

# Introduction to Programming with Scientific Applications

Gerth Stølting Brodal  
Department of Computer Science  
Aarhus University



**Course evaluation**  
*“The first lecture was intimidating  
and overwhelming”*



# 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

# Course description – [kursuskatalog.au.dk/en/course/130939/](https://kursuskatalog.au.dk/en/course/130939/)

## 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, no AI

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%

# Question – Primary Education?

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

# Question – Programming languages you know?

+750 listed on [en.wikipedia.org/wiki/List\\_of\\_programming\\_languages](https://en.wikipedia.org/wiki/List_of_programming_languages)

# Question – Programming experience?

For the programming language you know best (if any) please state your 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



	Monday	Tuesday	Wednesday	Thursday	Friday
8:15-9:00	Lecture				
9:15-10:00		Hold 2			
10:15-11:00	Study cafe		Study cafe		
11:15-12:00			Lecture	MA1	
12:15-13:00					
13:15-14:00					
14:15-15:00			DV		
15:15-16:00					
16:15-17:00		FY Hold 1	MØ1		
17:15-18:00			MØ2		

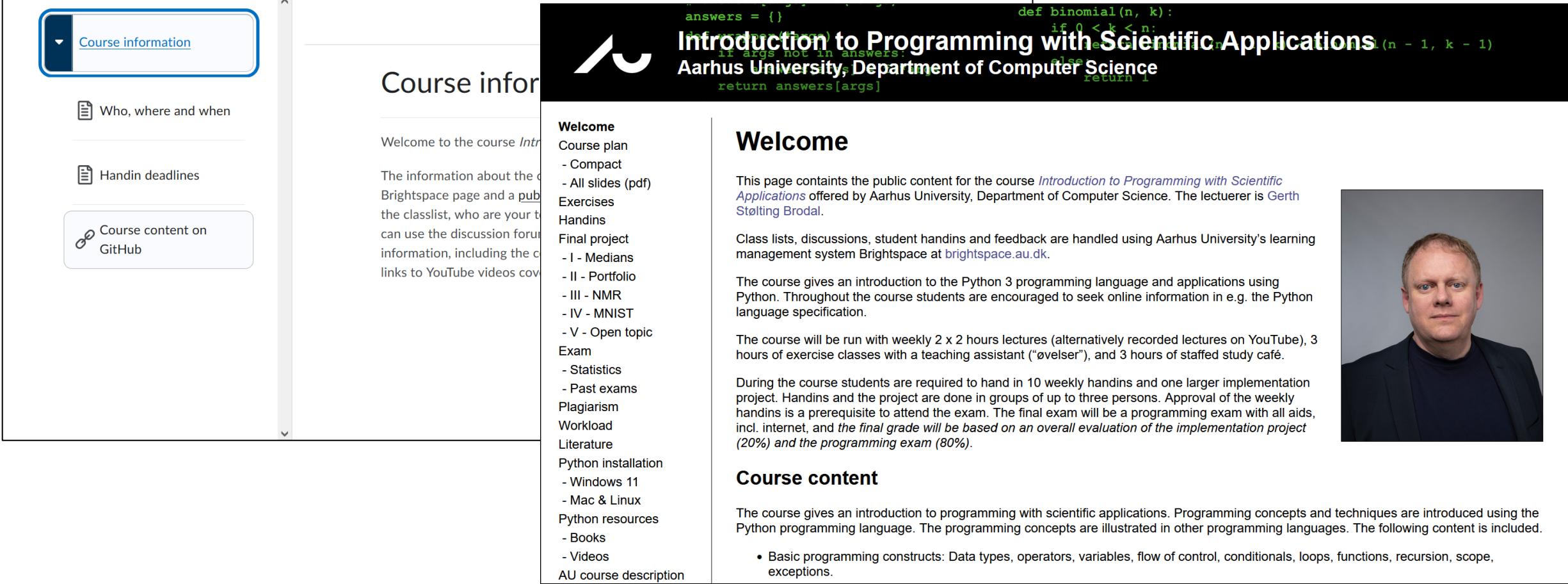


Week	Monday	Tuesday	Wednesday	Thursday	Friday
5	F1	no TA class	F2		
6	F3	TØ1	TØ1 / F4	TØ1	TØ1
7	F5	TØ2	TØ2 / F6	TØ2	TØ2
8	F7	TØ3	TØ3 / F8	TØ3	TØ3
9	F9	TØ4	TØ4 / F10	TØ4	TØ4
10	F11	TØ5	TØ5 / F12	TØ5	TØ5
11	F13	TØ6	TØ6 / F14	TØ6	TØ6
12	F15	TØ7	TØ7 / F16	TØ7	TØ7
13	F17	TØ8	TØ8 / F18	TØ8	TØ8
14				Easter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

# Course page on Brightspace and GitHub



gsbrodal.github.io/ipsa

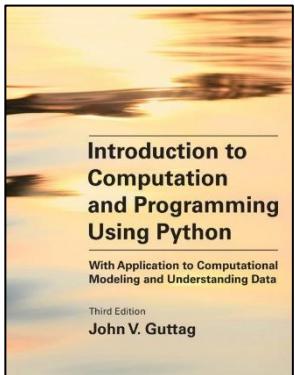


The screenshot shows a comparison between a Brightspace course page and a corresponding GitHub page for the same course. The GitHub page is a static website for the course "Introduction to Programming with Scientific Applications" offered by Aarhus University, Department of Computer Science. It includes:

- A header with the course title and a Python code snippet.
- A "Welcome" section with a brief introduction and the lecturer's name, Gerth Stølting Brodal.
- A "Course content" section with a detailed outline of topics covered in the course.
- A portrait photo of the lecturer, Gerth Stølting Brodal.

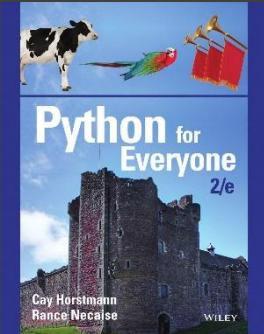
The Brightspace course page on the left shows the same content structure but lacks the GitHub-specific features like the code snippet and the portrait photo.

# 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.

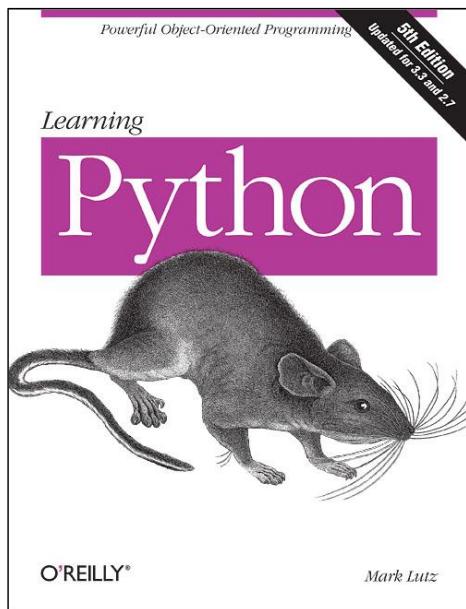
- *[Guttag, 2<sup>nd</sup> Ed., page 8] "The reader should be forewarned that this book is by no means a comprehensive introduction to Python". 3<sup>rd</sup> Ed. added about 80 pages on introduction to Python.*
- *Covers all basic features of Python enabling you to deal with data in Chapters 1-10 (212 pages) - remaining chapters are applications*
- *Other resources: Google, stackoverflow, Python.org, YouTube, Als...*



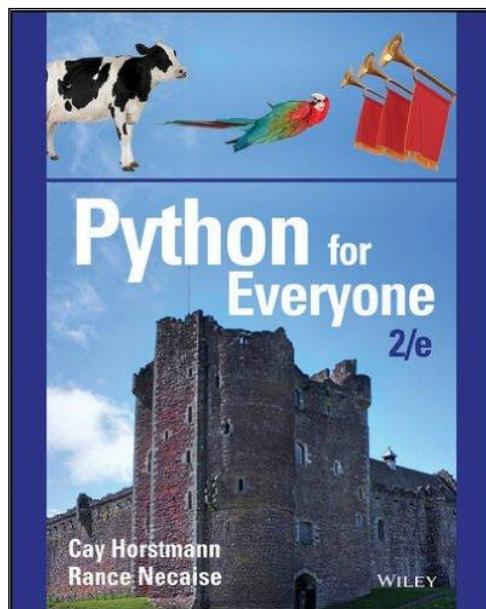
Comparison to a standard text book on the *programming language* Python by Cay Horstmann and Rance Necaise:

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

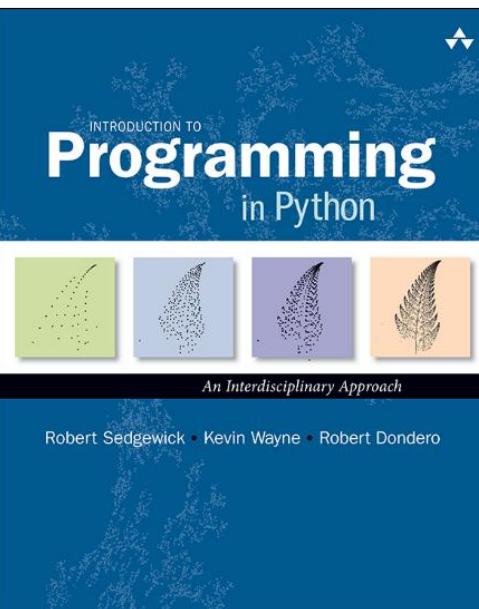
# Some other books on Python



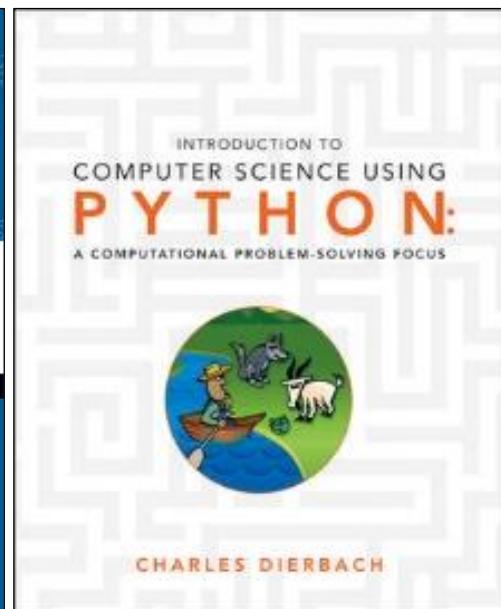
O'Reilly, 2013  
1684 pages



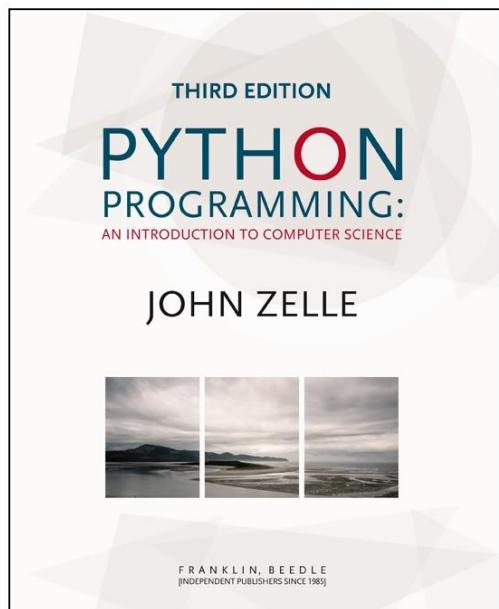
Wiley, 2016  
752 pages



Addison-Wesley, 2015  
794 pages



Wiley, 2013  
580 pages



FRANKLIN, BEEDLE  
INDEPENDENT PUBLISHERS SINCE 1985

Franklin & Beedle, 2016  
552 pages

... numerous online introduction texts/courses/videos on Python

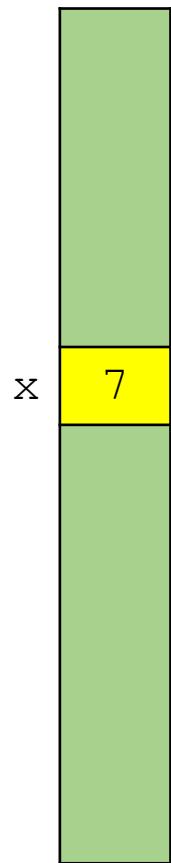
# Two Python programs

# A Python program

Python shell

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

Memory



- 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

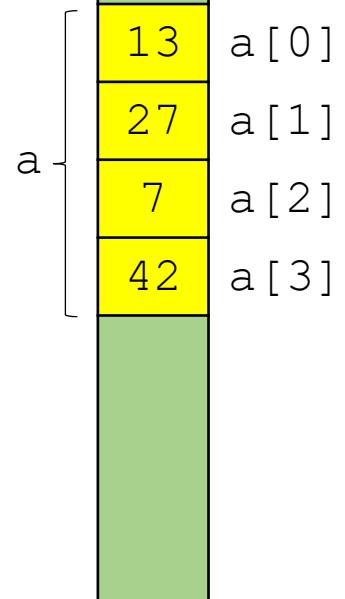
- a) 10
- b) 15
- c) 25
- d) [15, 10]
- e) Error
- f) Don't know

# Another Python program using lists

## Python shell

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

## Memory



- `[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 3<sup>rd</sup> 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

# Why Python ?



the next slides will be technical

# TIOBE Index January 2025



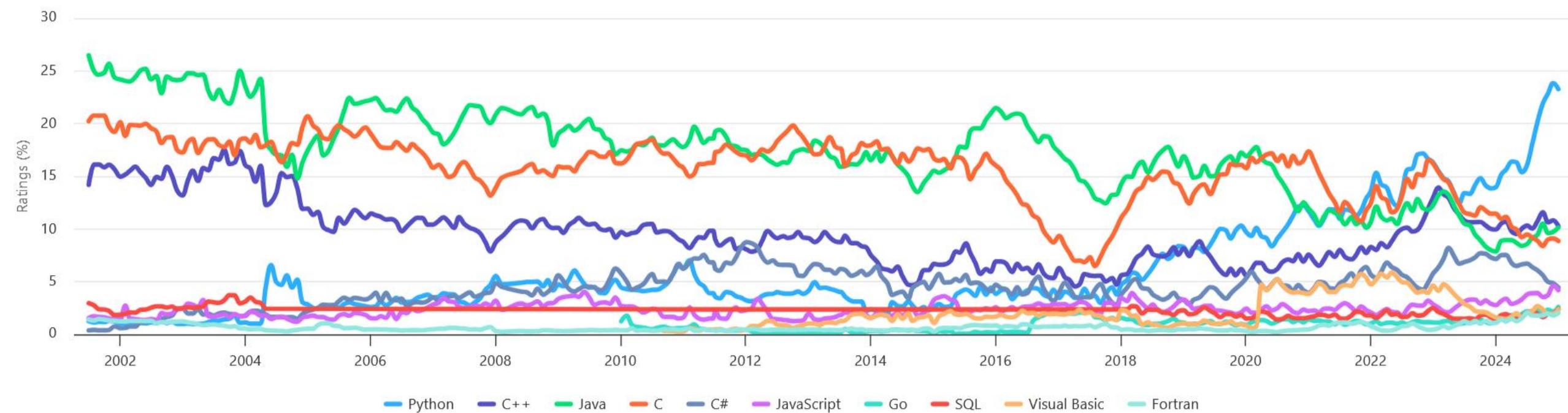
Jan 2025	Jan 2024	Change	Programming Language	Ratings	
1	1		Python	23.28%	+9.32%
2	3		C++	10.29%	+0.33%
3	4		Java	10.15%	+2.28%
4	2		C	8.86%	-2.59%
5	5		C#	4.45%	-2.71%
6	6		JavaScript	4.20%	+1.43%
7	11		Go	2.61%	+1.24%
8	9		SQL	2.41%	+0.95%
9	8		Visual Basic	2.37%	+0.77%
10	12		Fortran	2.04%	+0.94%

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.

# Popularity of programming languages

TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# “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

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

## C

```
#include <stdio.h>  
  
int main(int argc, char **argv) {  
    printf("Hello World");  
    return 0;  
}
```

## C++

```
#include <iostream>  
using namespace std;  
  
int main(int argc, char** argv) {  
    cout << "Hello, World!";  
    return 0;  
}
```

## Python 2

```
print "Hello world"
```

## Python 3

```
print("Hello world")
```

# Why Python ?

- Short concise code

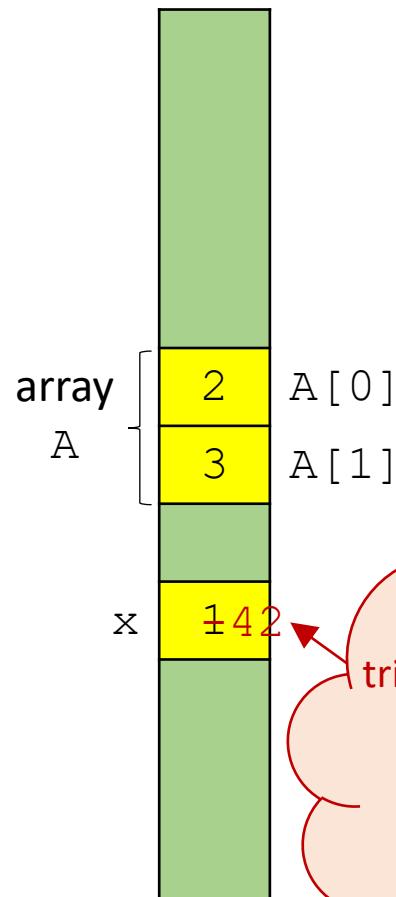
(C developed by Dennis Ritchie 1969-73)

# 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](https://en.wikipedia.org/wiki/Debugging)

Memory



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

## indexing.c

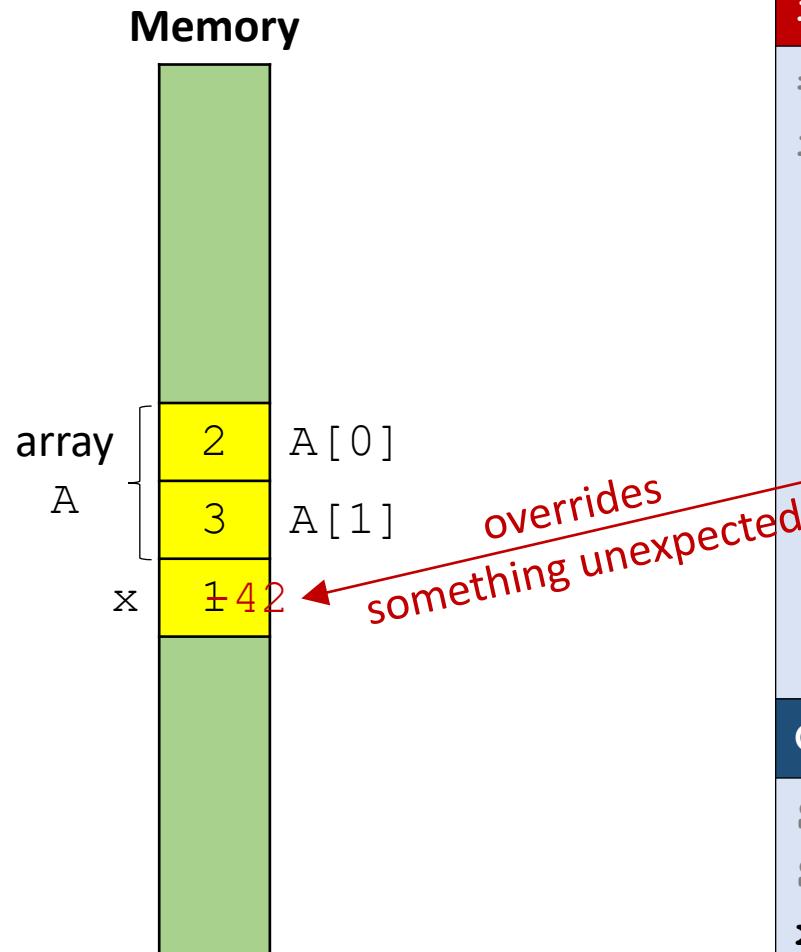
```
#include <stdio.h>
int main() {
    int x = 1;
    int A[2] = {2, 3}; // A[0] = 2, A[1] = 3
    printf("x = %d, A = {%d, %d}\n", x, A[0], A[1]);
    A[3] = 42; // index A[3] out of bounds
    printf("x = %d, A = {%d, %d}\n", x, A[0], A[1]);
    return 0;
}
```

## 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



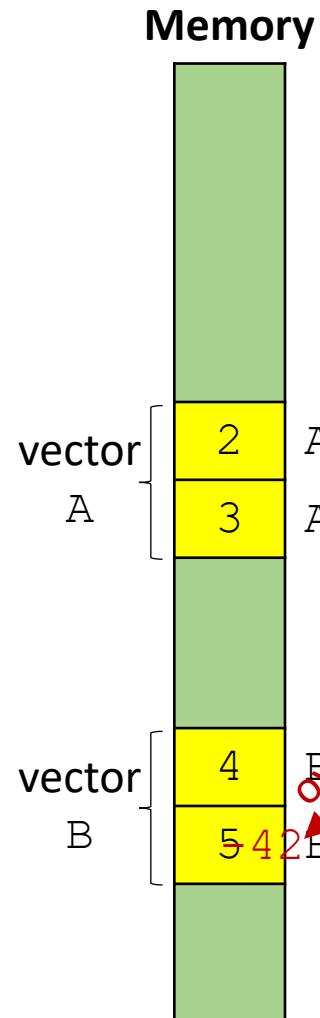
## indexing.cpp

```
#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



## indexing.cpp

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

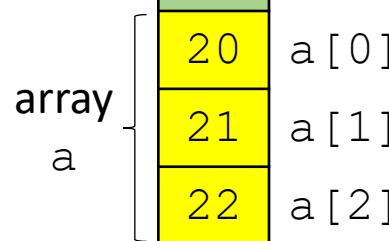
int main() {
    std::vector<int> A = {2, 3}; // A[0] = 2, A[1] = 3
    std::vector<int> B = {4, 5}; // B[0] = 4, B[1] = 5
    std::cout << "A={" << A[0] << ", " << A[1] << "}, ";
    std::cout << "B={" << B[0] << ", " << B[1] << "}" << std::endl;
    A[9]=42; // index A[9] out of bounds
    std::cout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

# ... and Java index out of bounds exception

Memory



## indexing.java

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

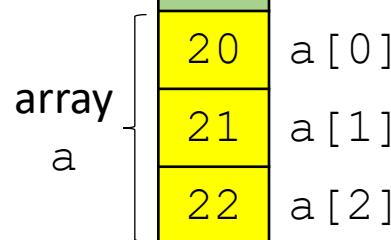
## Output

```
$ 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



## 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

# Memory safety

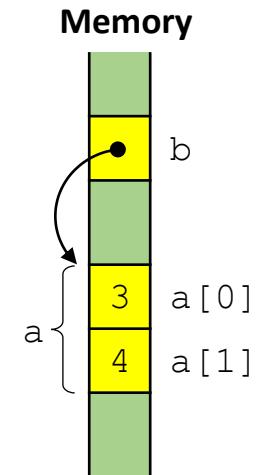
The White House 2024 | Press Release: “Future Software Should Be Memory Safe” ([www.whitehouse.gov](http://www.whitehouse.gov))

National Security Agency 2022 | Cybersecurity Information Sheet: *Software Memory Safety* ([media.defense.gov](http://media.defense.gov))

- C and C++ are flexible but **memory unsafe** programming languages
  - Unintended writes or reads to memory can be exploited by malicious cyber actors 
- Python, Java, Rust are examples of memory safe languages
- Rust aims at achieving the efficiency of C by slightly restricting flexibility

`indexing.rs`

```
fn main() {
    let mut a = [3, 4];
    a[2] = 7; // Compile error: this operation will panic at runtime
    for i in 2..3 { a[i] = 7 } // Run-time panic: index out of bounds
    let b = &mut a;
    a[1] = 6; // Compile error: cannot use `a` because it was mutably borrowed
    (*b)[0] = 5;
    for i in 0..2 { println!("a[{}] = {}", i, a[i]) }
}
```



# Why Python ?

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

# C++ different ways to print a vector

**vector-iterator.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

**vector-iterator.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**

```
$ 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++/ios: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>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, std::ostream_type& __os) [with _CharT = char; _Traits = 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_types)) [with _CharT = char; _Traits = 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::basic_ostream<char>::_ios_type = std::ios_base]
        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<_CharT, _Traits>:: operator<<(long _n)
        operator<<(long _n)
                                         ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:166:7: 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:170:7: note: candidate: std::basic_ostream<_CharT, _Traits>:: operator<<((unsigned long _n)
        operator<<((unsigned long _n)
                                         ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:170:7: 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:174:7: note: candidate: std::basic_ostream<_CharT, _Traits>:: operator<<((bool _n)
        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 'std::ios_base& (*)(std::ios_base)'
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_ostream<_CharT, _Traits>:: operator<<((unsigned short _n)
        operator<<((unsigned short _n)
                                         ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/bits/ostream.tcc:91:5: note:   no known conversion for argument 1 from 'std::vector<int>' to 'std::ios_base& (*)(std::ios_base)'
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<_CharT, _Traits>:: operator<<((unsigned short _n)
        operator<<((unsigned short _n)
                                         ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/bits/ostream.tcc:105:5: note: candidate: std::basic_ostream<_CharT, _Traits>:: operator<<((long long _n)
        operator<<((long long _n)
                                         ^
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/bits/ostream.tcc:105:5: note:   no known conversion for argument 1 from 'std::vector<int>' to 'std::ios_base& (*)(std::ios_base)'
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<_CharT, _Traits>:: operator<<((unsigned int _n)
        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 'std::ios_base& (*)(std::ios_base)'
/usr/lib/gcc/x86_64-pc-cygwin/5.4.0/include/c++/ostream:192:7: note: candidate: std::basic_ostream<_CharT, _Traits>:: operator<<((long long _n)
        operator<<((long long _n)
                                         ^
/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_ostream<_CharT, _Traits>::operator<<((double _f)
        operator<<((double _f)
                                         ^
/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 'double'
/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_ostream<_CharT, _Traits>::operator<<((float _f)
        operator<<((float _f)
                                         ^
/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<<((long double _f)
        operator<<((long double _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<<((const void*) _ptr)
        operator<<((const void*) _ptr)
                                         ^
/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 int _val)
        operator<<((const int _val)
                                         ^
```

# C++ how not to print a vector

## print-vector.cpp

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

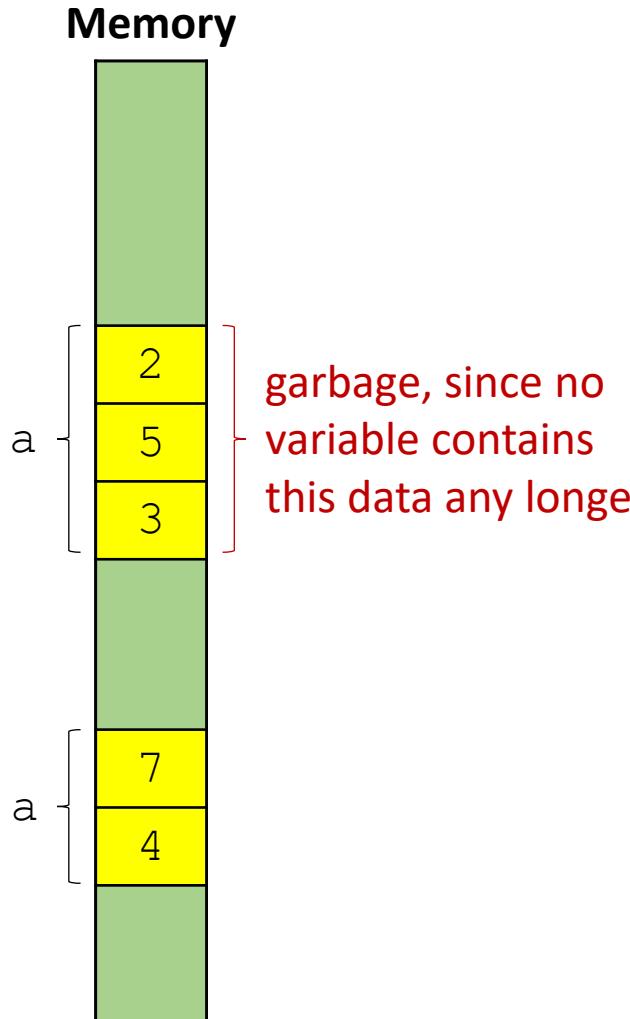
int main() {
    std::vector<int> A = {2, 3};
    std::cout << A << std::endl;
    return 0;
}
```

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



garbage.py

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

a gets new value

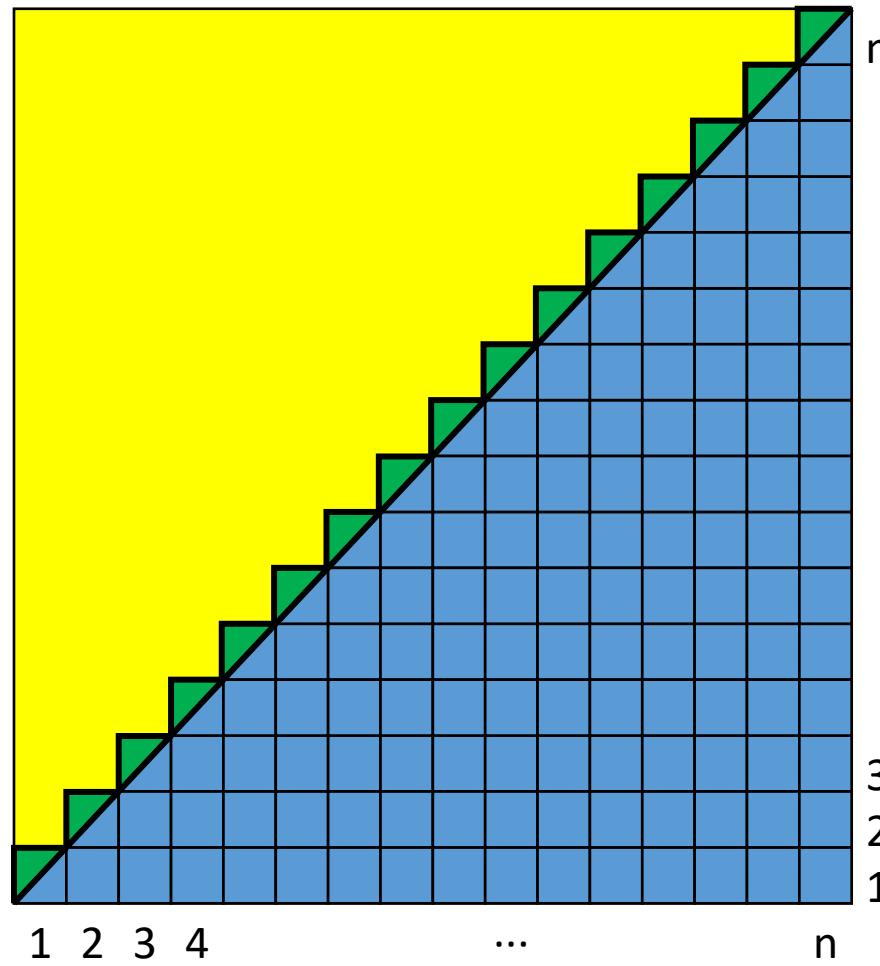
- Python and e.g. Java, C# and JavaScript have a **garbage collector** to automatically recycle garbage
- C and C++ garbage collection must be done explicitly by the program; forgetting to **free** memory again results in **memory leaks** – which can be really hard to find. **Have fun debugging!**
- Automatic garbage collection increases memory safety

# 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  $1 + 2 + 3 + \dots + n = \frac{n^2}{2} + \frac{n}{2}$



$$1 + 2 + \cdots + n$$

### add.py

```
import sys

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

### add.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

```
#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

```
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

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 position	66666666655555554444444433333333222222221111111110000000000 9876543210987654321098765432109876543210987654321098765432109876543210
bin(10**9)	111011100110101100101000000000
bin(500000500000)	101101011110011000100010010010110101101000000
bin(-2004260032+2**32)	10001000100010010110101101000000
bin(5000000050000000)	110111100000101101101011001111000101111110110010100000000
bin(-243309312+2**32)	11110001011111101100101000000000

# Timing results

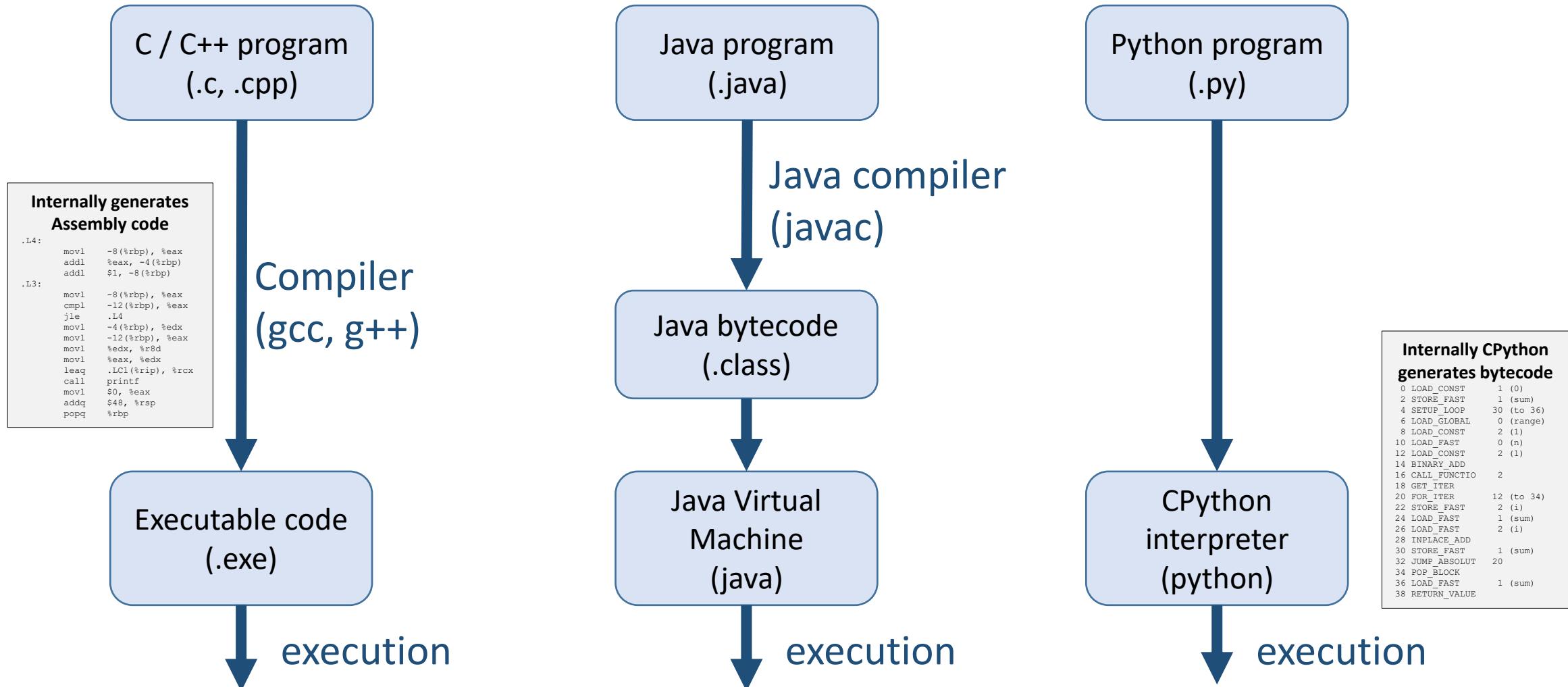
n	C (gcc 9.2)	C++, int (g++ 9.2 )	C++, long (g++ 9.2 )	Java (12.0)	Python (3.8.1)	PyPy (7.3.0)	Python Numba, int64
$10^7$	0.001 sec*	0.001 sec*	0.003 sec	0.006 sec*	1.5 sec	0.27 sec	0.002 sec
$10^9$	0.10 sec**	0.10 sec**	0.30 sec	0.40 sec**	145 sec	27 sec	0.2 sec

- Relative speed

**C ≈ C++ > Java >> Python**

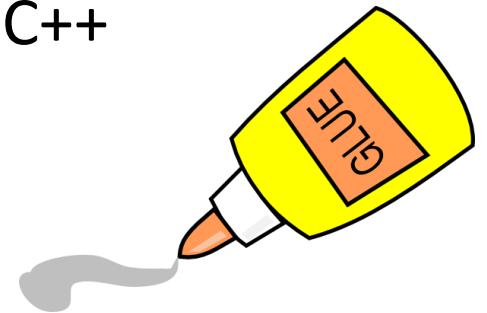
- C, C++, Java need to care about integer overflows – select integer representation carefully with sufficient number of bits (8, 16, 32, 64, 128)
- Python natively works with arbitrary long integers (as memory on your machine allows). Also possible in Java using the class `java.math.BigInteger`
- Python 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

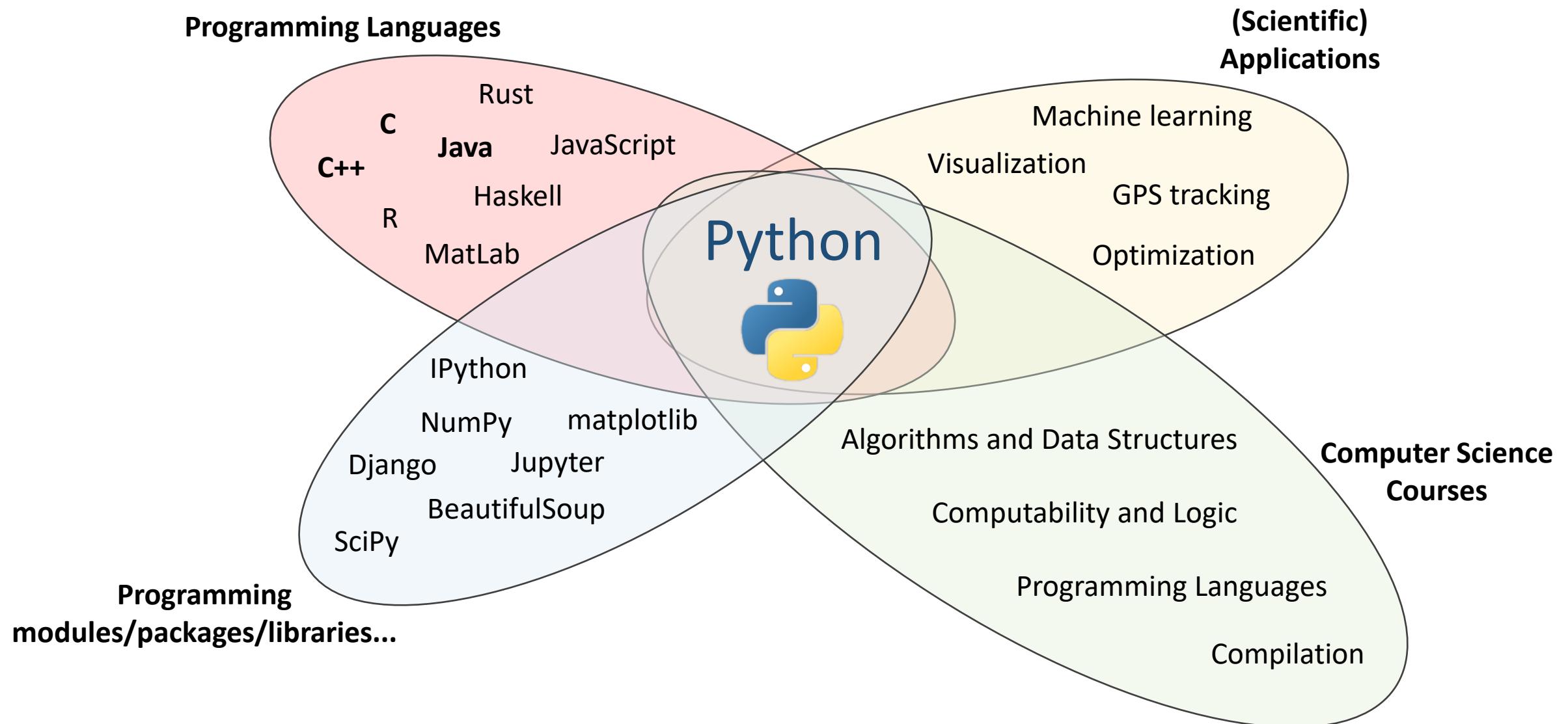


# Why Python ?

- 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



# Course overview

Basic programming  
Advanced / specific python  
Libraries & applications

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 3.13.1
- 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)

# Python.org

The screenshot shows the Python.org homepage with several red arrows highlighting different features:

- A red arrow points from the top-left towards the "Documentation" tab in the main navigation bar.
- A red arrow points from the top-right towards the "Downloads" tab in the main navigation bar.
- A large red diagonal arrow points from the bottom-left towards the "Download Python and IDLE" button.
- A red diagonal arrow points from the top-right towards the "PyPI" section, which displays "+500,000 Python packages".

The page content includes:

- The Python logo and "python™" text.
- A navigation bar with tabs: About, Downloads, Documentation (highlighted), Community, Success Stories, News, and Events.
- A code snippet demonstrating a for loop:

```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

- A section titled "All the Flow You'd Expect" explaining Python's control flow statements.
- A footer with navigation links: Get Started, Download, Docs, and Jobs.
- A footer note: "Looking for work or have a Python related position that you're trying to hire for? Our **relaunched community-run job board** is the place to go."

# Installing Python

The image shows the Python website's download page and a Windows setup window side-by-side.

**Python Website (Left):**

- The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community.
- The main header features the Python logo and the word "python".
- The "Downloads" menu item is highlighted with a red circle labeled 1.
- The "Download for Windows" section is shown, with the "Python 3.11.1" button highlighted with a red circle labeled 2.
- A note states: "Note that Python 3.9+ cannot be used on Windows 7 or earlier."
- Other download options include All releases, Source code, Windows, macOS, Other Platforms, License, and Alternative Implementations.

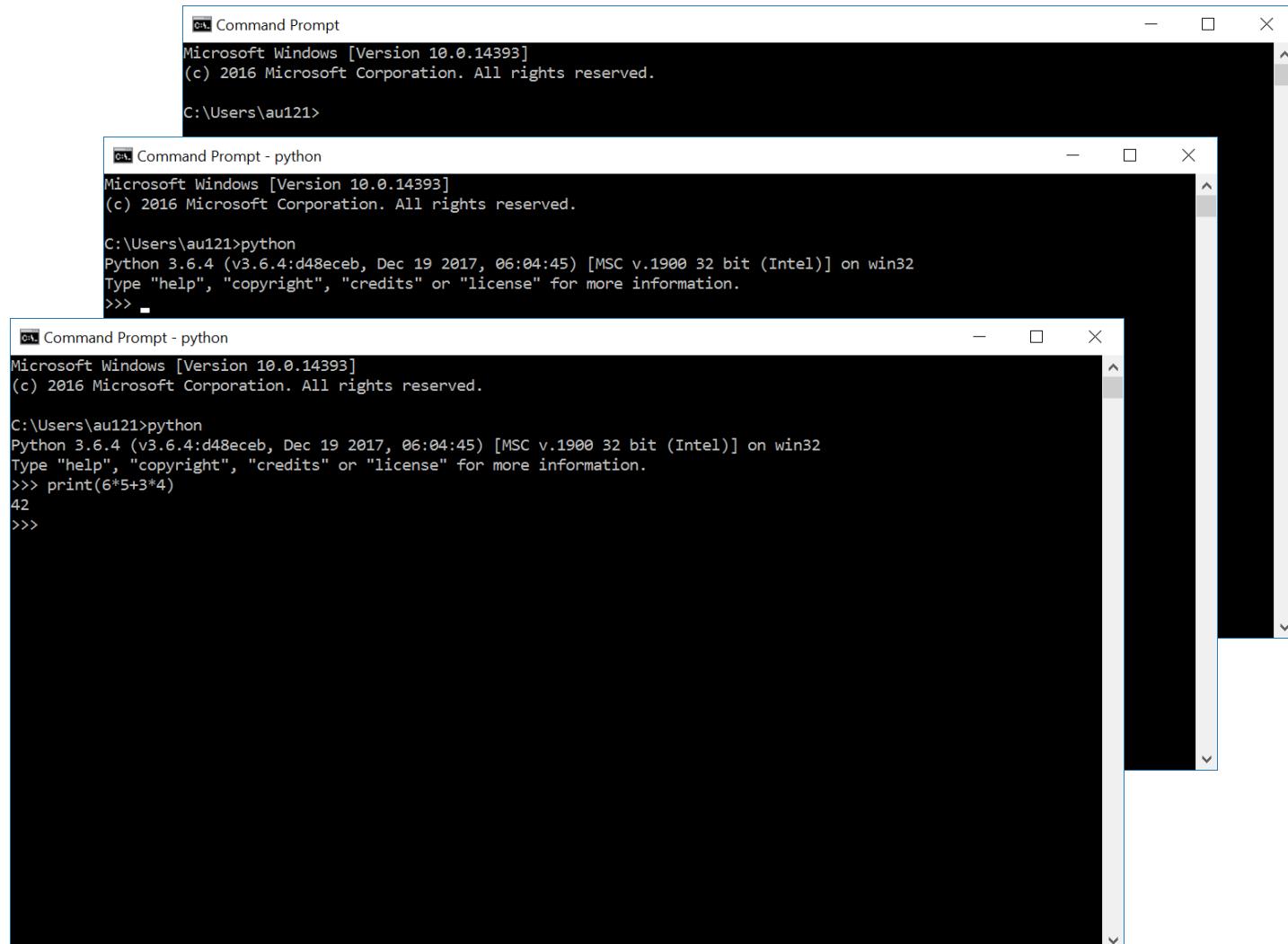
**Windows Installer (Right):**

- The title bar says "Python 3.11.1 (64-bit) Setup".
- The main window title is "Install Python 3.11.1 (64-bit)".
- Text: "Select Install Now to install Python with default settings, or choose Customize to enable or disable features."
- Install Now:** C:\Users\au121\AppData\Local\Programs\Python\Python311 (highlighted with a red circle labeled 4).
- Customize installation:** Choose location and features.
- Checkboxes:
  - Use admin privileges when installing py.exe (highlighted with a red circle labeled 3).
  - Add python.exe to PATH (highlighted with a red circle labeled 3).
- Buttons: "Cancel".

**IMPORTANT** (Large red arrow pointing to the "Add python.exe to PATH" checkbox)

# 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
- Note: Sometimes “python”  
is installed as “python3”



The image shows three stacked screenshots of a Microsoft Windows Command Prompt window. The top screenshot shows the initial state with the command prompt 'C:\Users\au121>'. The middle screenshot shows the result of typing 'python' followed by the Python interpreter's welcome message and the prompt 'C:\Users\au121>python'. The bottom screenshot shows the result of typing 'print(6\*5+3\*4)' followed by the output '42' and the prompt 'C:\Users\au121>'. The windows have standard minimize, maximize, and close buttons.

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

C:\Users\au121>

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

C:\Users\au121>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> -
```

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

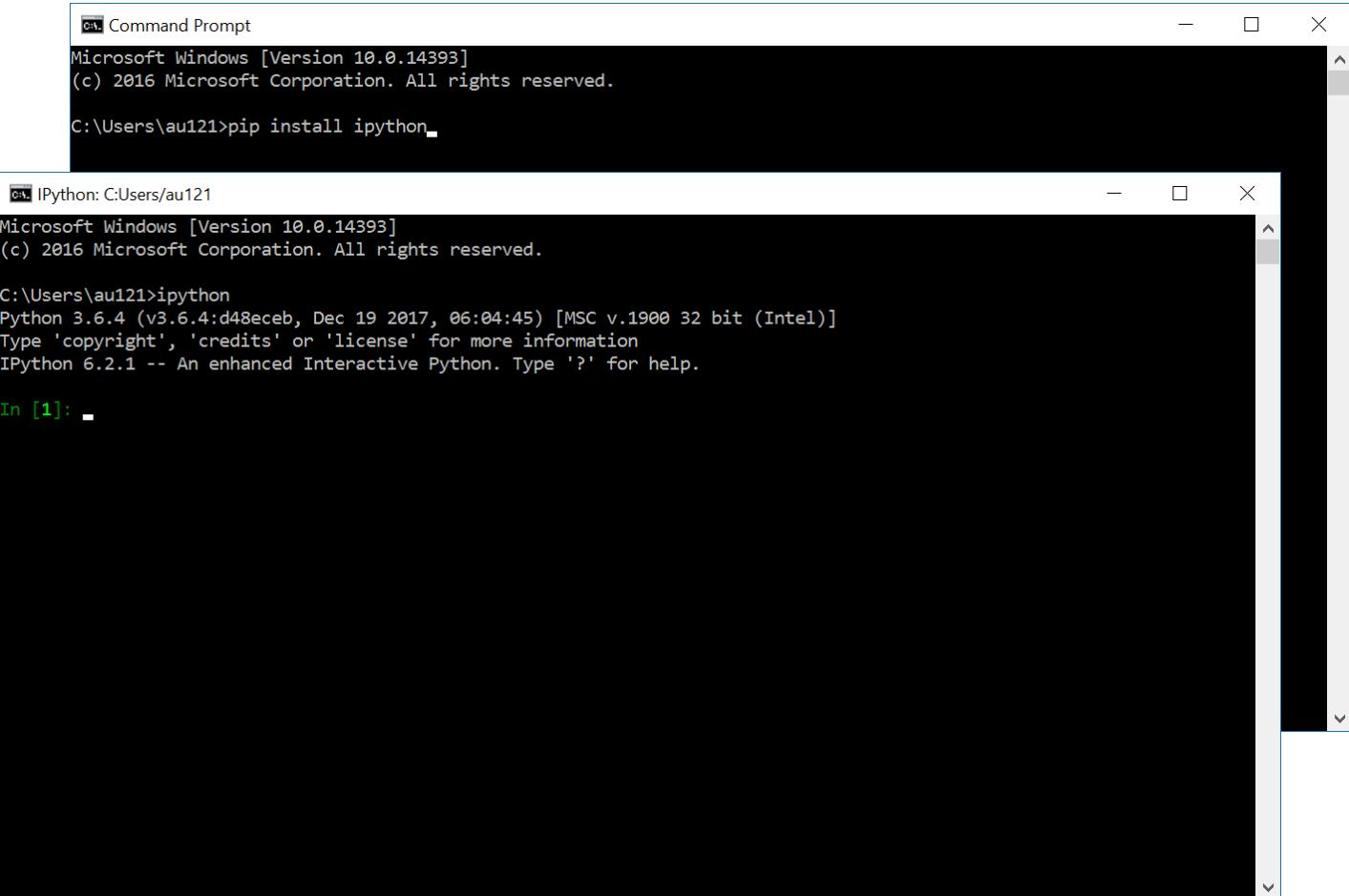
C:\Users\au121>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(6*5+3*4)
42
>>>
```

# Installing IPython – A more powerful interactive Python shell

- Open Command Prompt
  - Execute:  

```
pip install ipython
```
  - Start IPython  

```
ipython
```
- 
- pip = the Python package manager
  - Note: Sometimes “pip” is installed as “pip3”



The image shows a Windows desktop with two windows open side-by-side. The left window is a 'Command Prompt' window titled 'Command Prompt' with the path 'C:\Users\au121>'. It contains the command 'pip install ipython'. The right window is an 'IPython' window titled 'IPython: C:\Users\au121'. It shows the Python interpreter starting up, including the version (Python 3.6.4), build date (Dec 19 2017, 06:04:45), and build details ([MSC v.1900 32 bit (Intel)]). It also displays the IPython version (6.2.1) and a prompt 'In [1]:'.

# Some other usefull packages

- Try installing some more Python packages:

pip install numpy

linear algebra support (N-dimensional arrays)

pip install scipy

numerical integration and optimization

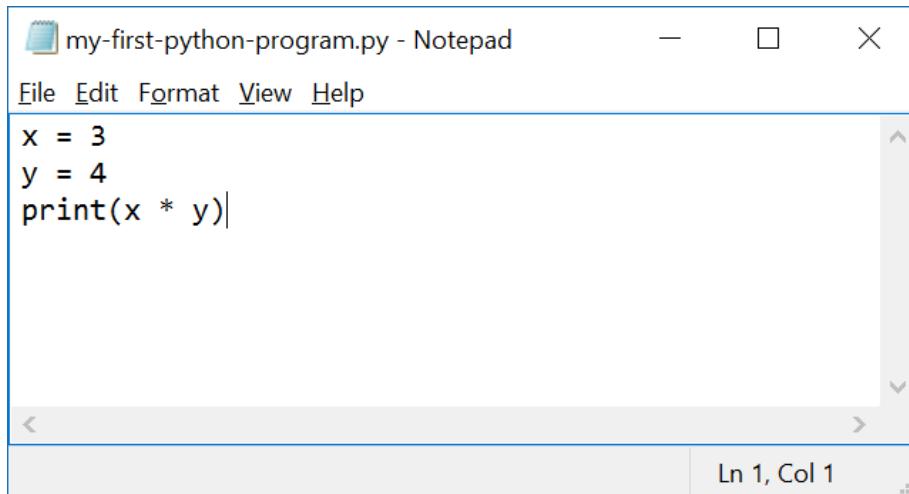
pip install matplotlib

2D and 3D plotting library

pip install pylint

Python source code analyzer enforcing a coding standard

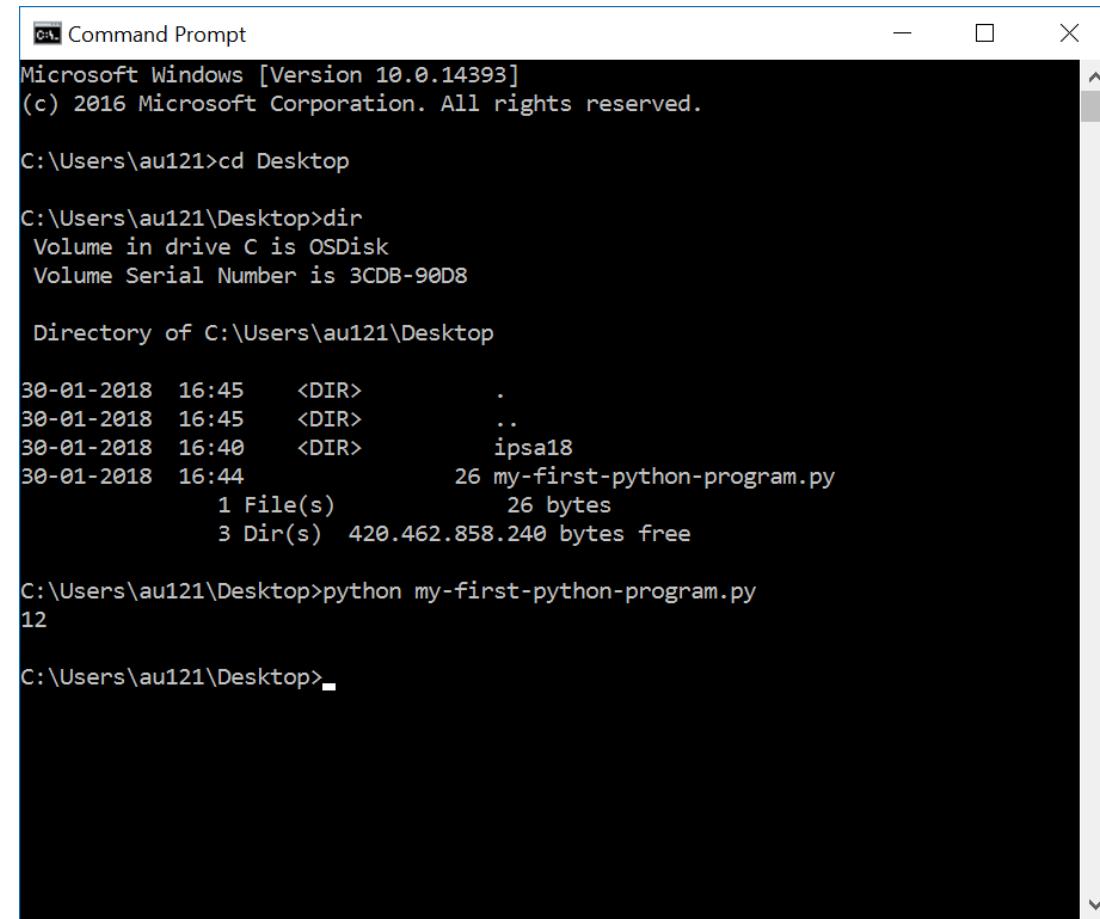
# Creating a Python program the very basic way



A screenshot of a Windows Notepad window titled "my-first-python-program.py - Notepad". The window contains the following Python code:

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

The status bar at the bottom right shows "Ln 1, Col 1".



A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the following terminal session:

```
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 (orTextEdit 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



```
my-first-python-program.py - C:\Users\au121\Desktop\my-first-python-program.py (3.11.0)
File Edit Format Run Options Window Help
1 x = 3
2 y = 4
3 print(x *y)
4
Ln: 4 Col: 0
```

```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\au121\Desktop\my-first-python-program.py =====
12
>>>
Ln: 6 Col: 0
```

- IDLE ships with Python from [python.org](https://python.org)
- Good beginner IDE (Integrated Development Environment)

# The Python Ecosystem

- **Interpreters/compiler**
  - CPython – reference C implementation from [python.org](http://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](http://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](http://docs.python.org/3/library/idle.html))
  - Anaconda w. Spyder, IPython ([www.anaconda.com/download](http://www.anaconda.com/download))
  - Canopy ([enthought.com/product/canopy](http://enthought.com/product/canopy))
  - Visual Studio Code ([code.visualstudio.com](http://code.visualstudio.com))
  - Python tools for Visual Studio ([github.com/Microsoft/PTVS](http://github.com/Microsoft/PTVS))
  - PyCharm ([www.jetbrains.com/pycharm/](http://www.jetbrains.com/pycharm/))
  - Emacs (Python mode and ElPy mode)
  - Notepad++

Good beginner Python IDE

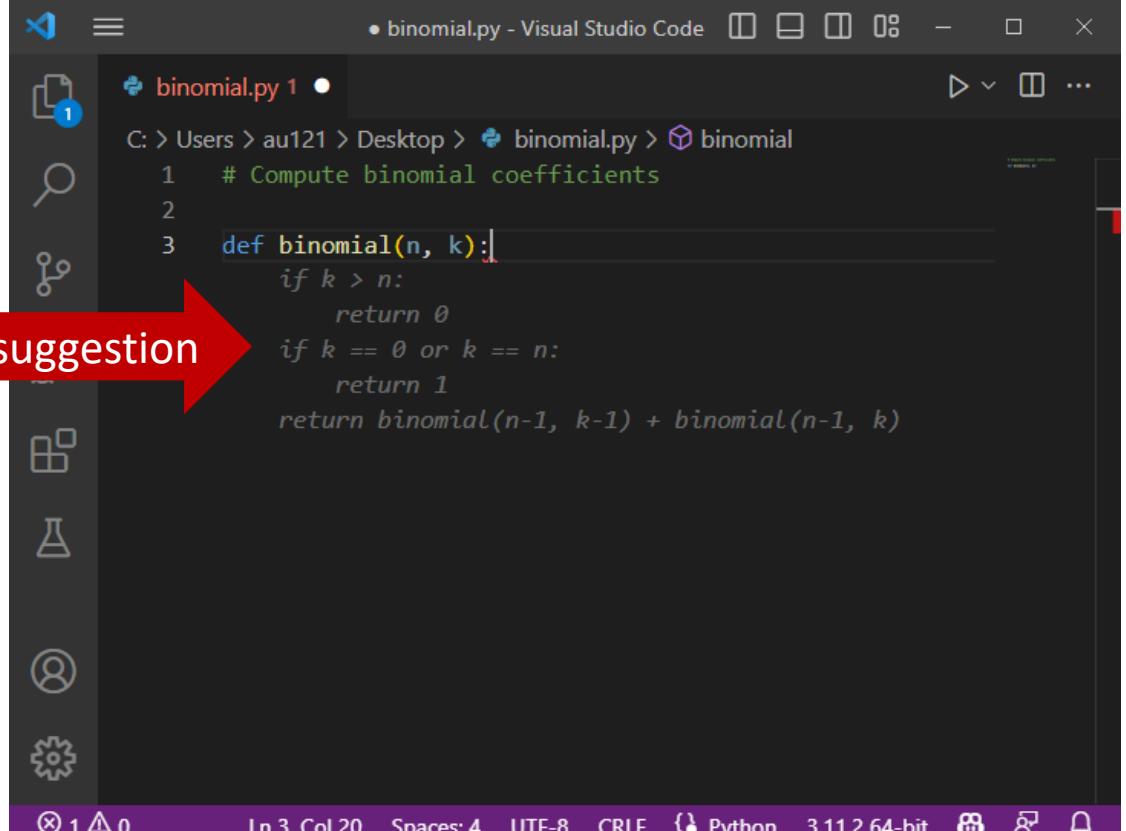
Visual Studio Code is used by more than twice as many developers than its nearest alternative, [Stack overflow survey 2025](#)
- **Python Style guide (PEP8)**
  - pylint, pep8, flake8
- **Python online**
  - Google colab ([colab.research.google.com](http://colab.research.google.com)), repl.it, [sagemath.org](http://sagemath.org), ...
- **Pair programming plugins (share your editor with others)**
  - PyCharm : [Code With Me](#)
  - Visual Studio Code : [Live Share](#)

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
- Interacting with an AI assistant can be a great programming tutor
-  ■ AI assistants are not allowed at the exam



A screenshot of the Visual Studio Code interface. The title bar shows "binomial.py - Visual Studio Code". The left sidebar has a file icon with a blue dot (1) and a search icon. The main editor area contains Python code for calculating binomial coefficients:

```
C: > Users > au121 > Desktop > binomial.py > binomial
1 # Compute binomial coefficients
2
3 def binomial(n, k):
    if k > n:
        return 0
    if k == 0 or k == n:
        return 1
    return binomial(n-1, k-1) + binomial(n-1, k)
```

A red arrow points from the word "suggestion" to the code completion placeholder "def binomial(n, k):". The status bar at the bottom shows "Ln 3, Col 20" and "Python 3.11.2 64-bit".

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 saves me typing."*  
[Python and the Future of Programming](#), Guido van Rossum interviewed by Lex Fridman

# PROGRAMMER'S PROBLEM

