



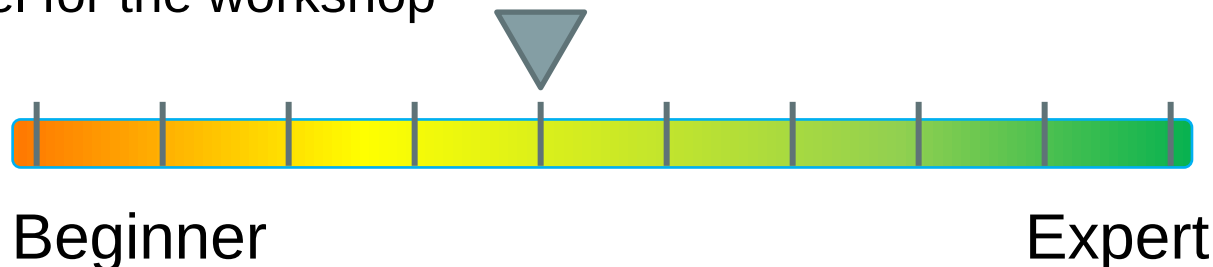
APROL Library Engineering Workshop



APROL Library Engineering Workshop

Time	Topic	Presenter
Day 1	Python Tutorial	Marböck Wolfgang
Day 2	Python API for DisplayCenter	Marböck Wolfgang
Day 3	Python API for DisplayCenter, LoginServer, UCB, ListViewWidget	Marböck Wolfgang

Aimed level for the workshop



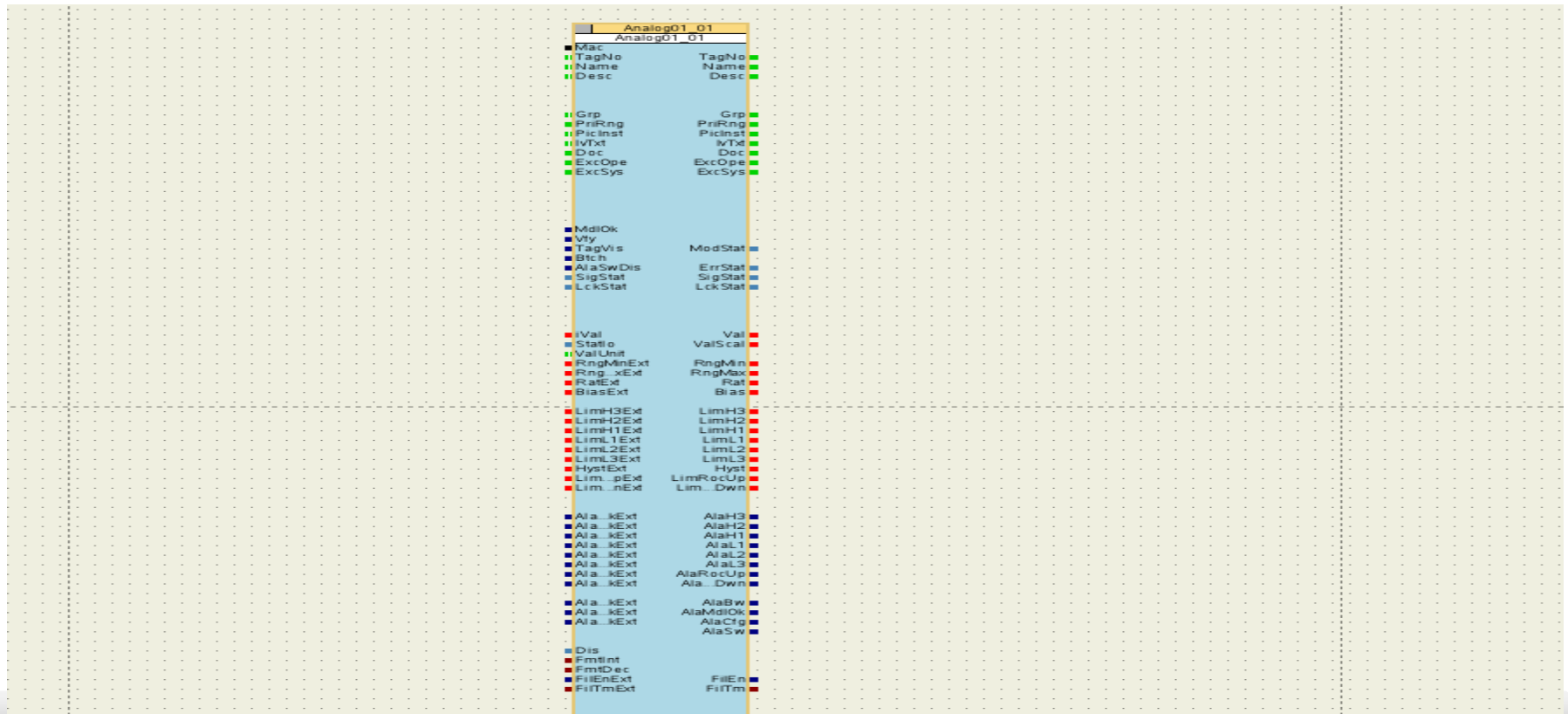


APROL Libraries



■ APROL Hypermacro concept

- Includes logic functionality (C- FUB)
- Includes visualization parts (graphic macros, faceplates)
- Includes control system blocks (alarm, trend)
- One block per typical, everything inside



APROL Libraries basic concepts 2/9

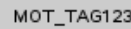
PERFECTION IN AUTOMATION
www.br-automation.com



Logic



Graphic Macro



MOT_TAG123

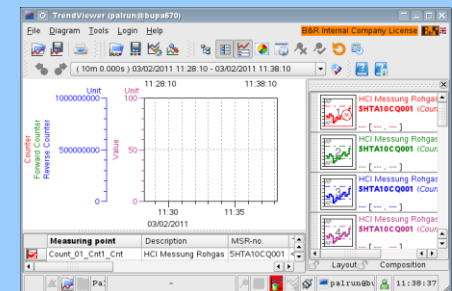
Faceplate



Alarm

Alarm	Value	Unit	Group	Operations counter interval 1	PV
0	100	NOT KNOWN	Group	Operations counter interval 1	PV - OpCh_R4_S5201_AW1_C.ASCH
1	100	NOT KNOWN	Group	Operations counter interval 1	PV - OpCh_R4_S5201_AW1_C.ASCH

Trend





Logic function blocks

- Pins
- Code = C programming
- Documentation B3.2.7
- Structured text (ST) and Sequential function chart (SFC) also possible

The screenshot displays the CaeManager software interface for a logic function block named 'CfgErrSel01'. The interface is divided into several panes:

- Master Data:** Shows the block name 'CfgErrSel01' and its dimensions '(10 x 6)'. Below this, four pins are listed: 'Err01', 'Err02', 'Err03', and 'Err04'. Each pin has a red square icon and a label 'Err'.
- Block:** Shows the block's internal structure, including a 'Name' field and a 'Bool' field.
- Declaration:** Shows the declaration of the block's variables, including 'Err01', 'Err02', 'Err03', and 'Err04'.
- Program:** Shows the C code for the block. The code includes comments for history and start, a declaration of 'ErrChanged', and logic for detecting changes in the error pins.
- Documentation:** Shows the documentation for the block, including the B3.2.7 standard.
- Init Code:** Shows the initialization code for the block.
- Exit Code:** Shows the exit code for the block.

The C code in the Program pane is as follows:

```
1  /*=====*/
2  /*.....*/
3  /*History:.....*/
4  /*20.04.2010→Start.....*/
5  /*.....*/
6  /*=====*/
7
8  /*DECLARATION*/
9
10 BOOL ErrChanged;
11
12 /*local variables for value changed detection*/
13 BOOL Err01Changed := (Err01 != Err01old)?1:0;
14 BOOL Err02Changed := (Err02 != Err02old)?1:0;
15 BOOL Err03Changed := (Err03 != Err03old)?1:0;
16 BOOL Err04Changed := (Err04 != Err04old)?1:0;
17
18 /*INITIALIZATION*/
19 Ala := 0;
20 ErrChanged := 0;
21
22
23 /*PROGRAM START*/
24
25 if(Err04Changed) → → /*Err04 has changed it's value. Err04 LOWEST
26 {
```



Graphic block

- Pins
- Graphical part
- Python Code
- Documentation B3.2.8

Master Data

Block

I/O

Preview

Python Code

Graphic Code

Documentation

Master Data

Block

I/O

Preview

Python Code

Graphic Code

Documentation

Master Data

Block

I/O

Preview

Python Code

Graphic Code

Documentation

Master Data

Block

I/O

Preview

Python Code

Graphic Code

Documentation

Master Data

Block

I/O

Preview

Python Code

Graphic Code

Documentation

Master Data

Block

I/O

Preview

Python Code

Graphic Code

Documentation

Value U
TagNo

I/O-variable						
P	Name	Remanent	IEC type	Default value	Unit	Description
Block inputs:						
1	TagNo	---	LSTRING	TagNo	---	Tag Number
2	Name	---	LSTRING		---	Name
3	Desc	---	LSTRING		---	Description
5	TagVis	---	BOOL	1	---	TagNo Visible
6	FaceEn	---	BOOL	1	---	Faceplate enal
7	Val	---	DINT	0	---	Value
8	FmtInt	---	USINT	0	---	Format Integer
9	FmtDec	---	USINT	0	---	Format Decim
10	Unit	---	LSTRING	*C	---	Unit
12	PriRng	---	SSTRING	0	---	Priority range
16	ModStat	---	DWORD	0	---	Mode state
17	ErrStat	---	DWORD	0	---	Error summary
18	LckStat	---	BYTE	0	---	Lock state
19	SigStat	---	BYTE	0	---	Signal Status
Block outputs:						
1	Lvl1Face	---	FACEPLATE		---	Level 1 facepla
Unconnected global graphic I/Os:						
Local graphic I/Os:						
BgCol						
Bli						

```

1 #####
2 #####
3 ##-History:-----##
4 ##-20.04.2010→Start-----##
5 ##-09.07.2010→use adapted names for functions from paldef_addon
6 #####
7 #####
8
9 #--*-coding: utf-8-*-
10 import paldef
11
12 def Alarm():
13     →# Get textcolor, backgroundcolor, etc. for dynamization
14     →paldef.GetAlaDyn(block.pin.ErrStat,block.pin.PriRng,block.local.
15
16
17 ##### Hooks
18 #####
19
20
21 ##### Function Collector for "None" detection
22 func_collector = paldef.class_func_collector( block, block.local.Nor
23
24
25 ##### Mode Image
26 DC.add_hook( block.pin.ModStat.changed_hook, →func_collector( pal
27
28
29 DC.add_hook( block.pin.FmtInt.changed_hook, →func_collector( pal
30 DC.add_hook( block.pin.FmtDec.changed_hook, →func_collector( pal
31 DC.add_hook( block.pin.Unit.changed_hook, →func_collector( pal
32
33
34 ##### Alarm
35 DC.add_hook( block.pin.ErrStat.changed_hook, →func_collector(
36
37 ##### Lock
38 DC.add_hook( block.pin.LckStat.changed_hook, →func_collector( pal
39
40 ##### Dynamization Signal Status of Input Signals
41 DC.add_hook( block.pin.SigStat.changed_hook, →func_collector( pal
42
43 ##### Mac Tooltip
44 DC.add_hook( block.pin.TagVis.changed_hook, →func_collector( pal

```



■ available animations:

The following are the basic functions available for animation:

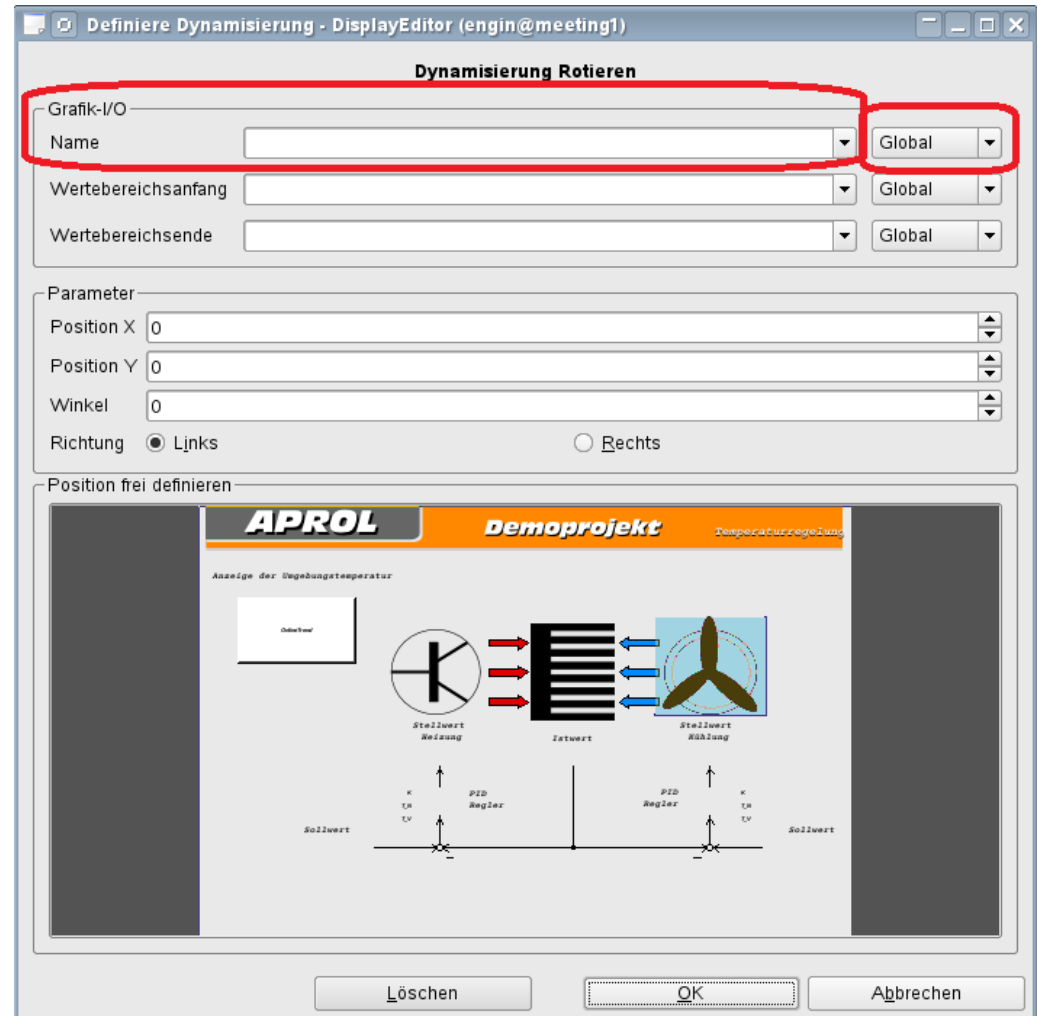
Animation type	Description
Blinking	Objects can be blink in many different ways depending on a process variable.
Colors	Objects can change colors depending on a process variable. Background and foreground colors, fill patterns, and line types can be changed.
Rotate	Objects rotate around a definable fixed point depending on a process variable.
Move	Makes it possible to move objects to a different location in the process graphic. Movement along the x-axis can be defined independently of that on the y-axis.
Scale	Objects can change their size depending on two process variables. The x and y axes can be defined independently of one another.
Percent	Objects are trimmed depending on a process variable and no longer displayed in full size.
Text	The actual value of a process variable is displayed. STRING variables can be displayed.
Text list	Defined texts can be displayed depending on a process variable.
Image list	Serves to dynamically display 1 from n images - depending on a process value. E.g. a significant increase in productivity can be reached during the creation of status displays from your system via image (icon).
Invisible	The object is switched invisible.



■ Apply animation to a graphical element

■ Local/Global graphic IO

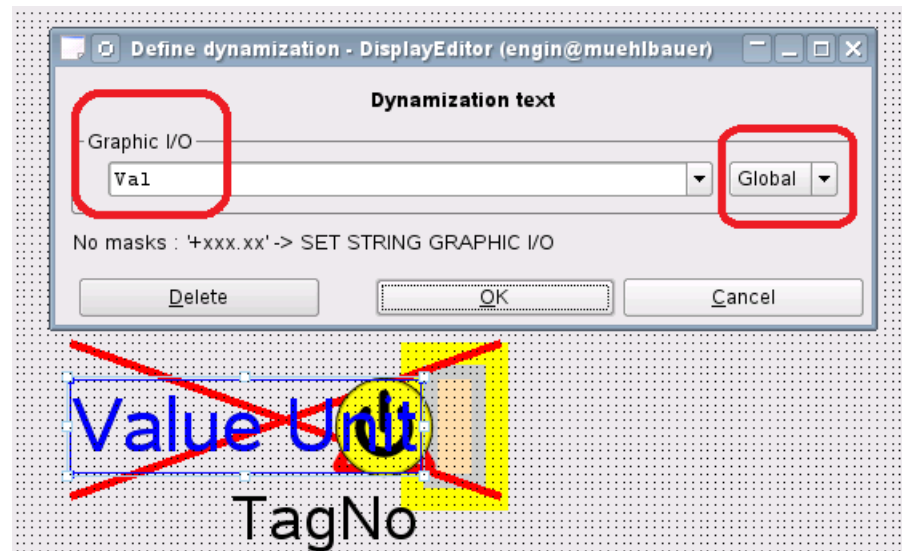
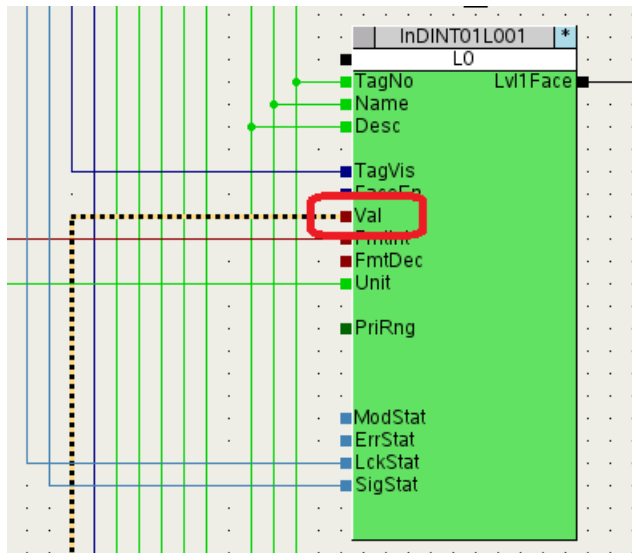
■ Documentation B3.2.8.2.3





■ Animation with global graphic IOs

- Use a pin of the graphic block as signal source
- The pin of the graphic block can be connected to a signal coming from a logic block, input pin, gateway io, hm- constant, ...

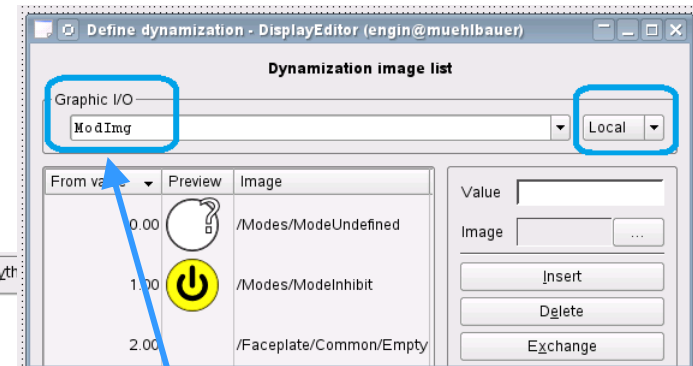




■ Animation with local graphic IOs (= Python variables)

■ Use pin(s) of the graphic block in Python Code.

■ The result of the Python Code is a local graphic IO



The screenshot shows the APROL software interface with three main panels: a graphic block list on the left, a Python code editor in the center, and a preview window on the right.

Graphic Block List (Left): A table with columns for TagNo, Name, Desc, TagVis, FaceEn, Val, FmtInt, FmtDec, Unit, PriRng, and ModStat. The 'ModStat' column is highlighted in blue.

Python Code (Center): The code is as follows:

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14 block.local.TxtCol,block.local.BgCol,block.local.El1,(None)  
15  
16  
17 #####  
18 #####  
19  
20  
21  
22 .local.NoneIndicator..)  
23  
24  
25  
26 ctor( paldef.GetModeImage, block.pin.ModStat, block.local.ModImg )  
27  
28  
29 ctor( paldef.GetDynWidth, block.name(), block.pin.FmtInt, block.pin.  
30 ctor( paldef.GetDynWidth, block.name(), block.pin.FmtInt, block.pin.  
31 ctor( paldef.GetDynWidth, block.name(), block.pin.FmtInt, block.pin.  
32  
33  
34 collector( Alarm )  
35
```

Preview Window (Right): The preview window shows a power button icon. A red 'X' is drawn over the 'Value' and 'TagNo' labels. A blue arrow points from the 'ModStat' pin in the Python code to the 'ModImg' dropdown in the 'Define dynamization - DisplayEditor' window.



- In general all animations are possible without Python (use global graphic IOs)
- Python is available for extended possibilities with animations
- Python-API gives you access to system events via HOOKS
- Python-API offers FUNCTIONS to perform specific actions

The screenshot shows the CaeManager software interface. The title bar reads 'CaeManager (engin@muehlbauer) <2>'. The menu bar includes 'File', 'Edit', 'Build', 'View', 'Reports', 'Extras', 'Login', and 'Help'. The toolbar contains various icons for file operations and development. The main window displays a Python module named 'paldef' with the description 'Python module, function definitions'. The code is shown in a text editor with line numbers from 612 to 634. The code defines a function 'CheckAccessRights' that checks access rights for a block and returns a tuple of check results.

```
612 #####
613 ### Check Access Rights
614 #####
615 def CheckAccessRights(blockname, LockPin, RightPin, ExcOpePin, ExcSysPin, ExcOffStrPin):
616     → Lock = Get( LockPin )
617     → #Get( .RightPin ) .## Only check for None, no need for the result
618     → Right = RightPin.right()
619     →
620     → if Lock:
621     →     → LogicCheckText = DC.tr('Locked from Logic;')
622     → else:
623     →     → LogicCheckText = ''
624     →
625     → if (DC.login.hasRight( .Right )):
626     →     → LoginCheckText = ''
627     → else:
628     →     → LoginCheckText = DC.tr('Operator has no right:')+Right+';'
629     →
630     → ExcOpeCheckText, ExcSysCheckText, nouse1, nouse2 = CheckExclusiveAccessRights( .blockname, .ExcOpePin, .ExcSysPin, .ExcOffStrPin ) .### tmp not us
631     →
632     → return ( .LogicCheckText, .LoginCheckText, .ExcOpeCheckText, .ExcSysCheckText ) .## returns '' if access granted
633
634
```



Python



- Python v2.6
- www.python.org
- APROL Product documentation - X3 - Python Documentation
- APROL Product Documentation - X11 – Python beginners course
- <http://openbook.galileocomputing.de/python/> (German)





Python Tutorial



■ Tutorial “A Byte of Python”

- from <http://www.swaroopch.com/notes/Python>
- or <http://www.ibiblio.org/swaroopch/byteofpython/read/index.html>

■ Introduction

- <http://www.ibiblio.org/swaroopch/byteofpython/read/introduction.html>
- simple & powerful

■ Features

- <http://www.ibiblio.org/swaroopch/byteofpython/read/features-of-python.html>
- High level language
- Portable
- Interpreted
- Object oriented
- Extensive libraries



■ Using the interpreter

- <http://www.ibiblio.org/swaroopch/byteofpython/read/interpreter-prompt.html>

■ Using a source file

- <http://www.ibiblio.org/swaroopch/byteofpython/read/source-file.html>

■ Executable program

- <http://www.ibiblio.org/swaroopch/byteofpython/read/executable-python-programs.html>

■ Workshop

- Start python interpreter
- “Hello World” program with interpreter
- “Hello World” program with executable source file



■ Basics

■ Numbers

- <http://www.ibiblio.org/swaroopch/byteofpython/read/numbers.html>

■ Strings

- <http://www.ibiblio.org/swaroopch/byteofpython/read/strings.html>

■ Variables

- <http://www.ibiblio.org/swaroopch/byteofpython/read/variables.html>
- <http://www.ibiblio.org/swaroopch/byteofpython/read/identifier.html>
- <http://www.ibiblio.org/swaroopch/byteofpython/read/data-types.html>

■ Objects

- <http://www.ibiblio.org/swaroopch/byteofpython/read/data-type-object.html>

■ Logical and physical lines

- <http://www.ibiblio.org/swaroopch/byteofpython/read/logical-and-physical-lines.html>

■ Indentation

- <http://www.ibiblio.org/swaroopch/byteofpython/read/indentation.html>

■ Operators

- <http://www.ibiblio.org/swaroopch/byteofpython/read/operators.html>

Numbers



■ Workshop

- Test in interactive mode
 - Numbers
 - Strings
 - Variables
 - Objects
 - Logical and physical lines
 - Indentation
 - Operators



■ Control Flow

- if
 - <http://www.ibiblio.org/swaroopch/byteofpython/read/if-statement.html>
- while
 - <http://www.ibiblio.org/swaroopch/byteofpython/read/while-statement.html>
- for
 - <http://www.ibiblio.org/swaroopch/byteofpython/read/for-loop.html>
- break
 - <http://www.ibiblio.org/swaroopch/byteofpython/read/break-statement.html>
- continue
 - <http://www.ibiblio.org/swaroopch/byteofpython/read/continue-statement.html>



■ Workshop

- Test with program samples from documentation
 - if
 - while
 - for
 - break
 - continue



■ Functions

- <http://www.ibiblio.org/swaroopch/byteofpython/read/function-parameters.html>

■ Functions with optional parameters

- <http://www.ibiblio.org/swaroopch/byteofpython/read/default-argument-values.html>

■ Functions with keyword arguments

- <http://www.ibiblio.org/swaroopch/byteofpython/read/keyword-arguments.html>

■ Return Statement

- <http://www.ibiblio.org/swaroopch/byteofpython/read/return.html>



■ Workshop

- Test with program samples from documentation
 - functions
 - functions with optional parameters
 - functions with keyword parameters
 - functions with return value



■ Modules

- <http://www.ibiblio.org/swaroopch/byteofpython/read/modules.html>

■ The dir() function

- <http://www.ibiblio.org/swaroopch/byteofpython/read/dir.html>

■ Workshop

- Import the module “math”
- Use attribute “pi” of the module
- Use the dir function for the module “math”



■ Data structures

■ List

- <http://www.ibiblio.org/swaroopch/byteofpython/read/list.html>
- Square brackets []
- mutable

■ Workshop

- Define a shopping list
- Use the append-method
- Use the remove-method
- Use the sort-method
- Use the reverse-method



■ Data structures

■ Tuple

- <http://www.ibiblio.org/swaroopch/byteofpython/read/tuple.html>
- Round bracket ()
- immutable

■ Workshop

- Define a tuple
- Use the count-method



■ Data structures

■ Dictionary

- <http://www.ibiblio.org/swaroopch/byteofpython/read/dictionary.html>
- Pairs of key:value
- Curly brackets { }
- keys need to be immutable
- keys need to be unique
- values can be either mutable or immutable

■ Workshop

- Define a dictionary
 - Key should be a name
 - Value should be phone number
- Print the number of elements in the dict with the function len()
- Use the keys-method; value-method
- Delete one element of the dictionary using “del” operator



■ Sequences

- <http://www.ibiblio.org/swaroopch/byteofpython/read/sequences.html>
- List, tuple, string
 - Indexing
 - Slicing

■ Workshop

- Define a shopping list
- Try indexing
- Try slicing



■ Standard libraries

- Extensive standard libraries – “Batteries included”
- <http://docs.python.org/tutorial/stdlib.html#>
- Examples:
 - math: sin, exp, log, pow, pi, ...
 - random
 - os.path: dirname, exists, ..
 - csv: csv-file reading and writing
 - os: environ, listdir, mkdir, remove, rename



YOUR GLOBAL PARTNER FOR AUTOMATION EXCELLENCE