Fundamentals of Computer Science 30398

Lecture 1

Introduction

Jarosław (Jarek) Błasiok
(jaroslaw.blasiok@unibocconi.it)

For office hours send me an email.

Tools we are using

- Python 3 (at least 3.6 — you are unlikely to easily download older version)
- Visual Studio Code; Extensions:
  - Python
  - Code Runner
  - Jupyter
- You might want to check out Jupyter Notebook as well. Run it and compare experience.
- All of the above are included in the python distribution Anaconda.

Programming session organization

- Mostly I will do live coding, explaining what I do.
- You are expected to follow along
  (bring your laptop to the class).
- Ask questions!
- We will do some exercises in the class.
- I will give some exercises to do on your own at home.

Study material

- Slides available on Piazza.
- You will get homework exercises, and access to past exams.
- You can complement the slides with the "Think Python 2e" (or any other introductory book/internet tutorial).

Learning Python vs learning programming

- Python is one of thousands of programming languages (and one of dozens of popular ones).
- Tools change much faster than concepts.
- Algorithmic/Programming thinking is essential conceptual milestone.

How to learn?

- Reading (from books/online material) gives almost nothing.
- You need to try things, experiment!
- When stuck, you can find answer on stackoverflow, ask ChatGPT, or ask on Piazza.
- Do not copy-paste the solution! Read it, close it, and try to write on your own.

High-level advise

- Do not get scared!
- Do not get (too) frustrated.
- Do not get frustrated too quickly.

Programming languages

- Learning first programming language is hardest. Learning second is easier, learning third is easier still...
- Eventually you will know 2-3 programming languages, you will have forgotten 10 more.
- But you will have confidence that you can learn any language you need for the task in 2-3 days.

Programming languages — levels of abstraction

- Machine code
- Assembly
- (Low-level programming languages:) C, Go, ...
- (Mid-level programming languages:) C++, Java, ...
- (High-level programming languages:) Python, JavaScript, ...

Programming languages — levels of abstraction

- Typically: higher level of abstraction leads to faster programming, but slower program.
- Low-level programs are more machine-specific.
  E.g. each processor type has each own assembly.
- C code, with some care, can be written in a portable way.
- High-level languages can usually be run on almost every (fast enough) computer.

Programming languages — special purpose vs general purpose

Python is a general purpose programming language. There are also several special purpose programming languages:

- R is tailored for statistic.
- Matlab for numerical analysis.
- Wolfram Mathematica for symbolic manipulations.
- ...

Assembly and Machine Code

```
55
8bec
83ec0c
c745fc0a000000
c745f814000000
c745f400000000
8b45fc
0345f8
8945f4
33c0
8be5
5d
c3
```

```
push    ebp
mov     ebp,esp
sub     esp,0Ch
mov     dword ptr [ebp-4],0Ah
mov     dword ptr [ebp-8],14h
mov     dword ptr [ebp-0Ch],0
mov     eax,dword ptr [ebp-4]
add     eax,dword ptr [ebp-8]
mov     dword ptr [ebp-0Ch],eax
xor     eax,eax
mov     esp,ebp
pop     ebp
ret
```

C code

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

int main(void) {
    int n;
    scanf("%d", &n);
    int result = 1;
    for (int i=1; i<=n; ++i) {
            result *= i;
    }
    printf("%d\n", result);
};
~
~
~
~
```

Python code; VS code environment; Jupyter notebook

Code golfing — some niche programming languages?

`https://codegolf.stackexchange.com`

Debugging

- Once you write a program, it is usually wrong.
- Debugging is a (usually time-consuming) process of detecting, finding and fixing errrors (bugs).

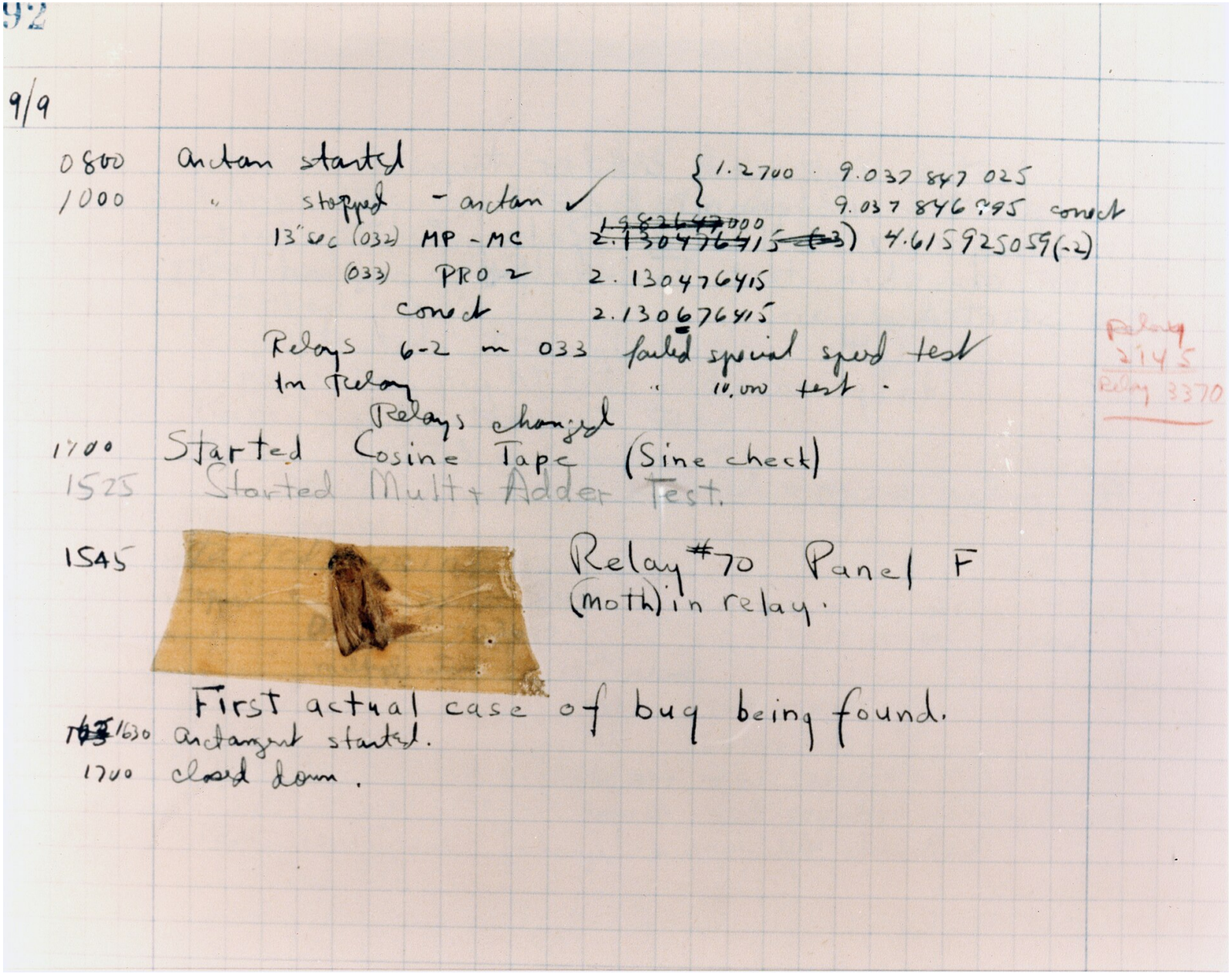
Debugging

- Once you write a program, it is usually wrong.
- Debugging is a (usually time-consuming) process of detecting, finding and fixing errrors (bugs).
- Fixing is usually relatively easier. Finding, and especially detecting errors tend to be very time-consuming.

Debugging

- Once you write a program, it is usually wrong.
- Debugging is a (usually time-consuming) process of detecting, finding and fixing errrors (bugs).
- Fixing is usually relatively easier. Finding, and especially detecting errors tend to be very time-consuming.
- We tend to auto-correct minor mistakes when reading, deducing the intent of the writer. Computers will not do that.

Debugging — first computer bug (Harvard Mark II, 1947)



Syntax errors

Each language has its syntax — it is a formal language, and the computer is extremely picky about following the syntax exactly. Syntax errors tend to be easily detected on an early phase.

Example

```
area = (2 + 2 * 2
print(area)
```

Syntax errors

Each language has its syntax — it is a formal language, and the computer is extremely picky about following the syntax exactly. Syntax errors tend to be easily detected on an early phase.

Example

```
area = (2 + 2 * 2
print(area)
```

What happens

```
Cell In[4], line 1
    a = (2 + 2 * 2
                  ^
SyntaxError: incomplete input
```

Installing Anaconda

Looking at some examples