The format of this document is plain, homemade PDFLATEX (from DocOnce).

This is a document with many test constructions for doconce syntax. It was used heavily for the development and kept for testing numerous constructions, also special and less common cases.
And exactly for test purposes we have an extra line here, which is part of the abstract.

1 Section 1

`{sec1}`

Here is a nested list:

- item1
- item2
- item3 which continues on the next line to test that feature
- and a sublist
  - with indented subitem1
  - and a subitem2
- and perhaps an ordered sublist
  1. first item
  2. second item, continuing on a new line

Here is a list with paragraph heading.

- item1
- item2

Here is a list with subsubsection heading.

- item1
- item2

> Here are two lines that make up a block quote for testing *emphasized words* and **boldface words**, also with hypens: *pre*-fix, post-*fix*, **pre**-fix, post-**fix**.

Here are two references. Equation (**??**) is fine. Eq. (**??**) too. Even Equation (**??**) without the tilde.

1.1 Subsection 1

nc function {subsec1}
{[illegible]vr}
{[illegible]vr}

More text, with a reference back to Section **??** and **??**, and further to the sections **??** and **??**, which encourages you to do the tasks in the Exercises **??** and **??**. Appendices **??** and **??** are also nice elements.

{[illegible]vr}
{[illegible]vr}

Test Section reference at beginning of line and after a sentence. Section **??** is fine. Section **??** too.

Computer code. Let's do some copying from files too. First from subroutine up to the very end,

```
      subroutine test()
      integer i
      real*8 r
      r = 0
      do i = 1, i
         r = r + i
      end do
      return
C     END1

      program testme
      call test()
      return
```

and then just the subroutine,

```
      subroutine test()
      integer i
      real*8 r
      r = 0
      do i = 1, i
         r = r + i
      end do
      return
```

and finally the complete file with a plain text verbatim environment (`envir=ccq`):

```
C     a comment

      subroutine test()
      integer i
      real*8 r
      r = 0
      do i = 1, i
         r = r + i
      end do
      return
C     END1

      program testme
      call test()
      return
```

Testing other code environments. First Python:

```
!bc pycod
def f(x):
    return x+1
!ec
```

which gets rendered as

```
def f(x):
    return x+1
```

Test paragraph and subsubsection headings before before code.

Paragraph heading before code.

```
import sys
sys.path.insert(0, os.pardir)
```

Subsubsection heading before code.

```
def h(z):
    return z+1
```

Now a complete program to be shown via Python Online Tutorial:

```
class Line:
    def __init__(self, a, b):
        self.a, self.b = a, b

    def __call__(self, x):
        a, b = self.a, self.b
        return a*x + b

line = Line(2, 1)
y = line(x=3)
print(y)
```

Some more Python code (actually specified as a sage cell, but such cells are not supported by this format).

```
a = 2
b = 3
print('a+b:', a + b)

# In a sage cell we can also plot
from matplotlib.pyplot import *
from numpy import *
x = linspace(0, 4*pi, 101)
y = exp(-0.1*x)*cos(x)
plot(x, y)
xlabel('x'); ylabel('y')
show()
```

Then Cython (with -h option so it is hidden in html/sphinx):

```
cpdef f(double x):
    return x + 1
```

Standard Python shell sessions:

```
>>> from numpy import linspace, sin
>>> # Some comment
>>> x = linspace(0, 2, 11)
>>> y = sin(x)
>>> y[0]
0
>>> import matplotlib.pyplot as plt
>>> plt.plot(x, y)
```

Similar IPython sessions:

```
In [1]: from numpy import linspace, sin
In [2]: # Some comment
In [3]: x = linspace(0, 2, 11)
In [4]: y = sin(x)
In [5]: y[0]
Out[5]: 0
In [6]: import matplotlib.pyplot as plt
In [7]: plt.plot(x, y)
In [8]: a='multiple-\nline\noutput'
In [9]: a
Out[9]: 'multiple-\nline\noutput'
In [10]: print(a)
multiple-
line
output
```

Here is the interactive session again, but with `pyshell-t`.

```
>>> from numpy import linspace, sin
>>> # Some comment
>>> x = linspace(0, 2, 11)
>>> y = sin(x)
>>> y[0]
0
>>> import matplotlib.pyplot as plt
>>> plt.plot(x, y)
```

C++:

```
#include <iostream>

int main()
{
   std::cout << "Sample output" << std::endl;
   return 0
}
```

And a little bit of Fortran: latex_figs/dizzy_face.png

```
!bc cod
      subroutine midpt(x, length, a, b)
      real*8 a, b, x
      x = (a + b)/2
      length = b - a
      return
      end
!ec
```

which then is typeset as

```
      subroutine midpt(x, length, a, b)
      real*8 a, b, x
      x = (a + b)/2
      length = b - a
      return
      end
```

HTML:

```
<table>
<tr><td>Column 1</td><td>Column 2</td></tr>
<tr><td>0.67526 </td><td>0.92871 </td></tr>
<!-- comment -->
</table>
```

But inline HTML code is also important, like text that starts with `<a href="` (which can destroy the following text if not properly quoted).

Matlab with comments requires special typesetting:

```
% Comment on the beginning of the line can be escaped by %%
if a > b
  % Indented comment needs this trick
  c = a + b
end
```

And here is a system call:

```
Terminal> mkdir test
Terminal> cd test
Terminal> myprog -f
output1
output2
```

Any valid pygments lexer/language name can appear to, e.g.,

```
!bc restructuredtext
=======
Heading
=======

Some text.
!ec
```

results in

```
=======
Heading
=======

Some text.
```

Finally, `!bc do` supports highlighting of DocOnce source:

```
======= DocOnce test file =======

===== Computer code =====

Inline verbatim code, as in `import numpy as np`, is allowed, as well as
code blocks:

!bc pycod
from math import sin

def f(x):
    """Example on a function."""
    return sin(x) + 1

print(f(0))
!ec


===== Mathematics =====

Formulas can be inline, as in $\nabla\cdot\bm{u} = 0$, or typeset
as equations:

!bt
\begin{align*}
\nabla\cdot\bm{u} &= 0,\\
\bm{u} &= \nabla\phi .
\end{align*}
!et

=== Subsubsection heading ===

DocOnce files can have chapters, sections, subsections, and subsubsections.

__Paragraph heading.__ Paragraphs may have headings.
```

It is time to test `verbatim inline font` especially with `a newline inside the text` and an exclamation mark at the end: `BEGIN`! For spellcheck, test `a verbatim expression` in `another` in a `third`. Also test exclamation mark as in `!bc` and `!ec` as well as `a != b`. Also test backslashes and braces like `\begin`, `\begin{enumerate}`, `\end{this}\end{that}`, and `{something \inside braces}`.

Here is some red color and an attempt to write with green color containing a linebreak
code. Some formats will only display
this correctly when `html`
is the output format. But here some more running text is added which is not part of the previous blocks with line breaks.

Running OS commands.

```
Terminal> python -c 'print("Testing\noutput\nfrom\nPython.")'
Testing
output
from
Python.
```

Footnotes. Here is a test of footnotes [1], which are handy in text. They are used in different flavors, now in

- list items (note below that footnotes work after math, verbatim, and URLs - bin fact old and emphasize too!)
- even with math ∇^2u^2
- and code `h[i] += 1`[3] (*must* have space between inline code and footnote!)
- and links[4][5]

which gives flexibility in writing. This is the third[7] example.

Here is some more text before a new definition of a footnote that was used above.

Non–breaking space character.

This paragraph aims to test non–breaking space character[a], and a typical example where this is needed is in physical units: 7.4 km is traveled in $7.4/5.5 \approx 1.345$ s. Also check that a link[b] is not broken across lines (drag the browser window to test this). (On the other hand, the tilde is used in computer code, e.g., as in `[ x for x in y]` or in `y= x`, and should of course remain a tilde in those contexts.)

[a] `https://en.wikipedia.org/wiki/Non-breaking_space`
[b] `https://google.com`

1.2 Subsection 2: Testing figures

Test of figures. In particular we refer to Figure ?? in which there is a flow. {subsec:ex}
{11@xr}

Figures without captions are allowed and will be inlined.

[1] Typesetting of the footnote depends on the format. Plain text does nothing, LaTeX removes the definition and inserts the footnote as part of the LaTeX text. reStructuredText and Sphinx employ a similar type of typesetting as Extended Markdown and DocOnce, and in HTML we keep the same syntax, just displayed properly in HTML.

[2] Math footnotes can be dangerous since it interferes with an exponent.

[3] One–line footnote.

[4] `https://google.com`

[5] `google.com`[6] is perhaps the most famous web site today.

[7] Not much to add here, but the footnote is at the end with only one newline.

`testfigs/wave1D.png`

Figure 1: Visualization **of** a *wave.* `fig:impact`

`testfigs/wave1D.png`

{SC@1}

Figure 2: A long caption spanning several lines and containing verbatim words like `my_file_v1` and `my_file_v2` as well as as math with subscript as in t_{i+1}.

{myfig}

movies
{12@xr}}

Here is figure **??** with a long (illegal) multi–line caption containing inline verbatim text:
Test URL as figure name:

Remark. Movies are tested in separate file `movies.do.txt`.

1.3 The θ parameter (not ∇?)

{decay:sec:theta}

Functions do not always need to be advanced, here is one involving θ:

```
def f(theta):
    return theta**2
```

More on θ. Here is more text following headline with math.

Newcommands must also be tested in this test report: $\frac{1}{2}$, $1/2$, $\boldsymbol{x}$, $\frac{Du}{dt}$, both inline and in block:

$$\begin{aligned}
\frac{Du}{dt} &= 0 \\
\frac{1}{2} &= 1/2 \qquad (1) \\
\frac{1}{2}\boldsymbol{x} &= \boldsymbol{n} \qquad (2)
\end{aligned}$$

Or with align with label and numbers:

{aligneq1}

$$\frac{Du}{dt} = 0 \tag{3}$$

$$\frac{1}{2} = 1/2 \tag{4}$$

{aligneq2}

$$\frac{1}{2}\boldsymbol{x} = \boldsymbol{n} \tag{5}$$

First one numbered (automatically):

$$\begin{pmatrix} G_2+G_3 & -G_3 & -G_2 & 0 \\ -G_3 & G_3+G_4 & 0 & -G_4 \\ -G_2 & 0 & G_1+G_2 & 0 \\ 0 & -G_4 & 0 & G_4 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} + \cdots \tag{6}$$

$$\begin{pmatrix} C_5+C_6 & -C_6 & 0 & 0 \\ -C_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ -i_0 \end{pmatrix}$$

Second numbered (automatically):

$$\begin{pmatrix} G_1+G_2 & \\ -G_3 & G_4 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} + \cdots$$

$$\begin{pmatrix} y & 2 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ x \end{pmatrix} = \begin{pmatrix} A \\ B \end{pmatrix} \tag{7}$$

Both numbered, with label by the user:

$$\begin{pmatrix} G_1+G_2 & \\ -G_3 & G_4 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} + \cdots \tag{8}$$

{mymatrix:eq1}

$$\begin{pmatrix} y & 2 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ x \end{pmatrix} = \begin{pmatrix} A \\ B \end{pmatrix} \tag{9}$$

{mymatrix:eq2}

Now we refer to (**??**)-(**??**).

Table 1: Testing table environment in LaTeX, enabled by testing on the "latex" format with the preprocessor. mytab

time	velocity	acceleration
0.0	1.4186	-5.01
2.0	1.376512	11.919
4.0	1.1E+1	14.717624

1.4 Custom Environments

Here is an attempt to create a theorem environment via Mako (for counting theorems) and comment lines to help replacing lines in the `.tex` by proper begin–end LaTeX environments for theorems. Should look nice in most formats! {theorem:fund

Theorem 5. Let $a = 1$ and $b = 2$. Then $c = 3$.

Proof. Since $c = a + b$, the result follows from straightforward addition. $\Diamond$
As we see, the proof of Theorem 5 is a modest achievement.

1.5 Tables

dex with{subsec:table}
is possible
`two` (separate) `verbatim`
`sions` which is also possi-

with!subindex
ace word in index
with **boldface word**
with!**boldface word** in
ry
boldface word! **bold-**
word in subentry too
with comma, and one

Let us take this table from the manual:

The DocOnce source code reads

```
|--------------------------------|
|time  | velocity | acceleration |
|--l--------r-----------r--------|
| 0.0  | 1.4186   | -5.01        |
| 2.0  | 1.376512 | 11.919       |
| 4.0  | 1.1E+1   | 14.717624    |
|--------------------------------|
```

Here is yet another table to test that we can handle more than one table:

time	velocity	acceleration
0.0	1.4186	-5.01
1.0	1.376512	11.919
3.0	1.1E+1	14.717624

And one with math headings (that are expanded and must be treated accordingly), verbatim heading and entry, and no space around the pipe symbol:

i	h_i	$\bar{T}_i$	`L_i`
0	0	288	-0.0065
1	11,000	216	0.0
2	20,000	216	0.001
3	32,000	228	0.0028
4	47,000	270	0.0
5	51,000	270	-0.0028
6	71,000	214	`NaN`

And add one with verbatim headings (with underscores), and rows starting with `|-` because of a negative number, and `|` right before and after verbatim word (with no space):

exact	`v_1`	a_i + `v_2`	`verb_3_`
9	9.62	5.57	8.98
-20	-23.39	-7.65	-19.93
10	17.74	-4.50	9.96
0	-9.19	4.13	-0.26

Pipe symbols in verbatim and math text in tables used to pose difficulties, but not anymore:

S	command
$\|\|a_0\|\|$	`norm|length`
$x\cap y$	`x|y`

Here is a table with X alignment:

Type	Description
X	Alignment character that is used for specifying a potentially very long text in a column in a table. It makes use of the `tabularx` package in LaTeX, otherwise (for other formats) it means `l` (centered alignment).
l,r,c	standard alignment characters

Finally, a table with math and URLs.

$\mathcal{L}=0$	../doc/src/manual/mov/wave_frames/frame_0080.png	../doc/src/manual/mov/wave_frames/frame_0085.png
$a=b$	../doc/src/manual/mov/wave_frames/frame_0090.png	../doc/src/manual/mov/wave_frames/frame_0095.png
$\nabla\cdot\boldsymbol{u} =0$	../doc/src/manual/mov/wave_frames/frame_0100.png	../doc/src/manual/mov/wave_frames/frame_0105.png

1.6 A test of verbatim words in heading with subscript a_i: `my_file_v1` and `my_file_v2`

Paragraph with verbatim and math: `my_file_v1.py` and `my_file_v2.py` define some math a_{i-1}. Here is more `__verbatim__` code and some plain text on a new line.

1.7 Just bold

Some text.

1.8 *Just emphasize*

Some text.

1.9 `Just verbatim`

Some text.

1.10 Bold beginning

Some text.

1.11 *Emphasize* beginning

Some text.

1.12 `Verbatim` beginning

Some text.

1.13 Maybe bold end

Some text.

1.14 Maybe *emphasize end*

Some text.

1.15 Maybe `verbatim end`

Some text.

1.16 The middle has bold word

Some text.

1.17 The middle has *emphasize* word

Some text.

1.18 The middle has `verbatim` word

Some text.

Just emphasize. Some text.

`Just verbatim.` Some text.

***Emphasize* beginning.** Some text.

`Verbatim beginning.` Some text.

Maybe *emphasize end.* Some text.

Maybe `verbatim end.` Some text.

The middle has *emphasize* word. Some text.

The middle has `verbatim` word. Some text.

Ampersand. We can test Hennes & Mauritz, often abbreviated H&M, but written as `Hennes & Mauritz` and `H & M`. A sole `&` must also work.

```
# Just to check that ampersand works in code blocks:
c = a & b
```

Quotes. Let us also add a test of quotes such as “double quotes, with numbers like 3.14 and newline/comma and hyphen (as in double–quote)”; written in the standard LaTeX–style that gives correct LaTeX formatting and ordinary double quotes for all non–LaTeX formats. Here is another sentence that “caused” a bug in the past because double backtick quotes could imply verbatim text up to a verbatim word starting with period, like `.txt`.

More quotes to be tested for spellcheck: (“with parenthesis”), “with newline” and “with comma”, “hyphen”-wise, and “period”.

1.19 Bibliography test

[illegible]

{Langtangen_1994a}

{Mardal_et_al_2003a}

{Langtangen_1988d}

2,Mardal_et_al_2003a}

[illegible]

[illegible]

[illegible]

Here is an example: [?] discussed propagation of large destructive water waves, [?] gave an overview of numerical methods for solving the Navier—Stokes equations, while the use of Backward Kolmogorov equations for analyzing random vibrations was investigated in [?]. The book chapter [?] contains information on C++ software tools for programming multigrid methods. A real retro reference is [?] about a big FORTRAN package. Multiple references are also possible, e.g., see [?, ?].

We need to cite more than 10 papers to reproduce an old formatting problem with blanks in the keys in reST format: [?, ?, ?, ?] and [?, ?, ?, ?, ?, ?, ?] and all the work of [?, ?, ?] as well as old work [?] and [?], and the talk [?]. Langtangen also had two thesis [?, ?] back in the days. More retro citations are the old ME-IN323 book [?] and the [?] OONSKI ’94 paper.

Example 1: Examples can be typeset as exercises

{Example}

Examples can start with a subsection heading starting with `Example:` and then, with the command–line option `--examples_as_exercises` be typeset as exercises. This is useful if one has solution environments as part of the example.

a) State some problem.

Solution. The answer to this subproblem can be written here.

b) State some other problem.

Hint 1. A hint can be given.

Hint 2. Maybe even another hint?

Solution. The answer to this other subproblem goes here, maybe over multiple doconce input lines.

1.20 User–defined environments

Example **??** demonstrates how to write a test function. That is, a special test function for a function `add` appears in Example **??**. {13@xrr} {14@xrr} {ex:test:1p1}

Example 1.1. *A test function.*

Suppose we want to write a test function for checking the implementation of a Python function for addition.

```
def add(a, b):
    return a + b

def test_add():
    a = 1; b = 1
    expected = a + b
    computed = add(a, b)
    assert expected == computed
```

{ex:math:1p1}

Example 1.2. *Addition.*

We have

$$1 + 1 = 2$$

or in tabular form:

Problem	Result
$1+1$	2

Highlight box!

This environment is used to highlight something:

$$E = mc^2$$

1.21 URLs

{subsubsec:ex}

Testing of URLs: hpl's home page hpl[8], or the entire URL if desired, `https://folk.uio.no/hpl`. Here is a plain file link `testdoc.do.txt`[9], or `testdoc.do.txt`[10], or `testdoc.do.txt`[11] or

[8]`https://folk.uio.no/hpl`

[9]`testdoc.do.txt`

[10]`testdoc.do.txt`

[11]`testdoc.do.txt`

`testdoc.do.txt`[12] or a link with newline[13]. Can test spaces with the link with word too: hpl[14] or hpl[15]. Also `file:///` works: link to a file[16] is fine to have. Moreover, "loose" URLs work, i.e., no quotes, just the plain URL as in `https://folk.uio.no/hpl`, if followed by space, comma, colon, semi–colon, question mark, exclamation mark, but not a period (which gets confused with the periods inside the URL).

Mail addresses can also be used: `hpl@simula.no`[17], or just a mail link[18], or a raw `mailto:hpl@simula.no`[19].

Here are some tough tests of URLs, especially for the `latex` format: Newton–Cotes[20] formulas and a good book[21]. Need to test Newton–Cotes with percentage in URL too: `https://en.wikipedia.org/wiki/Newton%E2%80%93Cotes_formulas` and `https://en.wikipedia.org/wiki/Newton--Cotes#Open_Newton.E2.80.93Cotes_formulae` which has a shebang.

For the `-device=paper` option it is important to test that URLs with monospace font link text get a footnote (unless the `--latex_no_program_footnotelink` is used), as in this reference to `decay_mod`, `ball1.py`, and `ball2.py`.

More tough tests: repeated URLs whose footnotes when using the `-device=paper` option must be correct. We have google[22], google[23], and google[24], which should result in exactly three footnotes.

1.22 Test of Some LaTeX Fixes

Let's check abbr. of some common kind, e.g. the well–known i.e. expression as an example, and 1 vs. 2 which is also often used. Dr. Tang and Prof. Monsen, or maybe also prof. Ting, will go to the Dept. of Science to test how Mr. Hansen is doing together with Ms. Larsen. A reference like
{{600@vr} Sec. ?? or Ch. ??, or even App. ??, must also be handled. Likewise, this is test no. *i* of DocOnce
{{800@vr} features. Also, look at Fig. 4 to see how the data compares with Tab. ??. Percentage must be fixed: 7%, 87.65% and 50% at the beginning of the line.

[12] `testdoc.do.txt`
[13] `testdoc.do.txt`
[14] `https://folk.uio.no/hpl`
[15] `https://folk.uio.no/hpl`
[16] `file:///home/hpl/vc/doconce/doc/demos/manual/manual.html`
[17] `mailto:hpl@simula.no`
[18] `mailto:hpl@simula.no`
[19] `mailto:hpl@simula.no`
[20] `https://en.wikipedia.org/wiki/Newton%E2%80%93Cotes_formulas`
[21] `https://www.springer.com/mathematics/computational+science+%26+engineering/book/978-3-642-23098-1`
[22] `https://google.com`
[23] `https://google.com`
[24] `https://google.com`

2 LaTeX Mathematics

Here is an equation without label using backslash–bracket environment:

$$a = b + c$$

or with number and label, as in (**??**), using the equation environment:

$$\frac{\partial u}{\partial t} = \nabla^2 u \qquad (10)$$ {my:eq1}

We can refer to this equation by (**??**).

Here is a system without equation numbers, using the align–asterisk environment:

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{q} \times \boldsymbol{n} \\ b &= \nabla^2 u + \nabla^4 v \end{aligned}$$

And here is a system of equations with labels in an align environment:

$$a = q + 4 + 5 + 6 \qquad (11)$$ {eq1}

$$b = \nabla^2 u + \nabla^4 x \qquad (12)$$ {eq2}

We can refer to (**??**)-(**??**). They are a bit simpler than the Navier—Stokes equations. And test LaTeX hyphen in `CG-2`. Also test a_{i-j} as well as $kx - wt$.

Testing `alignat` environment:

$$a = q + 4 + 5 + 6 \qquad \text{for } q \geq 0 \qquad (13)$$ {eq1a}

$$b = \nabla^2 u + \nabla^4 x \qquad x \in \Omega \qquad (14)$$ {eq2a}

Many of the next environments will fail in non–latex formats. Testing multiline:

$$\begin{multline} a = b = q + \\ f + \nabla \cdot \nabla u \qquad (15) \end{multline}$$ {multiline:eq1}

Testing split:

$$
\begin{split}
a = b = q + \\
f + \nabla \cdot \nabla u
\end{split} \tag{16}
$$
{split:envir:

We can refer to the last equation by (**??**).

Testing gather:

$$a = b \tag{17}$$
$$c = d + 7 + 9 \tag{18}$$

Let us refer to (**??**)-(**??**) again, and to the alignat variant (**??**)-(**??**), and to (**??**).

Testing eqnarray:

{myeq1}
$$\frac{\partial u}{\partial t} = \nabla^2 u + f, \tag{19}$$
{myeq2}
$$\frac{\partial v}{\partial t} = \nabla \cdot (q(u)\nabla v) + g \tag{20}$$

More mathematical typesetting is demonstrated in the coming exercises.

[illegible] Below, we have Problem **??** and Project **??**, as well as Projects **??** and **??**, and in between [illegible] there we have Exercise **??**.

3 Exercises

Problem 2: Flip a Coin

{demo:ex:1}

a) Make a program that simulates flipping a coin N times. Print out "tail" or "head" for each flip and let the program count the number of heads.

Hint 1. Use `r = random.random()` and define head as `r <= 0.5`.

Hint 2. Draw an integer among $\{1, 2\}$ with `r = random.randint(1,2)` and define head when `r` is 1.

Answer. If the `random.random()` function returns a number $< 1/2$, let it be head, otherwise tail. Repeat this N number of times.

Solution.

```
import sys, random
N = int(sys.argv[1])
heads = 0
for i in range(N):
    r = random.random()
    if r <= 0.5:
        heads += 1
print('Flipping a coin %d times gave %d heads' % (N, heads))
```

b) Vectorize the code in a) using boolean indexing.

Vectorized code can be written in many ways. Sometimes the code is less intuitive, sometimes not. At least there is not much to find in Section ??. {24@xr}}

c) Vectorize the code in a) using `numpy.sum`.

Answer. `np.sum(np.where(r <= 0.5, 1, 0))` or `np.sum(r <= 0.5)`.
In this latter subexercise, we have an example where the code is easy to read.

My remarks. Remarks with such a subsubsection is treated as more text after the last subexercise. Test a list too:

1. Mark 1.
2. Mark 2.

Filenames: `flip_coin.py`, `flip_coin.pdf`.

Remarks. These are the exercise remarks, appearing at the very end.

3.1 Not an exercise

Should be possible to stick a normal section in the middle of many exercises.

Exercise 3: Test of plain text exercise

{my:exer1}

Very short exercise. What is the capital of Norway? Filename: `myexer1`.

Project 4: Compute a Probability

{demo:ex:2}

What is the probability of getting a number between 0.5 and 0.6 when drawing uniformly distributed random numbers from the interval $[0, 1)$?

At the end we have a list because that caused problems in LaTeX in previous DocOnce versions:

1. item1
2. item2

Hint. To answer this question empirically, let a program draw N such random numbers using Python's standard `random` module, count how many of them, M, that fall in the interval $(0.5, 0.6)$, and compute the probability as M/N.

Project 5: Explore Distributions of Random Circles

{proj:circle1

The formula for a circle is given by

{circle:x}
{circle:y}

$$x = x_0 + R\cos 2\pi t, \tag{21}$$

$$y = y_0 + R\sin 2\pi t, \tag{22}$$

where R is the radius of the circle, (x_0, y_0) is the center point, and t is a parameter in the unit interval $[0, 1]$. For any t, (x, y) computed from (**??**)-(**??**) is a point on the circle. The formula can be used to generate `n` points on a circle:

```
import numpy as np

def circle(R, x0, y0, n=501):
    t = np.linspace(0, 1, n)
    x = x0 + R*np.cos(2*np.pi*t)
    y = y0 + R*np.sin(2*np.pi*t)
    return x, y

x, y = circle(2.0, 0, 0)
```

The goal of this project is to draw N circles with random center and radius. Plot each circle using the `circle` function above.

a) Let R be normally distributed and (x_0, y_0) uniformly distributed.

Hint. Use the `numpy.random` module to draw the x_0, y_0, and R quantities.

Answer. Here goes the short answer to part a).

Solution. Here goes a full solution to part a).

b) Let R be uniformly distributed and (x_0, y_0) normally distributed. Filename: `norm`.

c) Let R and (x_0, y_0) be normally distributed.
Filename: `circles`.

Remarks. At the very end of the exercise it may be appropriate to summarize and give some perspectives.

Exercise 6: Determine some Distance

{exer:dist}

Intro to this exercise. Questions are in subexercises below.

Solution. Here goes a full solution of the whole exercise. With some math $a = b$ in this solution:

$$\text{math in solution: } a = b$$

And code `a=b` in this solution:

```
a = b  # code in solution
```

End of solution is here.

a) Subexercises are numbered a), b), etc.

Hint 1. First hint to subexercise a). With math $a = b$ in hint:

$$a = b.$$

And with code (in plain verbatim) returning $x + 1$ in hint:

```
def func(x):
    return x + 1  # with code in hint
```

Hint 2. Second hint to subexercise a).
Test list in hint:

1. item1
2. item2

Filename: `subexer_a.pdf`.

Answer. Short answer to subexercise a). With math in answer: $a = b$.

b) Here goes the text for subexercise b).
Some math $\cos^2 x + \sin^2 x = 1$ written one a single line:

$$\cos^2 x + \sin^2 x = 1\,.$$

Hint. A hint for this subexercise.
Filename: `subexer_b.pdf`.

Solution. Here goes the solution of this subexercise.

The text here belongs to the main (intro) part of the exercise. Need closing remarks to have text after subexercises.

Test list in exercise:

1. item1
2. item2

Remarks. Some final closing remarks, e.g., summarizing the main findings and their implications in other problems can be made. These remarks will appear at the end of the typeset exercise.

Some exercise without the "Exercise:" prefix

Just some text. And some math saying that $e^0 = 1$ on a single line, to test that math block insertion is correct:

$$\exp{(0)} = 1$$

And a test that the code `lambda x: x+2` is correctly placed here:

```
lambda x: x+2
```

Exercise 7: Solution of differential equation

{se{quhizsdiferede}}

SOlution of differential equation

Given

$$\frac{dy}{dx} = -y(x),\quad y(0)=1$$

What is the solution of this equation?

A. $y=e^{-y}$

B. $y=e^y$

C.

```
from math import exp
def f(x):
    return exp(x)
```

D. The solution cannot be found because there is a derivative in the equation.

E. The equation is meaningless: an equation must be an equation for x or y, not a function $y(x)$.

Answer: A.

Solution:

A: Right.

B: Wrong. Almost, but the sign is wrong (note the minus!).

C: Wrong. Ooops, forgot a minus: `exp(-x)`, otherwise this Python code must be considered as a good answer. It is more natural, though, to write the solution to the problem in mathematical notation:

$$y(x) = e^{-y}.$$

D: Wrong. Equations with derivatives can be solved; they are termed *differential equations*.

E: Wrong. Equations where the unknown is a function, as $y(x)$ here, are called *differential equations*, and are solved by special techniques.

Example 8: Just an example

a) What is the capital of Norway?

Answer. Oslo.

4 Here goes another section

With some text, before we continue with exercises.

5 More Exercises

Exercise 9: Make references to projects and problems

{exer:some:formula}
{26@xr}

Pick a statement from Project ?? or Problem ?? and verify it.

Test list at the end of an exercise without other elements (like subexercise, hint, etc.):

1. item1
2. item2

Filename: `verify_formula.py`.

Project 10: References to Project ?? in a heading works for pdflatex

{27@xr}
{exer:you}
{30@xr}
{31@xr}

Refer to the previous exercise as Exercise ??, the two before that as Projects ?? and ??, and this one as Project ??. Filename: `selc_composed.pdf`.

A Just for testing; part I

{app1}

This is the first appendix.

A.1 A subsection within an appendix

Some text.

B Just for testing; part II

{app2}

This is more stuff for an appendix.

B.1 Appendix: Testing identical titles

Without label.

B.2 Appendix: Testing identical titles

{test:title:id1}

With label.

B.3 Appendix: Testing identical titles

{test:title:id2}
{quiz:2}

What about inserting a quiz?

Capital of Norway

Fundamental test: What is the capital of Norway?

A. Stockholm
B. London
C. Oslo
D. Bergen

Answer: C.

Solution:

A: Wrong. Stockholm is the capital of Sweden.
B: Wrong.
C: Right.
D: Wrong. Those from Bergen would claim so, but nobody else.

B.4 Appendix: Testing identical titles

Without label.

Tip.

Here is a tip or hint box, typeset as a notice box.

Need a lot of text to surround the summary box. Version control systems allow you to record the history of files and share files among several computers and collaborators in a professional way. File changes on one computer are updated or merged with changes on another computer. Especially when working with programs or technical reports it is essential to have changes documented and to ensure that every computer and person involved in the project have the latest updates of the files. Greg Wilson' excellent Script for Introduction to Version Control[25] provides a more detailed motivation why you will benefit greatly from using version control systems.

Summary.

Bold remark: Make some text with this summary. Much testing in this document, otherwise stupid content. Much testing in this document, otherwise stupid content. Much testing in this document, otherwise stupid content. Much testing in this document, otherwise stupid content. Much testing in this document, otherwise stupid content. Much testing in this document, otherwise stupid content. Much testing in this document, otherwise stupid content. Much testing in this document, otherwise stupid content. Much testing in this document, otherwise stupid content.

Projects that you want to share among several computers or project workers are today most conveniently stored at some web site "in the cloud" and updated through communication with that site. I strongly recommend you to use such sites for all serious programming and scientific writing work - and all other important files.

The simplest services for hosting project files are Dropbox[26] and Google Drive[27]. It is very easy to get started with these systems, and they allow you to share files among laptops and mobile units with as many users as you want. The systems offer a kind of version control in that the files are stored frequently (several times per minute), and you can go back to previous versions for the last 30 days. However, it is challenging to find the right version from the past when there are so many of them.

More seriously, when several people may edit files simultaneously, it can be difficult detect who did what when, roll back to previous versions, and to manually merge the edits when these are incompatible. Then one needs more sophisticated tools than Dropbox or Google Drive: project hosting services with true version control systems. The following text aims at providing you with the minimum information to started with such systems. Numerous other tutorials contain more comprehensive material and in–depth explanations of the concepts and tools.

The idea with project hosting services is that you have the files associated with a project in the cloud. Many people may share these files. Every time you want to work on the project you explicitly update your version of the files, edit the files as you like, and synchronize the files with

[25] `https://software-carpentry.org/2010/07/script-for-introduction-to-version-control/`

[26] `https://dropbox.com`

[27] `https://drive.google.com`

the "master version" at the site where the project is hosted. If you at some point need to go back to a version of the files at some particular point in the past, this is an easy operation. You can also use tools to see what various people have done with the files in the various versions.

All these services are very similar. Below we describe how you get started with Bitbucket, GitHub, and Googlecode. Launchpad works very similarly to the latter three. All the project hosting services have excellent introductions available at their web sites, but the recipes below are much shorter and aim at getting you started as quickly as possible by concentrating on the most important need-to–know steps. The Git tutorials we refer to later in this document contain more detailed information and constitute of course very valuable readings when you use version control systems every day. The point now is to get started.

B.5 Appendix: Testing inline comments

Projects that you want to share among several computers or project workers are today most conveniently stored at some web site "in the cloud" and updated through communication with that site. I strongly recommend you to use such sites for all serious programming and scientific writing work - and all other important files.

hpl's semi opinion
not sure if in the clou
understood by all.

The simplest services for hosting project files is Dropbox.

mp 2: Simply go to `https://dropbox.com` and watch the video. It explains how files, like `myfile.py`, perhaps containing much math, like $\partial u/\partial t$, are easily communicated between machines.

It is very easy to get started with Dropbox, and it allows you to share files among (hpl 3:) ~~laptops and mobile units~~ computers, tablets, and phones.

———

First, (**edit 4**: add comma) consider a quantity Q. (edit 5:) ~~To this end,~~ We note that $Q > 0$, because (**edit 6**:) ~~a~~ negative (edit 7:) ~~quantity is~~ quantities are (**edit 8**:) ~~just~~ negative. (**edit 9**:) This comes as no surprise.

Let us refer to Figure **??** again. {32@xr}}

Test references in a list:

- ?? {33@xr}}
- ?? {34@xr}}
- ?? {35@xr}}

B.6 Appendix: Testing headings ending with `verbatim inline`

The point here is to test 1) `verbatim` code in headings, and 2) ending a heading with verbatim code as this triggers a special case in LaTeX.

We also test mdash—used as alternative to hyphen without spaces around, or in quotes:

Fun is fun.—Unknown.

The ndash should also be tested – as in the Hanson—Nilson equations on page 277–278.

And finally, what about admons, quotes, and boxes? They are tested in a separate document: `admon.do.txt`.