

Output

```
$ gcc indexing.c
$ ./a.exe
x = 1, A = {2, 3}
x = 42, A = {2, 3}
```

Skipping checking for invalid indexing makes programs faster, but also requires disciplined programming

… and C++ index out of bounds

Memory

```
array ⌈  2   A[0]
  A  ⌊  3   A[1]
     x   1̶42
```

overrides
something unexpected
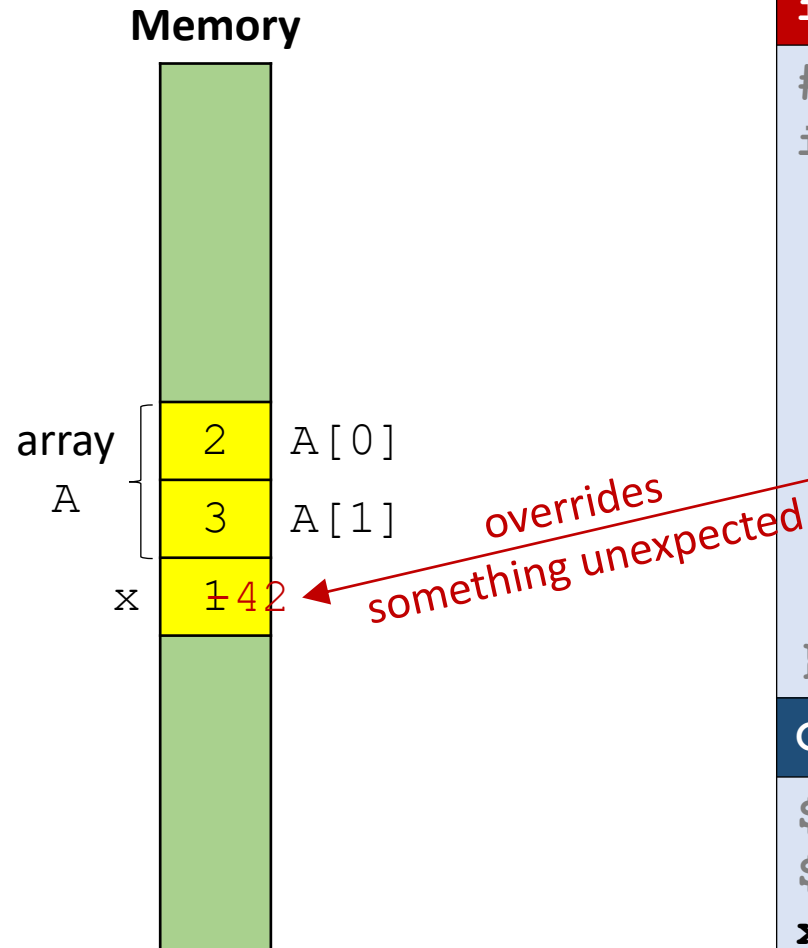
indexing.cpp

```cpp
#include <iostream>
int main() {
    int x = 1;
    int A[2] = {2, 3}; // A[0] = 2, A[1] = 3
    std::cout << "x = " << x << ", A = {"
              << A[0] << ", " << A[1] << "}" << std::endl;
    A[2] = 42; // index A[2] out of bounds
    std::cout << "x = " << x << ", A = {"
              << A[0] << ", " << A[1] << "}" << std::endl;
    return 0;
}
```

Output

```
$ g++ indexing.cpp
$ ./a.exe
x = 1, A = {2, 3}
x = 42, A = {2, 3}
```

... and C++ `vector` index out of bounds

Memory

```
         ┌─────┐
         │     │
         │     │
vector   │  2  │  A[0]
A        │  3  │  A[1]
         │     │
         │     │
vector   │  4  │  B[0]
B        │ 5 42│  B[1]
         │     │
         └─────┘
```

overrides something unexpected
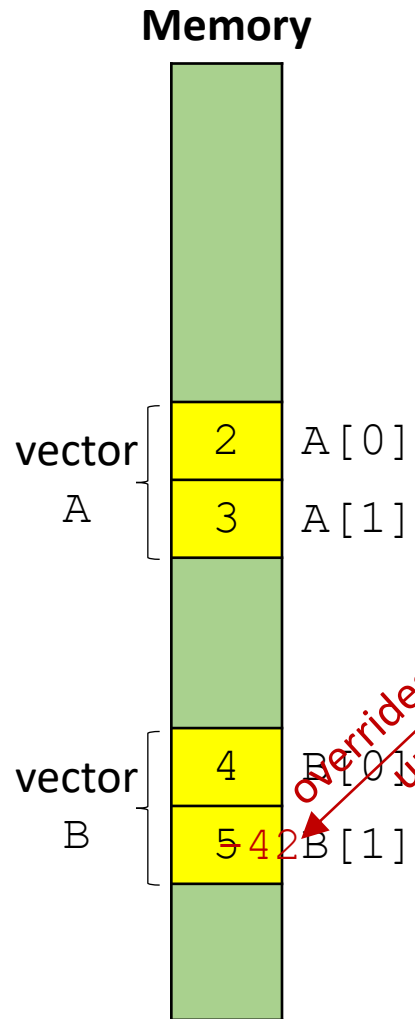
indexing.cpp

```cpp
#include <iostream>
#include <vector>
int main() {
    std::vector<int> A = {2, 3};   // A[0] = 2, A[1] = 3
    std::vector<int> B = {4, 5};   // B[0] = 4, B[1] = 5
    std::cout << "A={" << A[0] << ", " << A[1] << "}, ";
    std::cout << "B={" << B[0] << ", " << B[1] << "}" << std::endl;
    A[9]=42; // index A[9] out of bounds
    std::cout << "A={" << A[0] << ", " << A[1] << "}, ";
    std::cout << "B={" << B[0] << ", " << B[1] << "}" << std::endl;
    return 0;
}
```

Output

```
$ g++ -std=c++11 indexing-vector.cpp
$ ./a.exe
A={2, 3}, B={4, 5}
A={2, 3}, B={4, 42}
```

(Java was developed by James Gosling 1995)

… and Java index out of bounds exception

Memory

```
20   a[0]
21   a[1]
22   a[2]
```
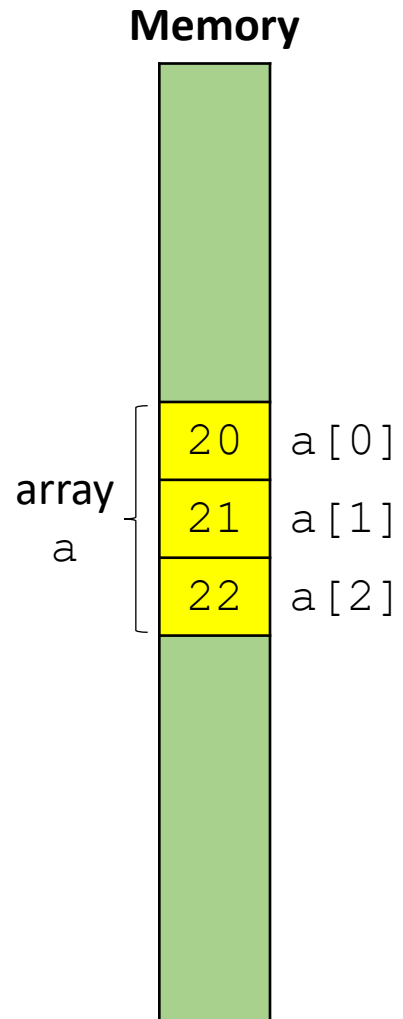
array a

indexing.java

```
class IndexingTest{
    public static void main(String args[]){
        int a[] = {20, 21, 22};
        a[5] = 42; // index a[5] out of bounds
    }
}
```

Output

```
$ javac indexing.java
$ java IndexingTest
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 5
        at IndexingTest.main(indexing.java:5)
```

Java provides error message when running the program

... and Python index out of bounds exception

Memory

array
a

20	a[0]
21	a[1]
22	a[2]
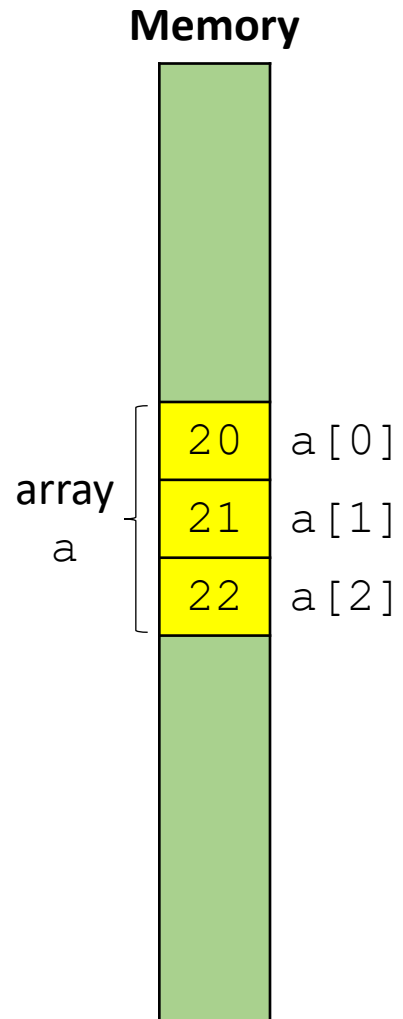
indexing.py

```
a = [20, 21, 22]
a[5] = 42   # index a[5] out of bounds
```

Output

```
$ python indexing.py
Traceback (most recent call last):
  File "indexing.py", line 3, in <module>
    a[5] = 42
IndexError: list assignment index out of range
```

Python provides error message when running the program

Why Python ?

- Short concise code
- **Index out-of-range exceptions**

C++ different ways to print a vector

```cpp
#include <iostream>
#include <vector>
int main() {
  // Vector is part of STL (Standard Template Library)
  std::vector<int> A = {20, 23, 26};
  // "C" indexing - since C++98
  for (int i = 0; i < A.size(); i++)
    std::cout << A[i] << std::endl;
  // iterator - since C++98
  for (std::vector<int>::iterator it = A.begin(); it != A.end(); ++it)
    std::cout << *it << std:: endl;
  // "auto" iterator - since C++11
  for (auto it = A.begin(); it != A.end(); ++it)
    std::cout << *it << std:: endl;
  // Range-based for-loop - since C++11
  for (auto e : A)
    std::cout << e << std:: endl;
}
```

elegant

Java - different ways to print a vector

```java
import java.util.Vector;
import java.util.Iterator;

class IteratorTest{
    public static void main(String[] args) {
        Vector<Integer> a = new Vector<Integer>();
        a.add(7);
        a.add(42);
        // "C" for-loop & get method
        for (int i = 0; i < a.size(); i++)
            System.out.println(a.get(i));
        // iterator
        for (Iterator it = a.iterator(); it.hasNext(); )
            System.out.println(it.next());
        // for-each loop – since Java 5
        for (Integer e : a)
            System.out.println(e);
    }
}
```

elegant

The Python way to print a list

```
print-list.py
a = [20, 23, 26]

for e in a:
    print(e)
```

```
Output
$ python print-list.py
20
23
26
```

Why Python ?

- Short concise code
- Index out of range exceptions
- **Elegant for-each loop**

C++ how not to print a vector

```
$ g++ -std=c++11 print-vector.cpp
cpp-error-message.cpp: In function 'int main()':
cpp-error-message.cpp:7:13: error: no match for 'operator<<' (operand types are 'std::ostream {aka std::basic_ostream<char>}' and 'std::vector<int>')
    std::cout << A << std::endl;
             ^
In file included from /usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/iostream:39:0,
                 from cpp-error-message.cpp:1:
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:628:5: note: candidate: std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&&, const _Tp&) [with _CharT = char; _Traits = std::char_traits<char>; _Tp = std::vector<int>] <near match>
    operator<<(basic_ostream<_CharT, _Traits>&& __os, const _Tp& __x)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:628:5: note:   conversion of argument 1 would be ill-formed:
cpp-error-message.cpp:7:16: error: cannot bind 'std::ostream {aka std::basic_ostream<char>}' lvalue to 'std::basic_ostream<char>&&'
    std::cout << A << std::endl;
                ^
In file included from /usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/iostream:39:0,
                 from cpp-error-message.cpp:1:
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:108:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& (std::basic_ostream<_CharT, _Traits>::__ostream_type&)) [with _CharT = char; _Tra
std::basic_ostream<_CharT, _Traits>::__ostream_type = std::basic_ostream<char>]
    operator<<(__ostream_type& (*__pf)(__ostream_type&))
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:108:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'std::basic_ostream<char>::__ostream_type& (*)(std::basic_ostream<char>::__ostream_type&) {aka std::basic_ostream<char>& (*)(std::basic_ostream<char>&)}'
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:117:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(std::basic_ostream<_CharT, _Traits>::__ios_type& (*)(std::basic_ostream<_CharT, _Traits>::__ios_type&)) [with _CharT = char; _Traits =
_Traits>::__ostream_type = std::basic_ostream<char>; std::basic_ostream<_CharT, _Traits>::__ios_type = std::basic_ios<char>]
    operator<<(__ios_type& (*__pf)(__ios_type&))
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:117:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'std::basic_ostream<char>::__ios_type& (*)(std::basic_ostream<char>::__ios_type&) {aka std::basic_ios<char>& (*)(std::basic_ios<char>&)}'
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:127:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(std::ios_base& (*)(std::ios_base&)) [with _CharT = char; _Traits = std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream
    operator<<(ios_base& (*__pf) (ios_base&))
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:127:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'std::ios_base& (*)(std::ios_base&)'
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:166:7: note: candidate: std::basic_ostream                     s = std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream_type = std::basic_ostream
    operator<<(long __n)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:166:7: note:   no known conversion for arg
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:170:7: note: candidate: std::basic_ostream                     ar; _Traits = std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream_type = std::basic_ostream<cha
    operator<<(unsigned long __n)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:170:7: note:   no known conversion for arg
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:174:7: note: candidate: std::basic_ostream                     std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream_type = std::basic_ostream<cha
    operator<<(bool __n)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:174:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'bool'
In file included from /usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:638:0,
                 from /usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/iostream:39,
                 from cpp-error-message.cpp:1:
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/bits/ostream.tcc:91:5: note: candidate: std::basic                     d::char_traits<char>]
    basic_ostream<_CharT, _Traits>::
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/bits/ostream.tcc:91:5: note:   no known conversion
In file included from /usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/iostream:39:0,
                 from cpp-error-message.cpp:1:
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:181:7: note: candidate: std::basic_ostream                     har; _Traits = std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream_type = std::bas
    operator<<(unsigned short __n)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:181:7: note:   no known conversion for arg
In file included from /usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:638:0,
                 from /usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/iostream:39,
                 from cpp-error-message.cpp:1:
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/bits/ostream.tcc:105:5: note: candidate: std::basi                     ar_traits<char>]
    basic_ostream<_CharT, _Traits>::
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/bits/ostream.tcc:105:5: note:   no known conversio
In file included from /usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/iostream:39:0,
                 from cpp-error-message.cpp:1:
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:192:7: note: candidate: std::basic_ostream                     ream<_CharT, _Traits>::__ostream_type = std::basic_ost
    operator<<(unsigned int __n)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:192:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'unsigned :
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:201:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& std::basic_                    tream<_CharT, _Traits>::__ostream_type = std::basic_os
    operator<<(long long __n)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:201:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'long long
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:205:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& std::basic_                    :basic_ostream<_CharT, _Traits>::__ostream_type = std::
    operator<<(unsigned long long __n)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:205:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'long long unsigned int'
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:220:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(double) [with _CharT = char; _Traits = std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream_type = std::basic_ostream<
    operator<<(double __f)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:220:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'double'
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:224:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(float) [with _CharT = char; _Traits = std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream_type = std::basic_ostream<
    operator<<(float __f)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:224:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'float'
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:232:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(long double) [with _CharT = char; _Traits = std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream_type = std::basic_ost
    operator<<(long double __f)
    ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:232:7: note:   no known conversion for argument 1 from 'std::vector<int>' to 'long double'
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:245:7: note: candidate: std::basic_ostream<_CharT, _Traits>::__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(const void*) [with _CharT = char; _Traits = std::char_traits<char>; std::basic_ostream<_CharT, _Traits>::__ostream_type = std::basic_ost
```

print-vector.cpp

```cpp
#include <iostream>
#include <vector>


int main() {
    std::vector<int> A = {2, 3};
    std::cout << A << std::endl;
    return 0;

}
```
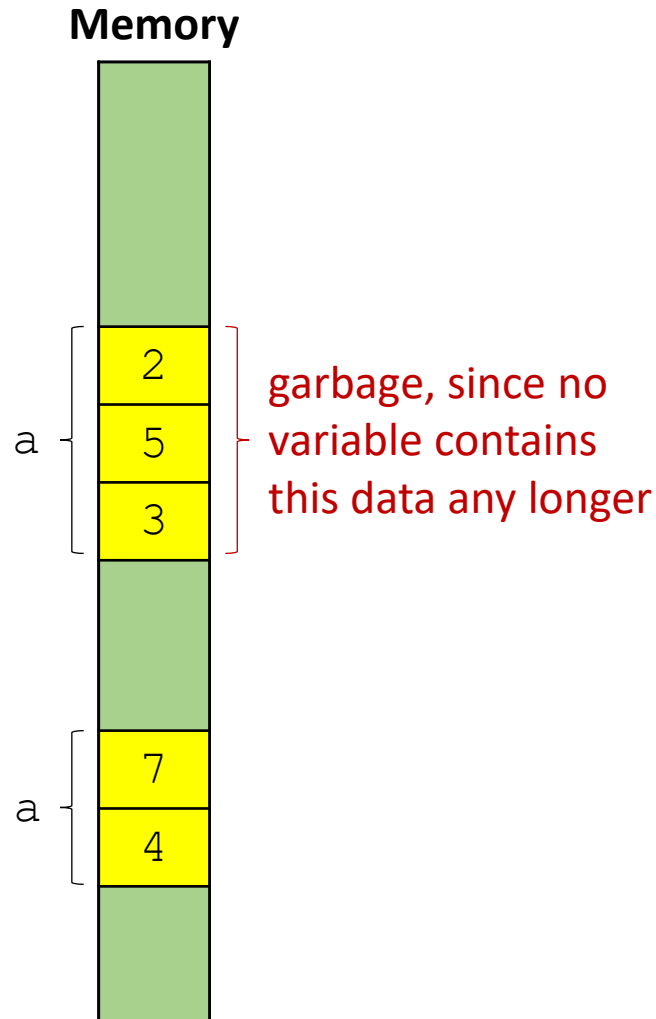
C++ vectors cannot be printed directly –
mistake results in +200 lines of error messages

Why Python ?

- Short concise code
- Index out of range exceptions
- Elegant for-each loop
- **Python hopefully better error messages than C++**

Python and garbage collection

Memory

```
a = [2, 5, 3]
a = [7,4]
```

garbage.py

a gets new value

2
5
3

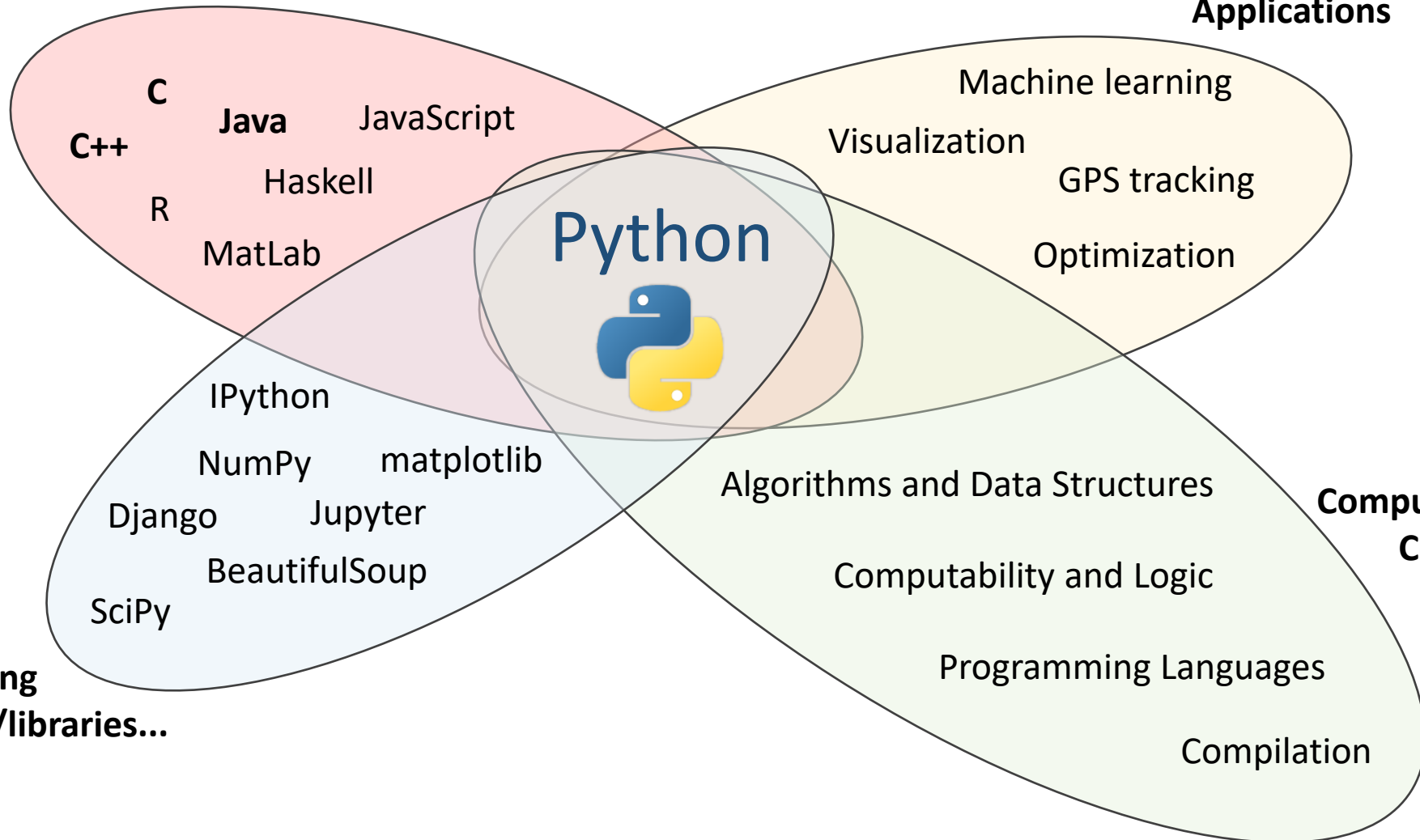a — garbage, since no variable contains this data any longer

7
4

a

- Python and e.g. Java, C# and JavaScript have a **garbage collector** to automatically recycle garbage

- C and C++ garbage collection must be done explicitly by the program; forgetting to **free** memory again results in **memory leaks** – which can be really hard to find. **Have fun debugging!**

Why Python ?

- Short concise code
- Index out of range exceptions
- Elegant for-each loop
- Python hopefully better error messages than C++
- **Garbage collection is done automatically**

Python performance vs C, C++ and Java

Compute sum $1 + 2 + 3 + \cdots + n = n^2/2 + n/2$

$$1 + 2 + \cdots + n$$
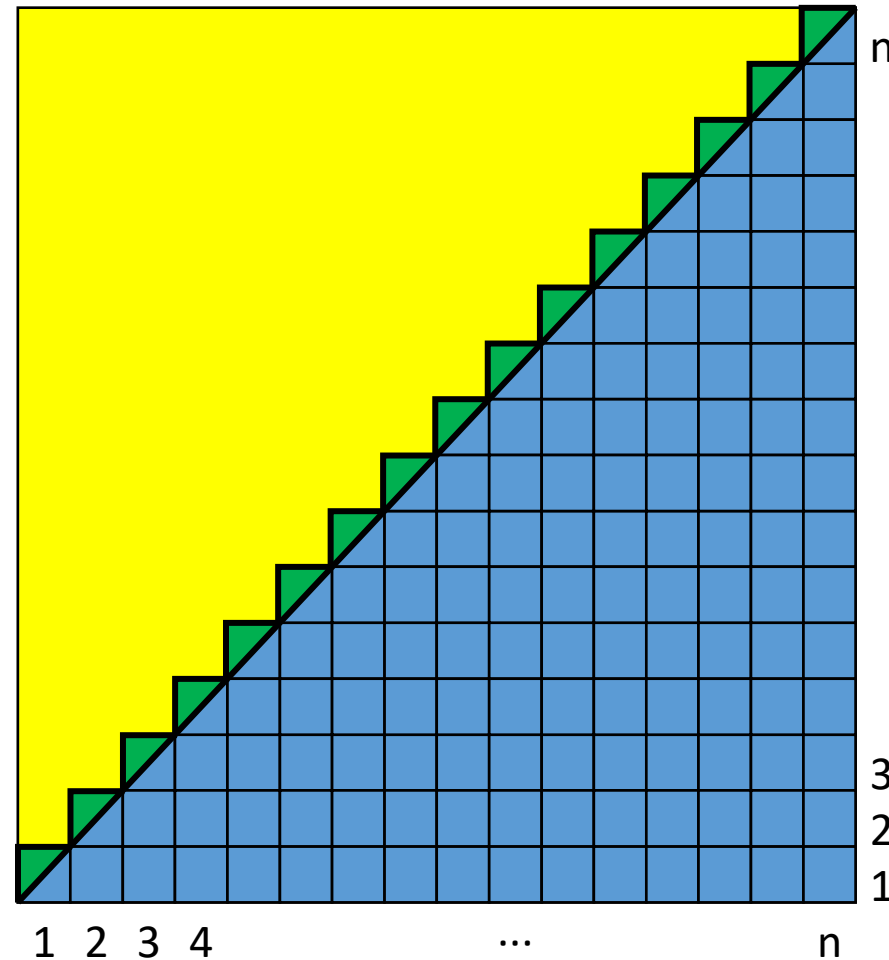
add.py
```python
import sys

n = int(sys.argv[1])
sum = 0
for i in range(1, n + 1):
    sum += i
print("Sum =", sum)
```

add.c
```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
  int n = atoi(argv[1]);
  int sum = 0;
  for (int i = 1; i <= n; i++)
    sum += i;
  printf("Sum = %d\n", sum);
}
```

add.cpp
```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int main(int argc, char *argv[]) {
  int n = atoi(argv[1]);
  int sum = 0;
  for (int i = 1; i <= n; i++)
    sum += i;
  cout << "Sum = " << sum << endl;
}
```

add.java
```java
class Add{
    public static void main(String args[]){
        int n = Integer.parseInt(args[0]);
        int sum = 0;
        for (int i = 1; i <= n; i++)
            sum += i;
        System.out.println("Sum = " + sum);
    }
}
```

Timing results

Python

n	C (gcc 9.2)	C++, int (g++ 9.2)	C++, long (g++ 9.2)	Java (12.0)	CPython (3.8.1)	PyPy (7.3.0)	Numba, int64
10^7	0.001 sec*	0.001 sec*	0.003 sec	0.006 sec*	1.5 sec	0.27 sec	0.002 sec
10^9	0.10 sec**	0.10 sec**	0.30 sec	0.40 sec**	145 sec	27 sec	0.2 sec

Wrong output (overflow)

* -2004260032 instead of 50000005000000

** -243309312 instead of 500000000500000000

- since C, C++, and Java only uses 32 bits to represent integers (and 64 bits for "long" integers)

Have fun debugging!

```
Bit            6666666666555555555544444444443333333333222222222211111111110000000000
position       9876543210987654321098765432109876543210987654321098765432109876543210

bin(10**9)                                                 111011100110101100101000000000

bin(50000005000000)                          1011010111100110001000100010010110101101000000
bin(-2004260032+2**32)                                     10001000100010010110101101000000

bin(500000000500000000)       1101111000001011011010110011111000101111111011001010000000
bin(-243309312+2**32)                                      11110001011111110110010100000000
```

Timing results

n	C (gcc 9.2)	C++, int (g++ 9.2)	C++, long (g++ 9.2)	Java (12.0)	Python (3.8.1)	PyPy (7.3.0)	Numba, int64
10^7	0.001 sec*	0.001 sec*	0.003 sec	0.006 sec*	1.5 sec	0.27 sec	0.002 sec
10^9	0.10 sec**	0.10 sec**	0.30 sec	0.40 sec**	145 sec	27 sec	0.2 sec

Python

- Relative speed

C ≈ C++ > Java >> Python

- C, C++, Java need to care about integer overflows – select integer representation carefully with sufficient number of bits (8, 16, 32, 64, 128)

- Python natively works with arbitrary long integers (as memory on your machine allows). Also possible in Java using the class java.math.BigInteger

- Python programs can (sometimes) run faster using PyPy

- Number crunching in **Python** should be delegated to **specialized modules (e.g. Numpy, CPLEX, Numba)** – often written in C or C++ and requires selecting right integer representation

Interpreter vs Compiler

C / C++ program
(.c, .cpp)

Java program
(.java)

Python program
(.py)

Internally generates Assembly code
```
.L4:
        movl    -8(%rbp), %eax
        addl    %eax, -4(%rbp)
        addl    $1, -8(%rbp)
.L3:
        movl    -8(%rbp), %eax
        cmpl    -12(%rbp), %eax
        jle     .L4
        movl    -4(%rbp), %edx
        movl    -12(%rbp), %eax
        movl    %edx, %r8d
        movl    %eax, %edx
        leaq    .LC1(%rip), %rcx
        call    printf
        movl    $0, %eax
        addq    $48, %rsp
        popq    %rbp
```

Compiler
(gcc, g++)

Java compiler
(javac)

Java bytecode
(.class)

Internally CPython generates bytecode
```
 0 LOAD_CONST      1 (0)
 2 STORE_FAST      1 (sum)
 4 SETUP_LOOP     30 (to 36)
 6 LOAD_GLOBAL     0 (range)
 8 LOAD_CONST      2 (1)
10 LOAD_FAST       0 (n)
12 LOAD_CONST      2 (1)
14 BINARY_ADD
16 CALL_FUNCTIO    2
18 GET_ITER
20 FOR_ITER       12 (to 34)
22 STORE_FAST      2 (i)
24 LOAD_FAST       1 (sum)
26 LOAD_FAST       2 (i)
28 INPLACE_ADD
30 STORE_FAST      1 (sum)
32 JUMP_ABSOLUT   20
34 POP_BLOCK
36 LOAD_FAST       1 (sum)
38 RETURN_VALUE
```

Executable code
(.exe)

Java Virtual
Machine
(java)

CPython
interpreter
(python)

execution

execution

execution

Why Python ?

- Short concise code
- Index out of range exceptions
- Elegant for-each loop
- Python hopefully better error messages than C++
- Garbage collection is done automatically
- **Exact integer arithmetic (no overflows)**
- **Can delegate number crunching to C, C++, …**
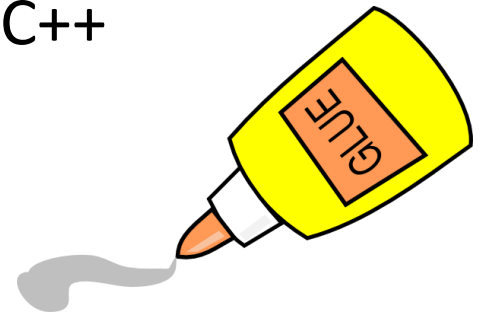
This course

Programming Languages

(Scientific) Applications

C

C++

Java   JavaScript

R   Haskell

MatLab

Python

Machine learning

Visualization

GPS tracking

Optimization

IPython

NumPy   matplotlib

Django   Jupyter

BeautifulSoup

SciPy

Algorithms and Data Structures

Computability and Logic

Programming Languages

Compilation

**Programming
modules/packages/libraries...**

**Computer Science
Courses**

Course overview

1. Introduction to Python	10. Functions as objects	19. Linear programming
2. Python basics / if	11. Object oriented programming	20. Generators, iterators, with
3. Basic operations	12. Class hierarchies	21. Modules and packages
4. Lists / while / for	13. Exceptions and files	22. Working with text
5. Tuples / comprehensions	14. Doc, testing, debugging	23. Relational data
6. Dictionaries and sets	15. Decorators	24. Clustering
7. Functions	16. Dynamic programming	25. Graphical user interfaces (GUI)
8. Recursion	17. Visualization and optimization	26. Java vs Python
9. Recursion and Iteration	18. Multi-dimensional data	27. Final lecture

10 handins
1 final project (last 1 month)

History of Python development

- Python created by Guido van Rossum in 1989, first release 0.9.0 1991

- Python 2 → Python 3 (clean up of Python 2 language)
  - Python 2 – version 2.0 released 2000, final version 2.7 released mid-2010
  - Python 3 – released 2008, current release 3.12.1

- Python 3 is *not* backward compatible, libraries incompatible

Python 2	Python 3
print 42	print(42)
int = C long (32 bits)	int = arbitrary number of digits (= named "long" in Python 2)
7/3 → 2 returns "int"	7/3 → 2.333… returns "float"
range() returns list (memory intensive)	range() returns iterator (memory efficient; xrange in Python 2)

Installing Python

Running the Python Interpreter from a terminal

- Open Command Prompt (Windows-key + cmd)

- Type "python" + return

- Start executing Python statements

- To exit shell:
Ctrl-Z + return *or*
exit() + return

Installing IPython –
A more powerful interactive Python shell

- Open Command Prompt
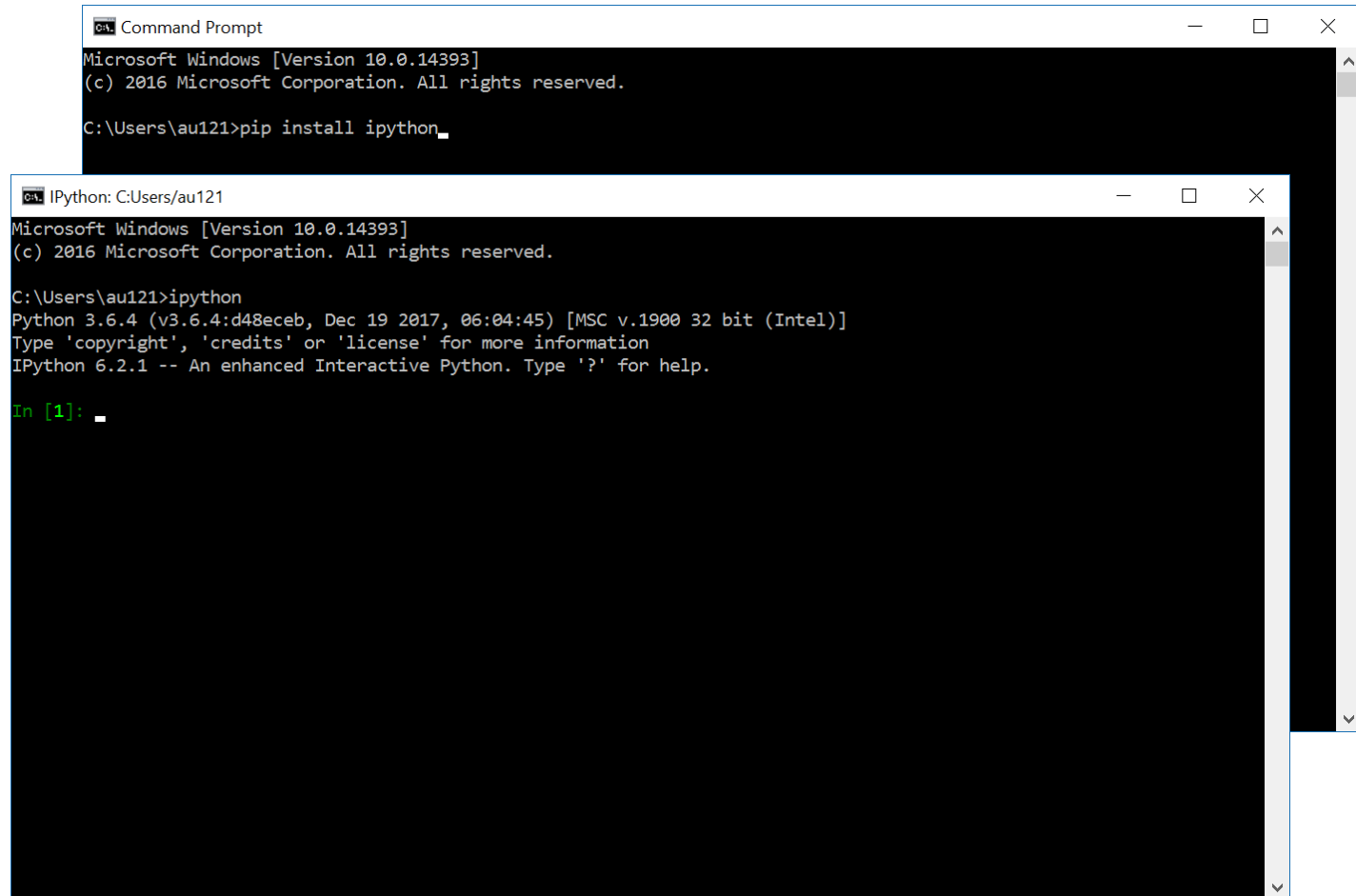
- Execute:

    ```
    pip install ipython
    ```

- Start ipython

    ```
    ipython
    ```

pip = the Python package manager

Some other usefull packages

- Try installing some more Python packages:

```
pip install numpy
pip install scipy
pip install matplotlib
pip install pylint
```

linear algebra support (N-dimensional arrays)

numerical integration and optimization

2D plotting library

Python source code analyzer enforcing a coding standard

Creating a Python program the very basic way
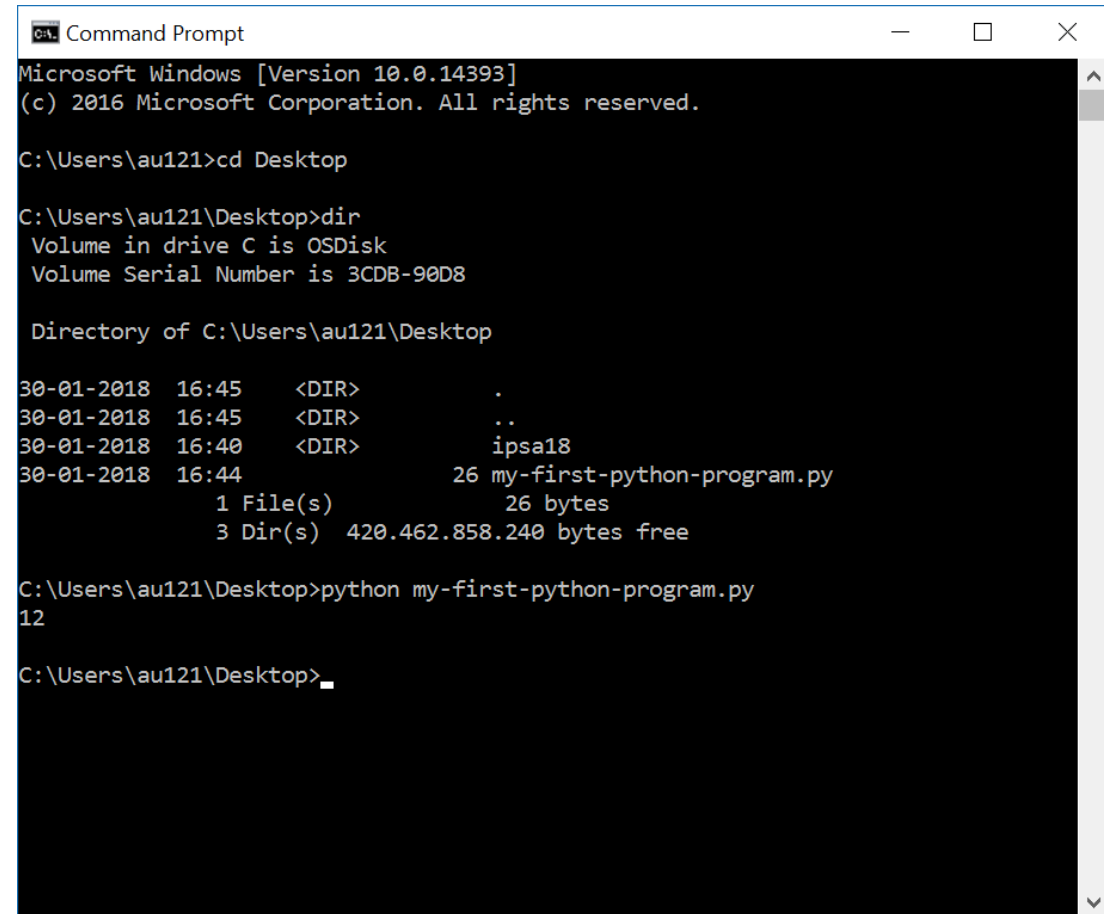
```
my-first-python-program.py - Notepad
File  Edit  Format  View  Help

x = 3
y = 4
print(x * y)

                                    Ln 1, Col 1
```

```
Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\au121>cd Desktop

C:\Users\au121\Desktop>dir
 Volume in drive C is OSDisk
 Volume Serial Number is 3CDB-90D8

 Directory of C:\Users\au121\Desktop

30-01-2018  16:45    <DIR>          .
30-01-2018  16:45    <DIR>          ..
30-01-2018  16:40    <DIR>          ipsa18
30-01-2018  16:44                26 my-first-python-program.py
               1 File(s)             26 bytes
               3 Dir(s)   420.462.858.240 bytes free

C:\Users\au121\Desktop>python my-first-python-program.py
12

C:\Users\au121\Desktop>
```
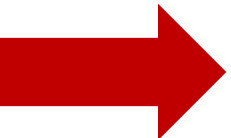
- Open Notepad (or TextEdit on Mac)
  - write a simple Python program
  - save it
- Open a command prompt
  - go to folder (using cd)
  - run the program using
    ```
    python <program name>.py
    ```

… or open IDLE and run program with F5

enable line numbers under options

- IDLE ships with Python from python.org
- Good beginner IDE (Integrated Development Environment)
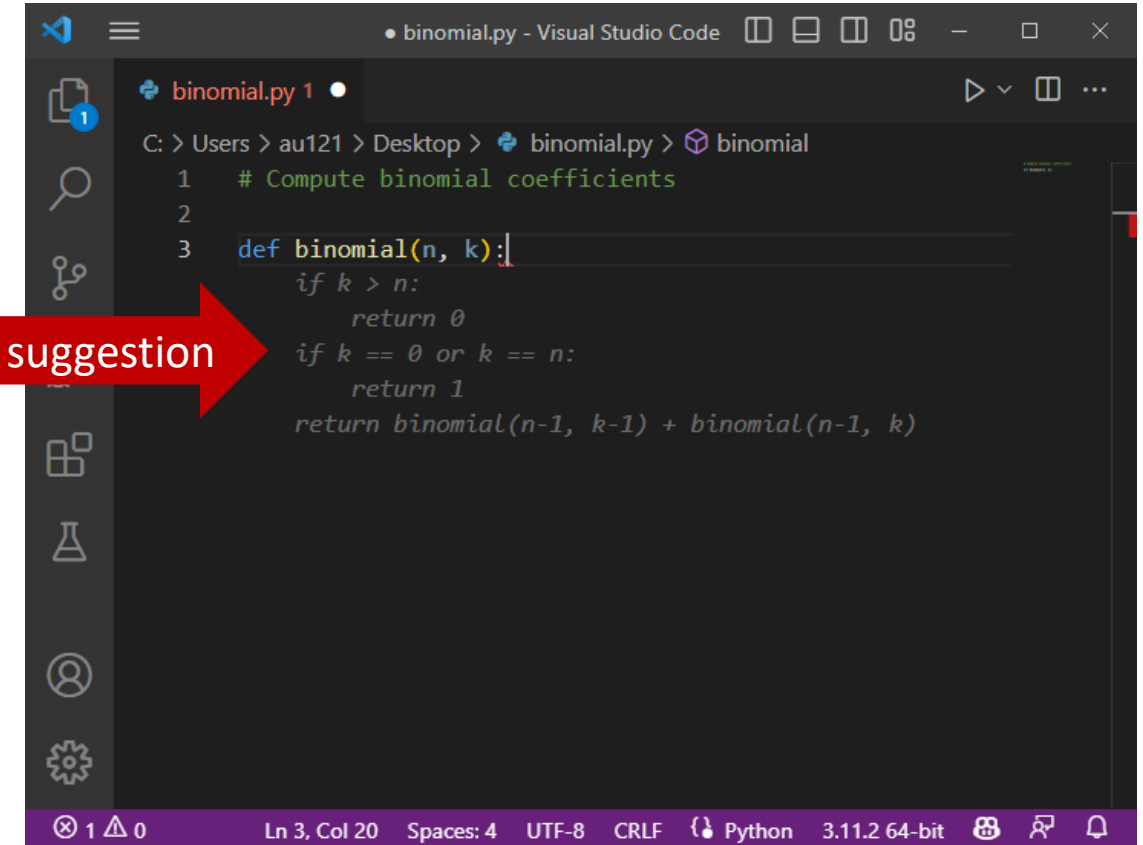
The Python Ecosystem

- **Interpreters/compiler**
  - CPython   – reference C implementation from python.org
  - PyPy   – written in RPython (a subset of Python) – faster than Cpython
  - Jython   – written in Java and compiles to Java bytecode, runs on the JVM
  - IronPython   – written in C#, compiles to Microsoft's Common Language Runtime (CLR) bytecode
  - Cython   – project translating Python-ish code to C

- **Shells (IPython, IDLE, Jupyter)**

- **Libraries/modules/packages**
  - pypi.python.org/pypi (PyPI - the Python Package Index, +500.000 packages)

- **IDEs (Integrated development environment)**
  - IDLE comes with Python (docs.python.org/3/library/idle.html)
  - Anaconda w. Spyder, IPython (www.anaconda.com/download)
  - Canopy (enthought.com/product/canopy)
  - Visual Studio Code (code.visualstudio.com)
  - Python tools for Visual Studio (github.com/Microsoft/PTVS)
  - PyCharm (www.jetbrains.com/pycharm/)
  - Emacs (Python mode and ElPy mode)
  - Notepad++

- **Python Style guide (PEP8)**
  - pylint, pep8, flake8

- **Python online**
  - Google colab (colab.research.google.com), repl.it, sagemath.org, …

Try to google "best ide python"

IDEs and AI assistants

- Some IDEs integrate AI assistants to support code suggestions, e.g. GitHub Copilot in VS Code

- AI assistants increase productivity *if* you understand their output

- AI assistants are not allowed at the exam

Guido van Rossum, inventor of Python, on GitHub Copilot
"I use it every day. It writes a lot of code for me… and usually it is slightly wrong but it still safes me typing."
Python and the Future of Programming, Guido van Rossum interviewed by Lex Fridman