

PythonT_EX

Damian Łączak

Edycja i prezentacja tekstów naukowych

Python: 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]

Część I

Basics of PythonT_EX

- 1 Python basics
 - First example
 - Functions

- 1 Python basics
 - First example
 - Functions
- 2 Compilation process
 - Outline

- ① Python basics
 - First example
 - Functions
- ② Compilation process
 - Outline
- ③ Python \TeX
 - Sessions
 - Commands
 - Other commands/functions
 - Beamer compatibility
 - Other languages

First example

No brackets only indentation



```
1 # And this is a comment.  
2 from random import randint  
3 number = randint(0, 9)  
4 if number < 5:  
5     print("0-4")  
6 else:  
7     print("5-9")
```

Example output

0-4

Functions

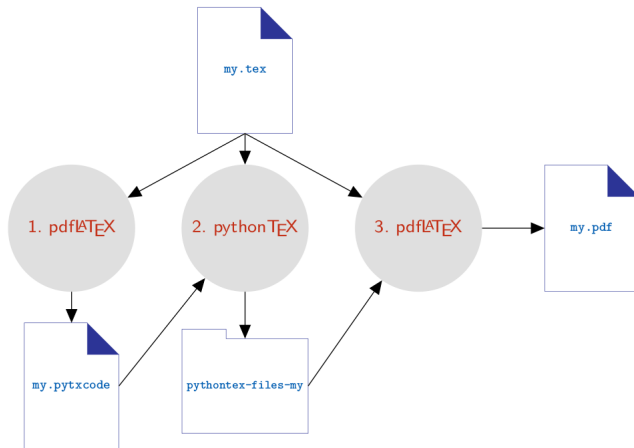


```
1 def sayMyName(name):  
2     print ("Your name is {0}".format(name))  
3 sayMyName("Damian")
```

Output

Your name is Damian

PDF creation process



Sessions

What are they for?



- Parallel execution

Sessions

What are they for?



- Parallel execution
 - Increase speed

Sessions

What are they for?



- Parallel execution
 - Increase speed
 - Different settings

Sessions

What are they for?



- Parallel execution
 - Increase speed
 - Different settings
- Default session

Sessions

What are they for?



- Parallel execution
 - Increase speed
 - Different settings
- Default session
- Session name

Sessions

What are they for?



- Parallel execution
 - Increase speed
 - Different settings
- Default session
- Session name
 - a-z

Sessions

What are they for?



- Parallel execution
 - Increase speed
 - Different settings
- Default session
- Session name
 - a-z
 - A-Z

Sessions

What are they for?



- Parallel execution
 - Increase speed
 - Different settings
- Default session
- Session name
 - a-z
 - A-Z
 - 0-9

Sessions

What are they for?



- Parallel execution
 - Increase speed
 - Different settings
- Default session
- Session name
 - a-z
 - A-Z
 - 0-9
 - hyphen and underscore

Commands

Overall look on them



Inline commands

- `py`
- `pyc`
- `pys`
- `pyv`
- `pyb`

Multi-line commands

- `pycode`
- `pysub`
- `pyverbatim`
- `pyblock`

Console commands

- `pyconsole`
- `pycon`

py

inline command



Usage

Returns text representation of it's argument.

```
1 \py{"Hello world"}  
2
```

output

Hello world

pyc

inline command



Usage

Prints evaluated expressions that are inside curly braces preceded by exclamation mark.

```
1 \pyc{a = 2**8}  
2 \py{a}  
3
```

output

256

pys

inline command



Usage

Evaluates and then substitute expressions that are surrounded by curly braces proceeded by exclamation mark by their string representation.

```
1 \pys{$1 + 1 = !{1+1}$}  
2
```

output

1 + 1 = 2

pyv

inline command



Usage

It typesets but do not execute the code.

```
1 \pyc{a = 1}  
2 \pyv{a = 256} \\  
3 \py{a}  
4
```

output

```
a = 256  
1
```

pyb

inline command



Usage

It typesets and executes the code.

```
1 \pyc{a = 1}  
2 \pyb{a = 256} \\  
3 \py{a}  
4
```

output

```
a = 256  
256
```

pycode

environment



Usage

Enclose the code that is going to be executed but not typeset.

```
1 \begin{pycode}
2 def sayMyName(name):
3     return "Your name is {0}".format(name)
4 sayMyName("Damian")
5 \end{pycode}
6 \py{sayMyName("Damian")}
7
```

output

Your name is Damian

pysub

environment



Usage

Similar to `\pys`. But this time this is an environment.

```
1 \begin{pysub}
2 1 + 5 = !{1 + 5} \\
3 Function output: !{sayMyName("Damian")} \\
4 2*32 = !{2**32}
5 \end{pysub}
6
```

output

```
1 + 5 = 6
Function output: Your name is Damian
2*32 = 4294967296
```

Usage

This environment enclose the code that is typeset and executed. Does not print any printed content even if autoprint flag is set to true.

```
1 \begin{pyblock}  
2 sayMyName("Damian")  
3 a = 125  
4 a + a  
5 \end{pyblock}  
6
```

output

```
sayMyName("Damian")  
a = 125  
a + a
```

pyverbatim

environment



Usage

This environment enclose the code that is typeset and not executed.

```
1 \begin{pyverbatim}  
2 sayMyName("Damian")  
3 a = 125  
4 a + a  
5 \end{pyverbatim}  
6
```

output

```
sayMyName("Damian")  
a = 125  
a + a
```

pyconsole

console environment



Usage

This environment treats its contents as series of commands passed to an active Python console. It shows input and output of commands.

```
1 \begin{pyconsole}  
2 a = [1, 2, 3]  
3 dir(a)  
4 print(a)  
5 \end{pyconsole}
```

output

```
>>> a = [1, 2, 3]  
>>> dir(a)  
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',  
>>> print(a)  
[1, 2, 3]
```

pycon

console inline command



Usage

This command executes code using emulated interpreter and shows the output back into the document, discarding the input.

```
1 \pycon{ dir(a) }
```

output

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',  
now exiting Console...
```

Other commands or functions

Which are not so important to have single slide for them.



- `\setpythontexoutputdir`

Other commands or functions

Which are not so important to have single slide for them.



- `\setpythontexoutputdir`
- `\setpythontexworkingdir`

Other commands or functions

Which are not so important to have single slide for them.



- `\setpythontexoutputdir`
- `\setpythontexworkingdir`
- `str`

Other commands or functions

Which are not so important to have single slide for them.



- `\setpythontexoutputdir`
- `\setpythontexworkingdir`
- `str`
- `add_dependencies`

Other commands or functions

Which are not so important to have single slide for them.



- `\setpythontexoutputdir`
- `\setpythontexworkingdir`
- `str`
- `add_dependencies`
- `before`

Other commands or functions

Which are not so important to have single slide for them.



- `\setpythontexoutputdir`
- `\setpythontexworkingdir`
- `str`
- `add_dependencies`
- `before`
- `after`

Beamer



Frames

PythonT_EX is compatible with Beamer. But beware, you need to use Beamer's fragile option for any frame containing typeset code.

Other languages



- Ruby

Other languages



- Ruby
- Octave

Other languages



- Ruby
- Octave
- Julia

Other languages



- Ruby
- Octave
- Julia
- Rust

Other languages



- Ruby
- Octave
- Julia
- Rust
- Bash

Bash

Available commands and environments



- bash

Bash

Available commands and environments



- bash
- bashblock

Bash

Available commands and environments



- bash
- bashblock
- bashverbatim

Bash

Available commands and environments



- bash
- bashblock
- bashverbatim
- bashsub

Część II

Python T_EXamples

4 Charts

- source code
- result

- 4 Charts
 - source code
 - result

- 5 Internet data
 - source code
 - result

- 4 Charts
 - source code
 - result

- 5 Internet data
 - source code
 - result

- 6 Dynamic tables
 - source code
 - result

Chart

Matplotlib



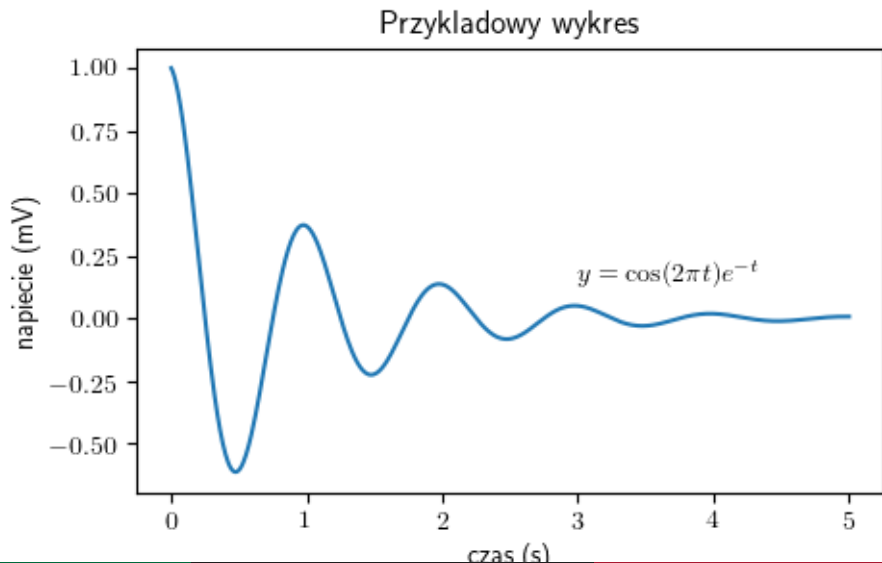
```

1 \begin{pycode}[chart]
2 from pylab import *
3 def f(t):
4     return cos(2 * pi * t) * exp(-t)
5 t = linspace(0, 5, 500)
6 y = f(t)
7 clf()
8 figure(figsize=(5, 3))
9 rc("text", usetex=True)
10 plot(t, y)
11 title("Przykładowy wykres")
12 text(3, 0.15, r"$y = \cos(2 \pi t) e^{-t}$")
13 xlabel("czas (s)")
14 ylabel("napiecie (mV)")
15 savefig("myplot.png", bbox_inches="tight")
16 print(r"\begin{center}")
17 print(r"\includegraphics[scale=1.0, keepaspectratio]{myplot.png}")
18 print(r"\end{center}")
19 \end{pycode}

```

Chart

File is saved to main folder by default



GPW

Using another session to improve speed of pythontex.



```
1 \begin{pycode}[internet]
2 from internet import getSymbolInfo
3 import time
4
5 wig20 = getSymbolInfo("WIG20")
6 kghm = getSymbolInfo("KGHM")
7 cd = getSymbolInfo("CDPROJEKT")
8 date = time.strftime("%Y/%m/%d");
9 \end{pycode}
10
11 \begin{exampleblock}{KGHM}
12 Current price: \py[internet]{wig20} PLN
13 \end{exampleblock}
```

GPW

Because it's funny to know the current price of stock.



WIG20

Current price: 2307.06

KGHM

Current price: 112 PLN

CDPROJEKT

Current price: 81.1 PLN

Actual price for date: 2017/06/12.

External files



```
1 \begin{pycode}[people]
2 from people import importPeople
3 people = importPeople()
4
5 print(r"\begin{tabular}{ l | r }")
6
7 print(r"{0} & {1} \\ \hline".format(people[0][0], people[0][1]))
8 people.pop(0)
9 for person in people:
10     print(r"{0} & {1} \\".format(person[0], person[1]))
11
12 print(r"\end{tabular}")
13 \end{pycode}
```

External files

The biggest hassle of creating tables is finally diminished.



List of people

Name	Surname
John	Smith
Victoria	Volpe
James	Jansen
Janice	Bishop
Charles	Stevens
Felicia	Appling
Nora	Sinkler

Część III

Dodatek



CTAN

PythonTeX Package documentation

<http://piotrkosoft.net/pub/mirrors/CTAN/macros/latex/contrib/pythontex/pythontex.pdf>



CTAN

Beamer user guide

<http://mirrors.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>