
Introducing Python Programming in the *Algorithms Design* Course

Costin Bădică, Alex Becheru
Ionuț Murărețu

Department of Computers and Information Technology
University of Craiova, Romania



Department of Computers and Information Technology

Cooperation at Academic Informatics Education across Balkan Countries and Beyond
Primošten, Croatia, September 2-8, 2018

September 03, 2018

Talk Outline

- Course Curricula Background
- Motivation
- Introducing Python
- Course Upgrade with Python topics
- Facts about Using Python
- Conclusions

Overview



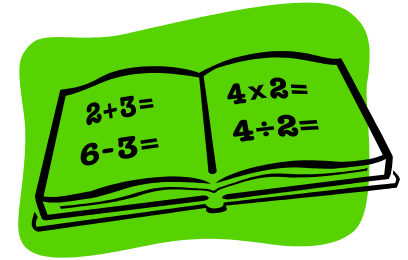
- ***Algorithms Design*** (former *Programming Techniques*)
 - Analysis, design, programming, experimenting fundamental algorithms
 - Alignment with CS curricula recommended by ACM and IEEE
 - 1st year, 2nd semester
- Courses that must be passed before AD:
 - Computer Programming
- Courses that benefit from AD:
 - Object-Oriented Programming
 - Data Structures and Algorithms
 - Artificial Intelligence

Overview – Learning Objectives



- **LO1:** *To introduce the principles of algorithm analysis, modular programming and data abstraction.*
- **LO2:** *To introduce fundamental algorithms and the fundamental methods of algorithm design.*
- **LO3:** *To develop practical experience in programming small-scale experiments involving implementation, testing and evaluation of algorithms.*

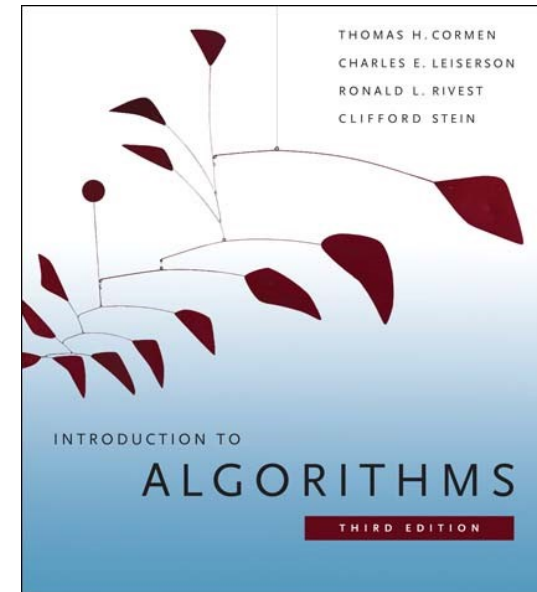
Overview – Topics



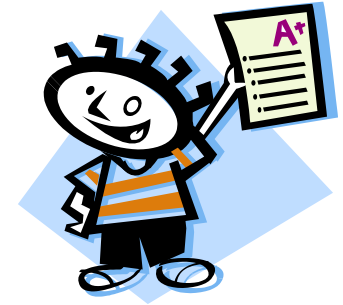
- ❑ Introduction to analysis and design of algorithms
- ❑ Divide and conquer
- ❑ Correctness and testing of algorithms
- ❑ Sorting algorithms
- ❑ Abstract data types
- ❑ Stacks and queues
- ❑ Graphs and trees
- ❑ Dynamic programming
- ❑ Greedy algorithms
- ❑ Backtracking
- ❑ Introduction to NP-completeness

Overview – Structure

- No single textbook; a good base is CLRS3 book.
- 2 modules:
 - Course (4 ECTS points)
 - Project (1 ECTS points)
- Both duration is 14 weeks:
 - Course: 2 h lectures/week (28h) + 2 h lab/week (28h)
 - Project: 1 h project/week (14h)



Overview – Grading



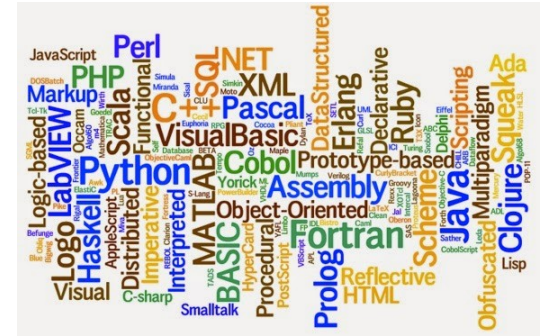
- Course module: final exam (70%)
 - Exercise: discuss, analyze, improve simple algorithm
 - Exercise: design and code a small-scale C program for solving an algorithmic problem
 - Exercise: algorithm design using fundamental method
- Course module: laboratory assignments (30%)
- Project module: project assignment
 - 20% intermediary delivery
 - 80% final delivery

Practical Aspects – Programming Language



- We are using **Standard C**
- Reasons:
 - Students learn C in 1st semester at Computer Progr.
 - C gives base for learning C-like lang: C++, Java, C#
 - C is defined as *high-level assembly language*, useful for:
 - Operating systems
 - Embedded systems
 - Compilers
 - C enables efficient implementation of algorithms

World of Programming Languages



■ Imperative vs Declarative Paradigms:

□ Imperative (state-oriented): focused on “how?”

- Procedural (von Neumann): C, Ada, Fortran
- Object-oriented: C++, Smalltalk, Eiffel, Java

□ Declarative (goal-oriented): focused on “what?”

- Functional: Lisp, Haskell, ML, F# (a kind of ML), Erlang, Haskell
- Logic: Prolog, spreadsheets

■ Compiled vs Interpreted Languages:

□ Compiled: C, Assembler

□ Interpreted (scripting): Perl, Python, PHP, JavaScript

□ Partly compiled & partly interpreted: Java, C#

Why Python?



- Python is an interpreted language, different from C compiled language
- Python is close to pseudocode
- Python is higher-level than C
- Python supports different styles of programming enabling various comparisons in terms of readability / comprehensibility and efficiency / speed
- Python enables fast prototyping & algorithm testing

IEEE Interactive Top of Programming Languages



- 11 metrics and 9 sources => popularity ranking
- Started from more than 300 languages
- Filtered out those with low searches on “X programming”
- Manually narrow down the rest to most “interesting”
- Labeled with one or more categories:
 - Web, mobile, enterprise / desktop, embedded
- Ranking based on metrics, sources + source weights
- 4 default rankings (IEEE Spectrum, Trending, Jobs, Open) as well as manually customizable rankings

2018 vs 2015 Interactive Top

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

Edit Ranking | Remove Comparison |  


























Choose a Comparison (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

Edit Ranking

Language Types (click to hide)

 Web  Mobile  Enterprise  Embedded

Language Rank	Types	Spectrum Ranking	Custom Ranking
1. Python	  	100.0	100.0
2. C++	  	99.7	99.9
3. Java	  	97.5	99.4
4. C	  	96.7	96.6
5. C#	  	89.4	91.5
6. PHP		84.9	85.1
7. R		82.9	84.6
8. JavaScript	 	82.6	83.3
9. Go	 	76.4	76.2
10. Assembly		74.1	73.1
11. Matlab		72.8	72.6
12. Scala	 	72.1	71.3

Choose a Comparison (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

Edit Ranking

Language Types (click to hide)

 Web  Mobile  Enterprise  Embedded

Compare a ranking. Click a data source to toggle its inclusion in the ranking and drag its slider to reweight it.

Use data from: 2018 2017 2016 2015 2014

Google (search)	<input type="range" value="50"/>	50	Google (trends)	<input type="range" value="50"/>	50
Github (active)	<input type="range" value="50"/>	50	Github (created)	<input type="range" value="30"/>	30
Stack Overflow (?s)	<input type="range" value="30"/>	30	Stack Overflow (views)	<input type="range" value="30"/>	30
Reddit	<input type="range" value="20"/>	20	Hacker News	<input type="range" value="20"/>	20
Career Builder	<input type="range" value="5"/>	5	Dice	<input type="range" value="5"/>	5
Twitter	<input type="range" value="20"/>	20	IEEE Xplore	<input type="range" value="100"/>	100

Cancel Save as Custom

Cooperation at Academic Informatics Education across Balkan Countries and Beyond

September 03, 2018

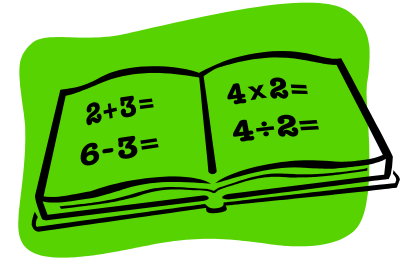
Primošten, Croatia, September 2-8, 2018

Introducing Python



- Python is an *interpreted* language.
- Python is characterized by:
 - Simple and readable syntax
 - Dynamic typing
 - High-level data types
- A Python program is a collection of functions and variables grouped into *modules*.
- A text file *.py* containing Python statements is called a *script*.
- A *module* is a *.py* file (or script) that contains more functions.
- A Python program can be run:
 - Using the Python *interaction mode*
 - Running a script from the command line under Python `ctrl` in *script mode*.

Upgrading AD Course with Python Topics



- Introduce Python using examples of simple algorithms.
- Present Python high-level data types close to the related AD topic – *abstract data types*.
- Use Python flexibility to show how different solutions of the same problem can be implemented, evaluating and comparing:
 - Readability
 - Time complexity
- Use Python tools to explore algorithmic solutions.

Educational Issues I



- Python already provides a variety of high-level data structures of “sequence” type including lists.
- This might be a source of confusion for students, between Python lists and linked lists.
- Approach:
 - When introducing linked lists with algorithms following CLRS textbook, we present also its explicit Python implementation.
 - Then we discuss separately Python lists, highlighting differences, as well as the many features of this structure.

Educational Issues II



- Issues of aliases, shallow and deep copy of Python complex objects is better explained using pointer diagrams.
- This is easier to understand after students are firstly exposed to low-level details of pointers and references, that in our opinion are better introduced using C.
- This discussion closes the gap between high-level Python structures and low-level details that are needed to correctly understand their implementation.

Python Tools

Jupyter Notebook



```
In [70]: """Reciprocal cycles
Problem 26
A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10
are given:

1/2*= ->0.5
1/3*= ->0.(3)
1/4*= ->0.25
1/5*= ->0.2
1/6*= ->0.1(6)
1/7*= ->0.(142857)
1/8*= ->0.125
1/9*= ->0.(1)
1/10-->= ->0.1
Where 0.1(6) means 0.166666..., and has a 1-digit recurring cycle. It can be seen that 1/7 has a 6-digit recurring cycle.

Find the value of d < 1000 for which 1/d contains the longest recurring cycle in its decimal fraction part."""
def cycle_length(n):
    l = 1
    a = 10
    a = (a % n) * 10
    while a != 10:
        l += 1
        a = (a % n) * 10
    return l

dlim = 1000
lmax = 0
dmax = 0
for d in range(2,dlim):
    if d % 2 != 0 and d % 5 != 0:
        l = cycle_length(d)
        if l > lmax:
            lmax = l
            dmax = d

print(dmax)
```

983

Python Tools

Latex2e in Markdown Cells



L^AT_EX

Quadratic primes

Problem 27

Euler discovered the remarkable quadratic formula:

$n^2 + n + 41$

It turns out that the formula will produce 40 primes for the consecutive values $n = 0, 1, \dots, 39$. The formula $40^2 + 40 + 41 = 40(40 + 1) + 41$ is divisible by 41, and certainly when $n = 40$.

The incredible formula $n^2 - 79n + 1601$ was discovered, which produces 80 primes for the consecutive values $n = 0, 1, \dots, 79$. The product of the coefficients, -79 and 1601 , is -126479 .

Considering quadratics of the form:

$n^2 + an + b$, where $|a| < 1000$ and $|b| \leq 1000$

where $|n|$ is the modulus/absolute value of n :

e.g. $|11| = 11$ and $|-4| = 4$

Find the product of the coefficients, a and b , for the quadratic expression that produces the most primes for consecutive values of n , starting with $n = 0$.

Quadratic primes

Problem 27

Euler discovered the remarkable quadratic formula:

$n^2 + n + 41$

It turns out that the formula will produce 40 primes for the consecutive integers $n = 0, 1, \dots, 39$. The formula $40^2 + 40 + 41 = 40(40 + 1) + 41$ is divisible by 41, and certainly when $n = 40$.

The incredible formula $n^2 - 79n + 1601$ was discovered, which produces 80 primes for the consecutive values $n = 0, 1, \dots, 79$. The product of the coefficients, -79 and 1601 , is -126479 .

Considering quadratics of the form:

$n^2 + an + b$, where $|a| < 1000$ and $|b| \leq 1000$

where $|n|$ is the modulus/absolute value of n :

e.g. $|11| = 11$ and $|-4| = 4$

Find the product of the coefficients, a and b , for the quadratic expression that produces the most primes for consecutive values of n , starting with $n = 0$.

Python Tools Magic Commands

`%%time`
`%%timeit`
`%%latex`

- `%%time` will time whatever you evaluate
- `%%timeit` will time whatever you evaluate multiple times and give you the best, and the average times
- `%%latex` will render cell contents as LaTeX

```
In [14]: %%latex  
$$  
\sum_{i=1}^n i = \frac{n(n+1)}{2}  
$$
```

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

Python Flexibility



- Using high-level features of list and set comprehension can result in very compact representation of some algorithms ...
- But they are not always efficient.
- Nevertheless, they can be read as a mathematical language.
- E.g.: How many distinct terms are in the sequence generated by a^b for $2 \leq a \leq 100$ and $2 \leq b \leq 100$?
- In mathematical notation:
$$| \{a^b \mid 2 \leq a \leq 100, 2 \leq b \leq 100\} |$$
- In Python:
`len({ a**b for a in range(2,101) for b in range(2,101) })`
- or a totally different solution ... (on the next slide)

Different Solution



```
def is_power(n):
    s = int(pow(n,0.5))
    for i in range(2,s+1):
        if n % i == 0:
            p = 1
            m = n // i
            while m % i == 0:
                p += 1
                m = m // i
            if m == 1:
                return p
    return 0

vmax = 100
counter = 0
for a in range(2,vmax+1):
    p = is_power(a)
    if p == 0:
        counter += vmax-1
    else:
        delta = vmax - 1
        for b in range(2,vmax+1):
            for q in range(1,p):
                if ((p*b) % q == 0) \
                    and (((p*b)//q) \
                        <= vmax):
                    delta -= 1
                    break
            counter += delta
print(counter)
```

How Can We Compare Them ?



- **Comprehensibility:** Homework
- **Theoretical Time Complexity:** Homework
- **Running Time:**

%%timeit -n50 -r10 produces:

19.6 ms ± 1.62 ms per loop (mean ± std. dev. of 10 runs, 50 loops each) for the 1st solution

3.2 ms ± 502 μs per loop (mean ± std. dev. of 10 runs, 50 loops each) for the 2nd solution

Python Usage Figures

Project Euler.net



- Web site containing a list of computational problems intended to be solved with computer programs.
 - Total of 829911 registered members.
 - 101 programming languages are used to solve the problems.
- Number of members using:
 - Python 50588
 - C/C++ 42919
 - Java 29012
 - C# 13539
 - Haskell 6797

Python Usage Figures in AI Course



- Course assignment of Artificial Intelligence (AI) course:
 - Compare two search algorithms for a given problem
 - Students could choose the programming language. Suggested languages were: C, C++, Java, Prolog, and Python
- 66 students from 120 submitted their homework.
- Statistics:

Java	19
C++	21
Python	21
C#	1
Prolog	2
C	1

Conclusions



- We proposed approaches for introducing Python to AD course.
- We presented few issues regarding our proposal and proposed measures how to deal with them.
- More results are needed to properly assess our proposals.

