

Welcome to MCS 275

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- 1 About the Course
Course Content
Prerequisites & Expectations
- 2 Introduction to Programming
Scripting in Python
from OOP to LAMP
example: Factorization in Primes
- 3 Introduction to Computing
numpy, scipy, and Sage
- 4 Summary

MCS 275 Lecture 1
Programming Tools and File Management
Jan Vershelde, 11 January 2010

Welcome to MCS 275

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- 1 About the Course
Course Content
Prerequisites & Expectations
- 2 Introduction to Programming
Scripting in Python
from OOP to LAMP
example: Factorization in Primes
- 3 Introduction to Computing
numpy, scipy, and Sage
- 4 Summary

Content of the Course

recommended Text Books

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- (1) J.G. Brookshear *Computer Science. An Overview*. 9th or 10th Edition Addison-Wesley 2007, 2009.
→ intro to CS, examples of C and C++
- (2) B.N. Miller and D.L. Ranum: *Python Programming in Context*. Jones and Bartlett 2009.
→ good introduction to Python for mcs 260,
for mcs 275: chapters 7 to 9

The material between midterms is still based on
Rashi Gupta: *Making Use of Python*. Wiley 2002.
Overview:

- 1 Python glues software components
- 2 programming techniques, e.g.: recursion
- 3 applications to science & engineering
- 4 algorithms and data structures

Content of the Course

recommended Text Books

- (1) J.G. Brookshear *Computer Science. An Overview*. 9th or 10th Edition Addison-Wesley 2007, 2009.
→ intro to CS, examples of C and C++
- (2) B.N. Miller and D.L. Ranum: *Python Programming in Context*. Jones and Bartlett 2009.
→ good introduction to Python for mcs 260,
for mcs 275: chapters 7 to 9

The material between midterms is still based on
Rashi Gupta: *Making Use of Python*. Wiley 2002.

Overview:

- 1 Python glues software components
- 2 programming techniques, e.g.: recursion
- 3 applications to science & engineering
- 4 algorithms and data structures

Content of the Course

recommended Text Books

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- (1) J.G. Brookshear *Computer Science. An Overview*. 9th or 10th Edition Addison-Wesley 2007, 2009.
→ intro to CS, examples of C and C++
- (2) B.N. Miller and D.L. Ranum: *Python Programming in Context*. Jones and Bartlett 2009.
→ good introduction to Python for mcs 260,
for mcs 275: chapters 7 to 9

The material between midterms is still based on Rashi Gupta: *Making Use of Python*. Wiley 2002.
Overview:

- 1 Python glues software components
- 2 programming techniques, e.g.: recursion
- 3 applications to science & engineering
- 4 algorithms and data structures

Revised Catalog Description for mcs 275

still in progress...

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Programming in Python, graphical user interfaces.

Recursive problem solving, enumeration and backtracking.

Recursive data structures, divide and conquer.

Programming with CGI, MySQL, sockets, threads,
and web servers.

Welcome to MCS 275

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

- 1 About the Course
Course Content
Prerequisites & Expectations
- 2 Introduction to Programming
Scripting in Python
from OOP to LAMP
example: Factorization in Primes
- 3 Introduction to Computing
numpy, scipy, and Sage
- 4 Summary

Prerequisites & Expectations

MCS 275 follows MCS 260

About the Course

Course Content

Prerequisites & Expectations

Introduction to Programming

Scripting in Python

from OOP to LAMP

example:

Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

What you should know already:

- 1 basic computer/computing literacy; and
- 2 elements of algorithm and program design.

Programming involves understanding of specifications, design of solution methods, definition of algorithms, and coding (or implementation).

Only coding depends on a programming language.

Programming is a skill, we **learn by doing**.

Therefore,

- 1 emphasis on five computer projects
- 2 active participation in the lab sessions

Prerequisites & Expectations

MCS 275 follows MCS 260

About the Course

Course Content

Prerequisites & Expectations

Introduction to Programming

Scripting in Python

from OOP to LAMP

example:

Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

What you should know already:

- 1 basic computer/computing literacy; and
- 2 elements of algorithm and program design.

Programming involves understanding of specifications, design of solution methods, definition of algorithms, and coding (or implementation).

Only coding depends on a programming language.

Programming is a skill, we **learn by doing**.

Therefore,

- 1 emphasis on five computer projects
- 2 active participation in the lab sessions

Prerequisites & Expectations

MCS 275 follows MCS 260

About the Course

Course Content

Prerequisites & Expectations

Introduction to Programming

Scripting in Python

from OOP to LAMP

example:

Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

What you should know already:

- 1 basic computer/computing literacy; and
- 2 elements of algorithm and program design.

Programming involves understanding of specifications, design of solution methods, definition of algorithms, and coding (or implementation).

Only coding depends on a programming language.

Programming is a skill, we **learn by doing**.

Therefore,

- 1 emphasis on five computer projects
- 2 active participation in the lab sessions

Welcome to MCS 275

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

- 1 About the Course
Course Content
Prerequisites & Expectations
- 2 Introduction to Programming
Scripting in Python
from OOP to LAMP
example: Factorization in Primes
- 3 Introduction to Computing
numpy, scipy, and Sage
- 4 Summary

[About the Course](#)

[Course Content](#)
[Prerequisites & Expectations](#)

[Introduction to Programming](#)

[Scripting in Python](#)
from OOP to LAMP
example:
[Factorization in Primes](#)

[Introduction to Computing](#)

[numpy, scipy, and Sage](#)

[Summary](#)

Scripting in Python

programming in the small

- **rapid prototyping**
- use at command prompt (the “shell”) or in integrated development environment (IDLE)
- Python is interpreted
 - + runs immediately
 - + dynamic typing and garbage collection
 - not as efficient as compiled code
- Python is open to C extensions
 - computationally intensive portions in C

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

Scripting in Python

programming in the small

- rapid prototyping
- use at command prompt (the “shell”) or in integrated development environment (IDLE)
- Python is interpreted
 - + runs immediately
 - + dynamic typing and garbage collection
 - not as efficient as compiled code
- Python is open to C extensions
 - computationally intensive portions in C

[About the Course](#)

[Course Content](#)
[Prerequisites & Expectations](#)

[Introduction to Programming](#)

[Scripting in Python](#)
from OOP to LAMP
example:
[Factorization in Primes](#)

[Introduction to Computing](#)

[numpy, scipy, and Sage](#)

[Summary](#)

Scripting in Python

programming in the small

- rapid prototyping
- use at command prompt (the “shell”) or in integrated development environment (IDLE)
- Python is interpreted
 - + runs immediately
 - + dynamic typing and garbage collection
 - not as efficient as compiled code
- Python is open to C extensions
→ computationally intensive portions in C

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

Scripting in Python

programming in the small

- rapid prototyping
- use at command prompt (the “shell”) or in integrated development environment (IDLE)
- Python is interpreted
 - + runs immediately
 - + dynamic typing and garbage collection
 - not as efficient as compiled code
- Python is open to C extensions
 - computationally intensive portions in C

Welcome to MCS 275

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

- 1 About the Course
Course Content
Prerequisites & Expectations
- 2 Introduction to Programming
Scripting in Python
from OOP to LAMP
example: Factorization in Primes
- 3 Introduction to Computing
numpy, scipy, and Sage
- 4 Summary

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

from OOP to LAMP

programming in the large

- OOP = Object Oriented Programming
→ organization of code to promote
reuse, change, and scalability
- The Open Source revolution has given us
 - L Linux, operating system
 - A Apache, web server
 - M MySQL, database
 - P Python, scripting languagePython glues software components

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

from OOP to LAMP

programming in the large

- OOP = Object Oriented Programming
→ organization of code to promote
reuse, change, and scalability
- The Open Source revolution has given us

L Linux, operating system

A Apache, web server

M MySQL, database

P Python, scripting language

Python glues software components

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

from OOP to LAMP

programming in the large

- OOP = Object Oriented Programming
→ organization of code to promote
reuse, change, and scalability
- The Open Source revolution has given us
 - L Linux, operating system
 - A Apache, web server
 - M MySQL, database
 - P Python, scripting languagePython glues software components

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

from OOP to LAMP

programming in the large

- OOP = Object Oriented Programming
→ organization of code to promote
reuse, change, and scalability
- The Open Source revolution has given us
 - L Linux, operating system
 - A Apache, web server
 - M MySQL, database
 - P Python, scripting languagePython glues software components

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

from OOP to LAMP

programming in the large

- OOP = Object Oriented Programming
→ organization of code to promote
reuse, change, and scalability
- The Open Source revolution has given us
 - L Linux, operating system
 - A Apache, web server
 - M MySQL, database
 - P Python, scripting languagePython glues software components

Welcome to MCS 275

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- 1 About the Course
Course Content
Prerequisites & Expectations
- 2 Introduction to Programming**
Scripting in Python
from OOP to LAMP
example: Factorization in Primes
- 3 Introduction to Computing
numpy, scipy, and Sage
- 4 Summary

Factorization in Primes

an example program

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

A prime has only two divisors: 1 and itself.

Every natural number can be factored
as a unique product of primes.

Problem (input/output specification):

input : a natural number n
output : a list of primes $[p_1, p_2, \dots, p_k]$,
$$n = p_1 \times p_2 \times \dots \times p_k$$

A related (easier) problem:

input : a natural number n
output : a list of all divisors of n

Let us look at the easier problem first.

Factorization in Primes

an example program

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

A prime has only two divisors: 1 and itself.

Every natural number can be factored as a unique product of primes.

Problem (input/output specification):

input : a natural number n
output : a list of primes $[p_1, p_2, \dots, p_k]$,
$$n = p_1 \times p_2 \times \dots \times p_k$$

A related (easier) problem:

input : a natural number n
output : a list of all divisors of n

Let us look at the easier problem first.

Factorization in Primes

an example program

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

A prime has only two divisors: 1 and itself.

Every natural number can be factored as a unique product of primes.

Problem (input/output specification):

input : a natural number n
output : a list of primes $[p_1, p_2, \dots, p_k]$,
$$n = p_1 \times p_2 \times \dots \times p_k$$

A related (easier) problem:

input : a natural number n
output : a list of all divisors of n

Let us look at the easier problem first.

Enumerating all Divisors

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Problem (input/output specification):

input : a natural number n
output : a list of all nontrivial divisors of n
(1 and n are trivial divisors)

first, we care about clarity and correctness

Ingredients in a *first* algorithm:

- 1 d divides n if $n \% d == 0$ (zero remainder)
- 2 enumerate all candidate divisors, for d ranging between 2 and $n - 1$, for d in `range(0, n)`
- 3 append divisor d to a list L : `L.append(d)`

Now we will put the ingredients into a recipe.

Enumerating all Divisors

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Problem (input/output specification):

- input : a natural number n
output : a list of all nontrivial divisors of n
(1 and n are trivial divisors)

first, we care about clarity and correctness

Ingredients in a *first* algorithm:

- 1 d divides n if $n \% d == 0$ (zero remainder)
- 2 enumerate all candidate divisors, for d ranging between 2 and $n - 1$, for d in `range(0, n)`
- 3 append divisor d to a list L : `L.append(d)`

Now we will put the ingredients into a recipe.

Enumerating all Divisors

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Problem (input/output specification):

- input : a natural number n
output : a list of all nontrivial divisors of n
(1 and n are trivial divisors)

first, we care about clarity and correctness

Ingredients in a *first* algorithm:

- 1 d divides n if $n \% d == 0$ (zero remainder)
- 2 enumerate all candidate divisors, for d ranging between 2 and $n - 1$, for d in `range(0, n)`
- 3 append divisor d to a list L : `L.append(d)`

Now we will put the ingredients into a recipe.

Enumerating all Divisors

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Problem (input/output specification):

- input : a natural number n
output : a list of all nontrivial divisors of n
(1 and n are trivial divisors)

first, we care about clarity and correctness

Ingredients in a *first* algorithm:

- 1 d divides n if $n \% d == 0$ (zero remainder)
- 2 enumerate all candidate divisors, for d ranging between 2 and $n - 1$, for d in `range(0, n)`
- 3 append divisor d to a list L : `L.append(d)`

Now we will put the ingredients into a recipe.

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

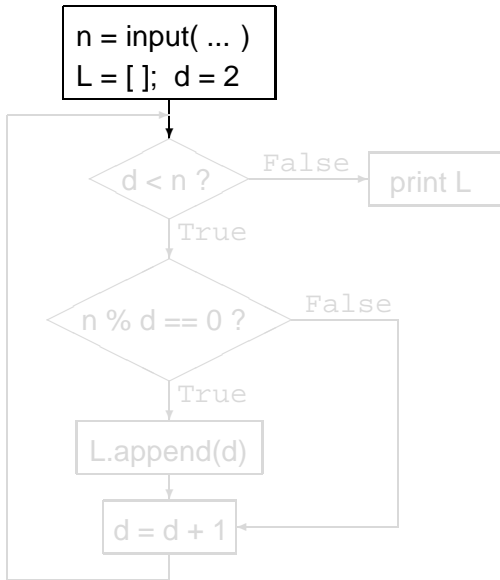
Introduction to Computing

numpy, scipy, and
Sage

Summary

Flowchart

enumerating all divisors



About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

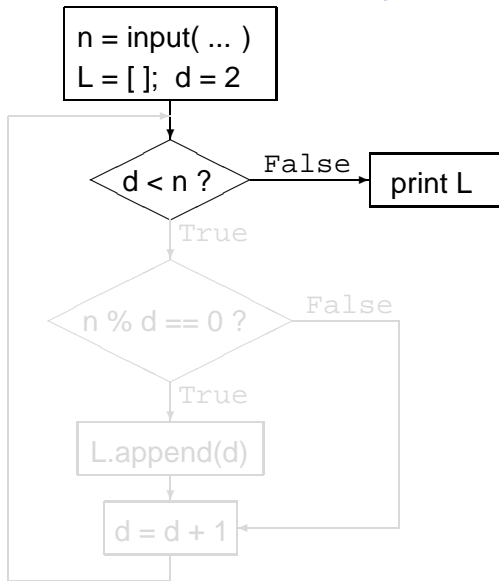
Introduction to Computing

numpy, scipy, and
Sage

Summary

Flowchart

enumerating all divisors



About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

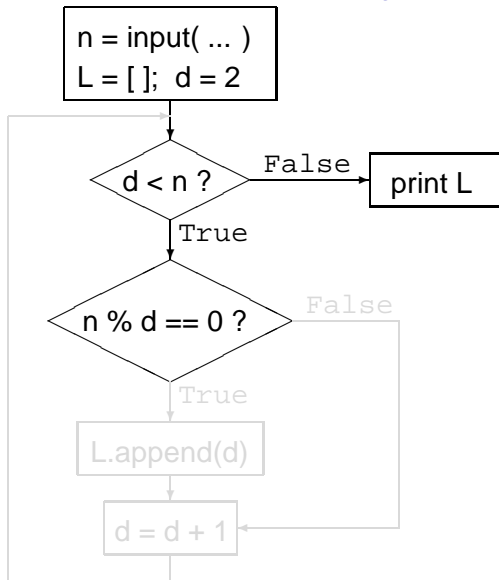
Introduction to Computing

numpy, scipy, and
Sage

Summary

Flowchart

enumerating all divisors



About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

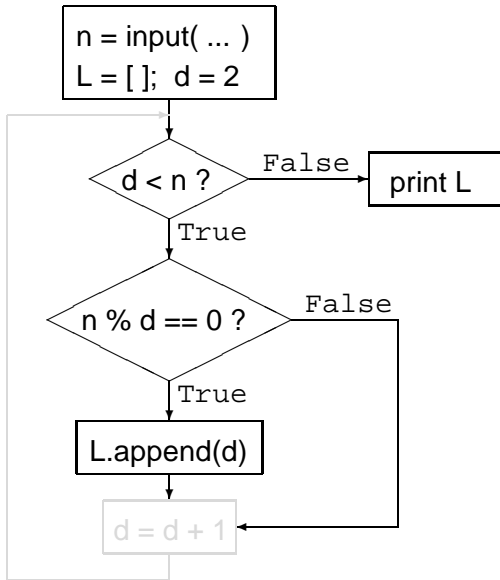
Introduction to Computing

numpy, scipy, and
Sage

Summary

Flowchart

enumerating all divisors



About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

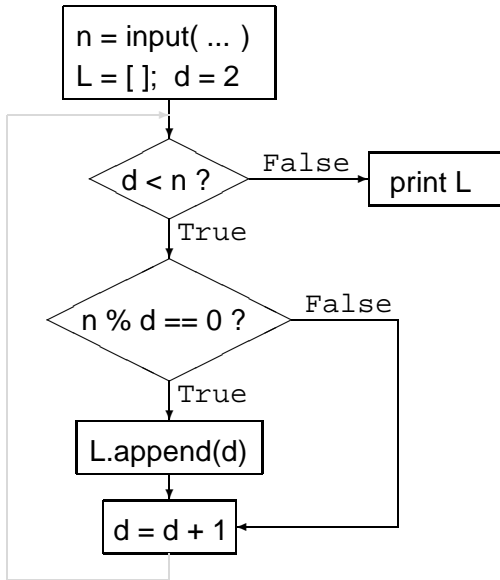
Introduction to Computing

numpy, scipy, and
Sage

Summary

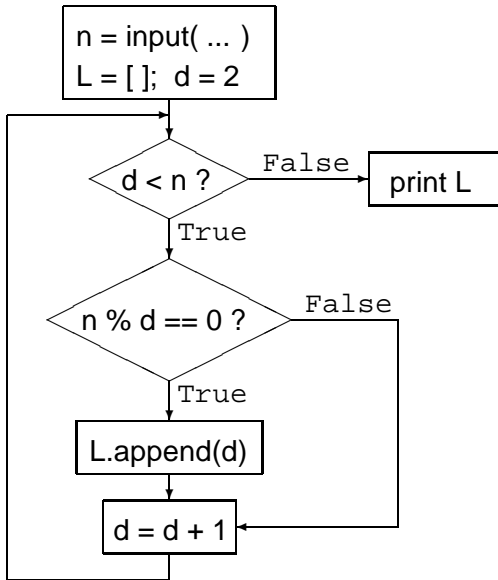
Flowchart

enumerating all divisors



Flowchart

enumerating all divisors



About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Python code

the file `enumdivs.py`

```
# MCS 275 L-1 Mon 11 Jan 2010 : enumdivs.py
# enumerate nontrivial divisors of a number

n = input('give a number : ')
L = []
for d in range(2,n):
    if n % d == 0:
        L.append(d)
print 'nontrivial divisors of %d :' % n, L
```

Running the script at the command prompt \$

```
$ python enumdivs.py
give a number : 2*3*4
nontrivial divisors of 24 : [2, 3, 4, 6, 8, 12]
$
```

Python code

the file `enumdivs.py`

```
# MCS 275 L-1 Mon 11 Jan 2010 : enumdivs.py
# enumerate nontrivial divisors of a number

n = input('give a number : ')
L = []
for d in range(2,n):
    if n % d == 0:
        L.append(d)
print 'nontrivial divisors of %d :' % n, L
```

Running the script at the command prompt \$

```
$ python enumdivs.py
give a number : 2*3*4
nontrivial divisors of 24 : [2, 3, 4, 6, 8, 12]
$
```

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

Some Observations

on `enumdivs.py`

- dynamic typing: the input `2 * 3 * 4` gets evaluated into a natural number
- `range(2, n)` is `[2, 3, ..., n-1]`
- all code in the same `for` or `if` must be preceded by same number of spaces
- three different occurrences of `%`:
 - 1 `n % d`: remainder after division by `d`
 - 2 `%d`: formatting code for decimal output
 - 3 `%`: format conversion operator in `print`
- `append` is a method for the list data type
- *long* numbers will take a *long* time

Some Observations

on `enumdivs.py`

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- dynamic typing: the input `2*3*4` gets evaluated into a natural number
- `range(2, n)` is `[2, 3, ..., n-1]`
- all code in the same `for` or `if` must be preceded by same number of spaces
- three different occurrences of `%`:
 - 1 `n % d`: remainder after division by `d`
 - 2 `%d`: formatting code for decimal output
 - 3 `%:` format conversion operator in `print`
- `append` is a method for the list data type
- *long* numbers will take a *long* time

Some Observations

on `enumdivs.py`

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- dynamic typing: the input `2*3*4` gets evaluated into a natural number
- `range(2,n)` is `[2,3,...,n-1]`
- all code in the same `for` or `if` must be preceded by same number of spaces
- three different occurrences of `%`:
 - 1 `n % d`: remainder after division by `d`
 - 2 `%d`: formatting code for decimal output
 - 3 `%:` format conversion operator in `print`
- `append` is a method for the list data type
- *long* numbers will take a *long* time

Some Observations

on `enumdivs.py`

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- dynamic typing: the input `2*3*4` gets evaluated into a natural number
- `range(2, n)` is `[2, 3, ..., n-1]`
- all code in the same `for` or `if` must be preceded by same number of spaces
- three different occurrences of `%`:
 - 1 `n % d`: remainder after division by `d`
 - 2 `%d`: formatting code for decimal output
 - 3 `%:` format conversion operator in `print`
- `append` is a method for the list data type
- *long* numbers will take a *long* time

Some Observations

on `enumdivs.py`

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- dynamic typing: the input `2 * 3 * 4` gets evaluated into a natural number
- `range(2, n)` is `[2, 3, ..., n-1]`
- all code in the same `for` or `if` must be preceded by same number of spaces
- three different occurrences of `%`:
 - 1 `n % d`: remainder after division by `d`
 - 2 `%d`: formatting code for decimal output
 - 3 `%:` format conversion operator in `print`
- `append` is a method for the list data type
- *long* numbers will take a *long* time

Some Observations

on `enumdivs.py`

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- dynamic typing: the input `2 * 3 * 4` gets evaluated into a natural number
- `range(2, n)` is `[2, 3, ..., n-1]`
- all code in the same `for` or `if` must be preceded by same number of spaces
- three different occurrences of `%`:
 - 1 `n % d`: remainder after division by `d`
 - 2 `%d`: formatting code for decimal output
 - 3 `%:` format conversion operator in `print`
- `append` is a method for the list data type
- *long* numbers will take a *long* time

Some Observations

on `enumdivs.py`

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- dynamic typing: the input `2*3*4` gets evaluated into a natural number
- `range(2, n)` is `[2, 3, ..., n-1]`
- all code in the same `for` or `if` must be preceded by same number of spaces
- three different occurrences of `%`:
 - ① `n % d`: remainder after division by `d`
 - ② `%d`: formatting code for decimal output
 - ③ `%:` format conversion operator in `print`
- `append` is a method for the list data type
- *long* numbers will take a *long* time

Some Observations

on `enumdivs.py`

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- dynamic typing: the input `2*3*4` gets evaluated into a natural number
- `range(2, n)` is `[2, 3, ..., n-1]`
- all code in the same `for` or `if` must be preceded by same number of spaces
- three different occurrences of `%`:
 - ① `n % d`: remainder after division by `d`
 - ② `%d`: formatting code for decimal output
 - ③ `%:` format conversion operator in `print`
- `append` is a method for the list data type
- *long* numbers will take a *long* time

Factorization into Primes

back to the original problem

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Problem (input/output specification):

input : a natural number n

output : a list of primes $[p_1, p_2, \dots, p_k]$,

$$n = p_1 \times p_2 \times \dots \times p_k$$

Ingredients in a *first* algorithm:

- 1 we enumerate all candidate divisors, starting at 2
- 2 when we find a divisor d of n , we continue with the quotient q , so we assign $n = q$
- 3 $(q, r) = \text{divmod}(n, d)$ computes quotient q and remainder r of the division of n by d
- 4 all divisors are appended to a list

Factorization into Primes

back to the original problem

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Problem (input/output specification):

input : a natural number n

output : a list of primes $[p_1, p_2, \dots, p_k]$,

$$n = p_1 \times p_2 \times \dots \times p_k$$

Ingredients in a *first* algorithm:

- 1 we enumerate all candidate divisors, starting at 2
- 2 when we find a divisor d of n , we continue with the quotient q , so we assign $n = q$
- 3 $(q, r) = \text{divmod}(n, d)$ computes quotient q and remainder r of the division of n by d
- 4 all divisors are appended to a list

Factorization into Primes

back to the original problem

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

Problem (input/output specification):

input : a natural number n
output : a list of primes $[p_1, p_2, \dots, p_k]$,
$$n = p_1 \times p_2 \times \dots \times p_k$$

Ingredients in a *first* algorithm:

- 1 we enumerate all candidate divisors, starting at 2
- 2 when we find a divisor d of n , we continue with the quotient q , so we assign $n = q$
- 3 $(q, r) = \text{divmod}(n, d)$ computes quotient q and remainder r of the division of n by d
- 4 all divisors are appended to a list

Factorization into Primes

back to the original problem

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

Problem (input/output specification):

input : a natural number n

output : a list of primes $[p_1, p_2, \dots, p_k]$,

$$n = p_1 \times p_2 \times \dots \times p_k$$

Ingredients in a *first* algorithm:

- 1 we enumerate all candidate divisors, starting at 2
- 2 when we find a divisor d of n , we continue with the quotient q , so we assign $n = q$
- 3 $(q, r) = \text{divmod}(n, d)$ computes quotient q and remainder r of the division of n by d
- 4 all divisors are appended to a list

Factorization into Primes

back to the original problem

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

Problem (input/output specification):

input : a natural number n

output : a list of primes $[p_1, p_2, \dots, p_k]$,

$$n = p_1 \times p_2 \times \dots \times p_k$$

Ingredients in a *first* algorithm:

- 1 we enumerate all candidate divisors, starting at 2
- 2 when we find a divisor d of n , we continue with the quotient q , so we assign $n = q$
- 3 $(q, r) = \text{divmod}(n, d)$ computes quotient q and remainder r of the division of n by d
- 4 all divisors are appended to a list

Flowchart for Factoring in Primes

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

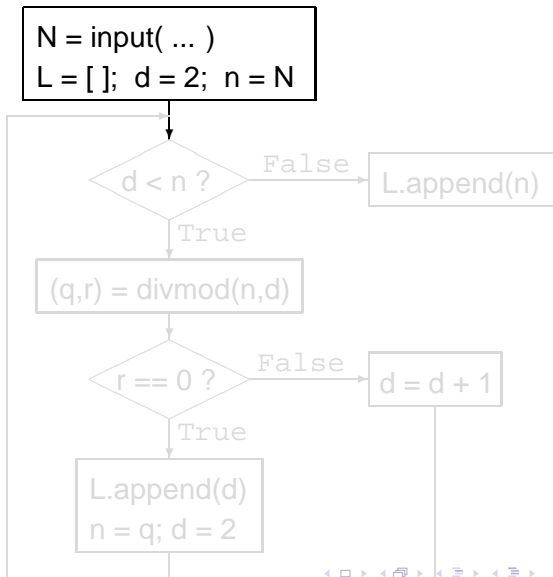
Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary



Flowchart for Factoring in Primes

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

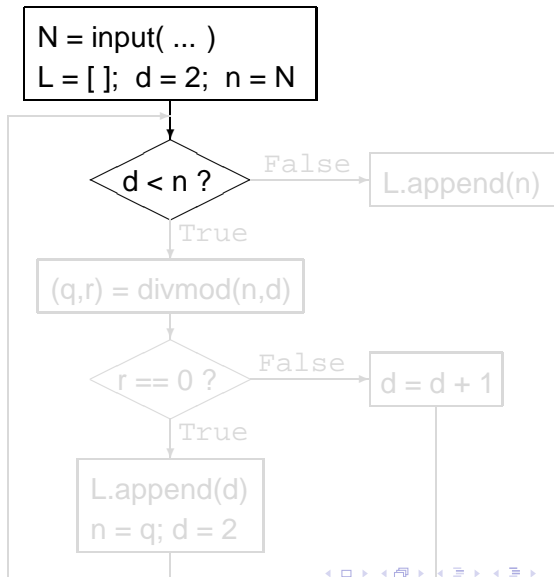
Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

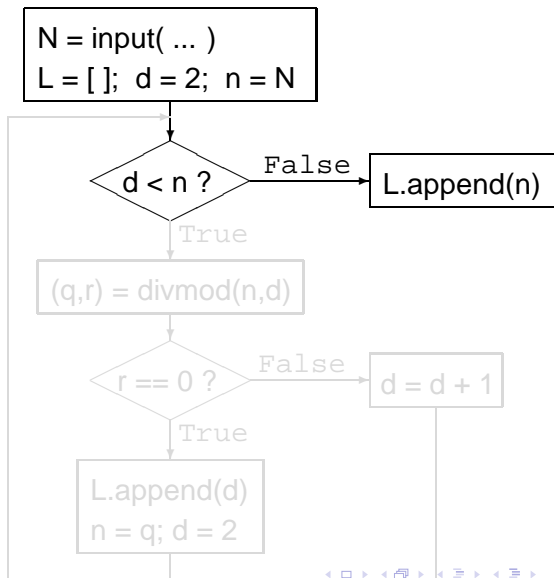
Introduction to Computing

numpy, scipy, and Sage

Summary



Flowchart for Factoring in Primes



Flowchart for Factoring in Primes

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

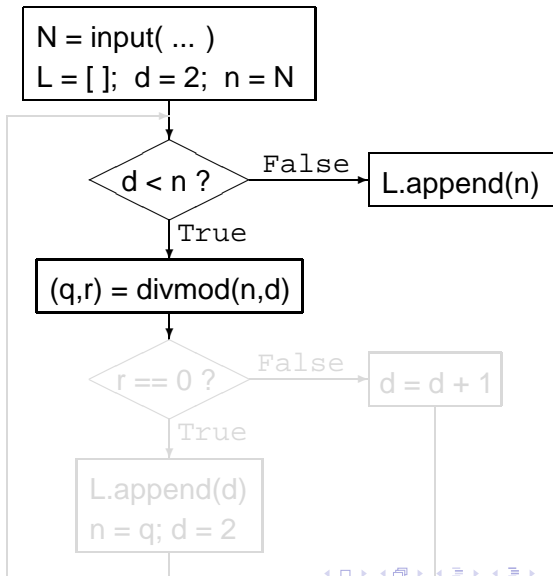
Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary



Flowchart for Factoring in Primes

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

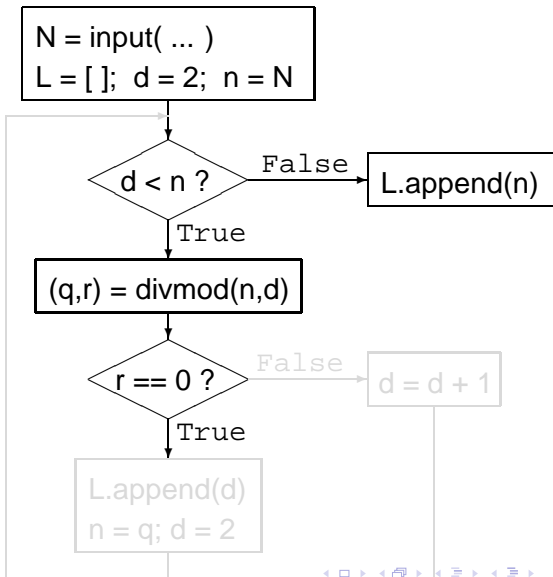
Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary



Flowchart for Factoring in Primes

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

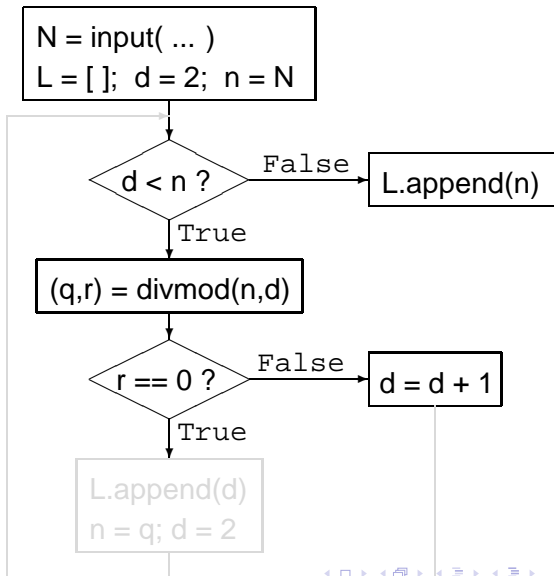
Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary



Flowchart for Factoring in Primes

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

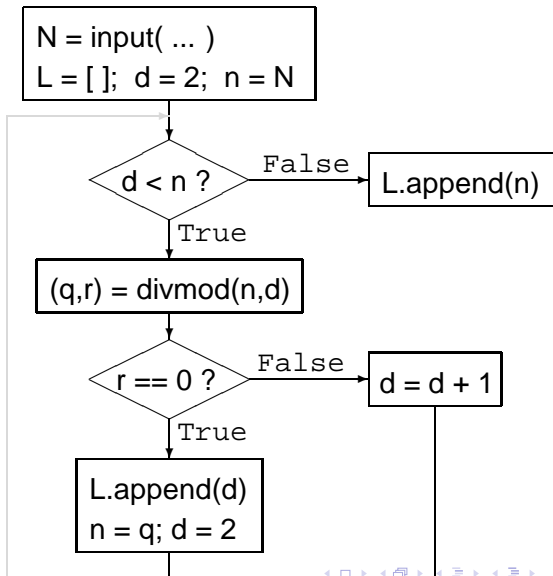
Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary



Flowchart for Factoring in Primes

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

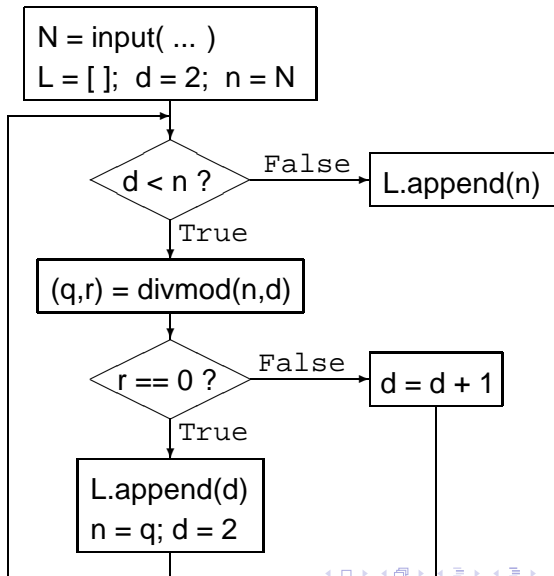
Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary



the script `facnum.py`

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

```
# MCS 275 L-1 Mon 11 Jan 2010 : facnum.py
# factor a number into product of primes
```

```
N = input('give a natural number n : ')
L = []; d = 2; n = N
while(d < n):
    (q,r) = divmod(n,d)
    if(r == 0):
        L.append(d)
        n = q; d = 2
    else:
        d = d + 1
L.append(n)
print 'factors of ' + str(N) + ' :', L
p = reduce(lambda x,y: x*y , L)
print 'product of factors :', p
```

the script `facnum.py`

```
# MCS 275 L-1 Mon 11 Jan 2010 : facnum.py
# factor a number into product of primes
```

```
N = input('give a natural number n : ')
L = []; d = 2; n = N
while(d < n):
    (q,r) = divmod(n,d)
    if(r == 0):
        L.append(d)
        n = q; d = 2
    else:
        d = d + 1
L.append(n)
print 'factors of ' + str(N) + ' :', L
p = reduce(lambda x,y: x*y , L)
print 'product of factors :', p
```

Functional Programming

short but powerful scripts

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP

example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

The statement

```
p = reduce(lambda x,y: x*y , L)
```

computes the product

$$p = \prod_{p_i \in L} p_i = p_1 \times p_2 \times \cdots \times p_k.$$

The anonymous function `lambda x,y: x*y` is applied to the list `L` **reducing** the list `L` to the product of all its elements.

Welcome to MCS 275

About the Course

Course Content
Prerequisites & Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in Primes

Introduction to Computing

numpy, scipy, and Sage

Summary

- 1 About the Course
Course Content
Prerequisites & Expectations
- 2 Introduction to Programming
Scripting in Python
from OOP to LAMP
example: Factorization in Primes
- 3 Introduction to Computing
numpy, scipy, and Sage
- 4 Summary

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

Introduction to Computing

numpy, scipy, and Sage

- **MCS = Mathematical Computer Science**
prepare for interdisciplinary scientific projects
- Python has no array type; **numpy** adds fast multidimensional array facilities to Python.
- Sage: Open Source Mathematics Software uses Python to glue many mathematical software systems.

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

Introduction to Computing

numpy, scipy, and Sage

- MCS = Mathematical Computer Science
prepare for interdisciplinary scientific projects
- Python has no array type; **numpy** adds fast
multidimensional array facilities to Python.
- Sage: Open Source Mathematics Software uses
Python to glue many mathematical software systems.

About the
Course

Course Content
Prerequisites &
Expectations

Introduction to
Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to
Computing

numpy, scipy, and
Sage

Summary

Introduction to Computing

numpy, scipy, and Sage

- MCS = Mathematical Computer Science
prepare for interdisciplinary scientific projects
- Python has no array type; **numpy** adds fast
multidimensional array facilities to Python.
- Sage: Open Source Mathematics Software uses
Python to glue many mathematical software systems.

Exercises

About the Course

Course Content
Prerequisites &
Expectations

Introduction to Programming

Scripting in Python
from OOP to LAMP
example:
Factorization in
Primes

Introduction to Computing

numpy, scipy, and
Sage

Summary

- 1 Make `enumdivs.py` more efficient by examining only \sqrt{n} candidate divisors.
- 2 Relate the number of $n\%d$ operations in `enumdivs.py` to the number of decimal places of n . If we add one more decimal place to the input n , how many more operations does `enumdivs.py` do?
- 3 Improve the efficiency of `facnum.py` also by taking fewer candidate divisors as in exercise 1 above.
- 4 Verify the correctness of `facnum.py` by an exhaustive enumeration of all numbers n within the range $1, \dots, m$, where m is given by the user.
- 5 Analyze the running time of `facnum.py` experimentally by trying various input numbers. For which numbers does `facnum.py` take the longest time? Which numbers are easiest?

About the
CourseCourse Content
Prerequisites &
ExpectationsIntroduction to
ProgrammingScripting in Python
from OOP to LAMP
example:
Factorization in
PrimesIntroduction to
Computingnumpy, scipy, and
Sage

Summary

In this lecture we covered

- 1 write algorithms from i/o specifications;
- 2 encode algorithms in Python scripts.

Lab sessions take place in SEL 2263 (Mac Lab).

Topics of the lab sessions this week:

- 1 get familiar with Python – you can bring your laptop to install Python and numpy
- 2 run scripts `enumdivs.py` and `facnum.py`
- 3 consider exercises of the lectures
- 4 go over final of MCS 260 last fall
- 5 small quiz in the last half hour