Welcome to
Rave from Python & PRT I

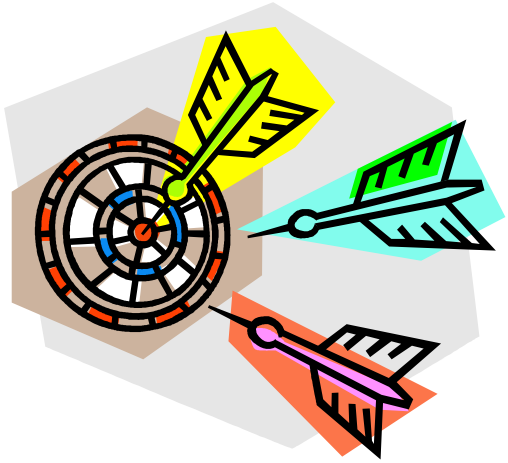
Developed by Crew Academy

Major Release 22

Course goals

After the course you will be able to:

- **create and maintain basic PRT reports**
- **use Rave's Python API.**



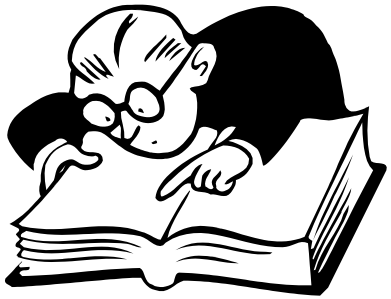
PRT and **Rave's Python API** are component-independent.
Studio is not needed, but used as “mother component” in the course.

Course prerequisites

To get the most out of this course, you need basic knowledge and experience from:

- Python
- Rave
- Studio

Agenda day 1



start 09:00

PRT – Introduction

Coffee break

PRT – Basic Objects and Properties

Lunch

PRT – Pages, Header, Footer, Cross-ref

Coffee break

Rave API – Basics

end 17:00

Agenda day 2

start 09:00

Recap of Day 1

Rave API – consider data

Coffee break

Rave API – more about contexts

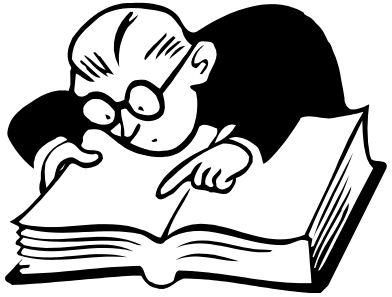
Lunch

Rave API – rule failures + constraints

Coffee break

Rave API – transforms + performance

end 17:00



Introduction PRT – some names

- **Rave Publisher**
  all tools for report generation in Crew and Fleet products
- **PRT** (Python Report Toolkit)
  the new tool for report generation
- **PDL** (Page Description Language)
  the old report language
- **CRG** (Carmen Report Generator)
  former name of Rave Publisher

Basics

PRT is:

- a tool for report generation
  - will replace PDL
  - output formats: TXT, PDF and HTML
  - support for interaction and graphics.
- object-oriented Python API
- embedded in and supported by Studio
- developed and maintained by Jeppesen in Göteborg
  - implemented in Python/C++
  - a part of CARMSYS

You have to use low level functions.
Covered in PRT2.

Show some PRT reports

Documentation

- **API**

  Help> API Documentation> Python Report Toolkit

In the course material the documentation created by *WebWorks* you start with  is called **System Help.**

Note:
The search function in **System Help** does not consider the API documentation (Rave, PRT…)

(or *manual pages*).

- **General**

  System Help> Development> Rave Publisher PRT Reference

- **Example reports**

  `$CARMSYS/lib/python/carmensystems/publisher/examples`

First example – Python code

Contains the entire PRT api

```
> cat SlideIntro1.py:
import carmensystems.publisher.api as p

class Report(p.Report):
    def create(self):
        self.add(p.Text('Example 1 is here'))
```

When PRT generates a report:

1. The `module` specified as argument is loaded.
2. An instance of the class `Report` is created.
3. The method `create` is called.

You can also use the name of the module as class name. (backward compatibility)

First example – run without Studio

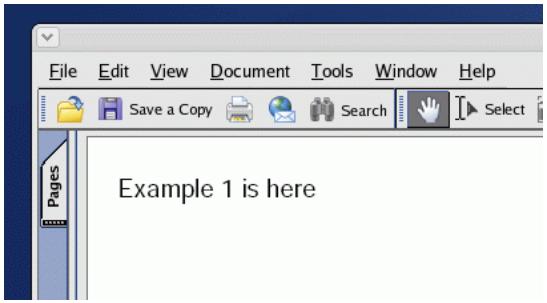
You need some of the environment variables which are defined when Studio is started.

Use an **xterm** started from **Studio.**

Run the shell commands:

```
> cd $CARMUSR/lib/python/report_sources/report_menu
> publisher SlideIntro1
> evince SlideIntro1.pdf &
```

For documentation. `publisher -h`



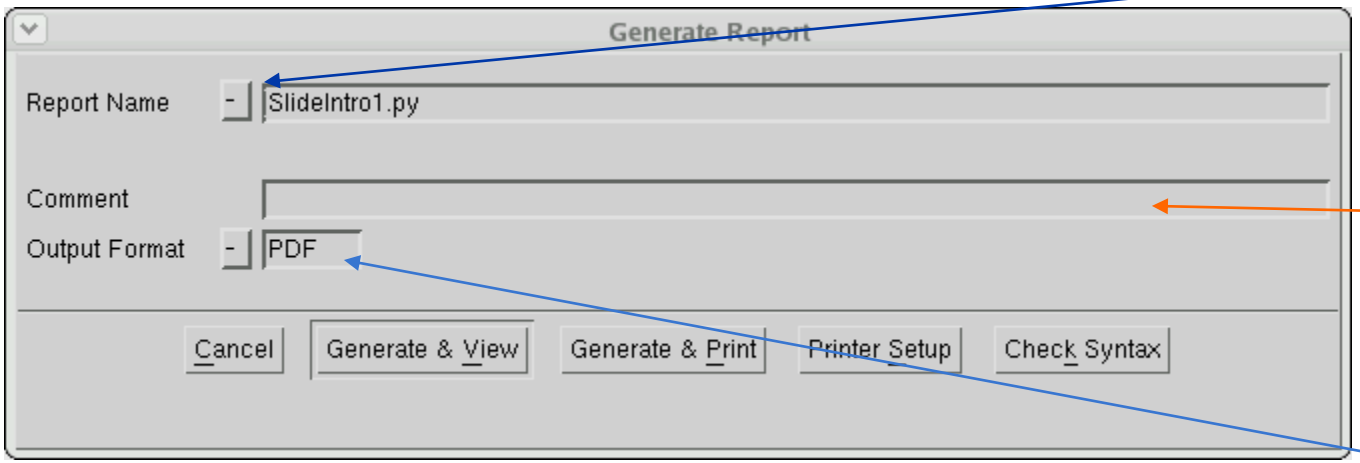
You can be in any directory.
Current directory is a member of `sys.path`, which is used by Python to define module directories.

From Studio – example

We want to generate our report by the menu entry
Planning Tools> Generate Report

Put the file in:
`$CARMUSR/lib/python/report_sources/report_menu`

Activate the menu entry:



Added to the pick list

keywords:
`report_comment`
`crg_comment`

PDF or HTML

Studio – what you must know

In the course we'll generate all reports using Studio.
Good to know:

Studio has a rich Python API.
Well described in the course:
Python in Studio.

- `CARMUSR/lib/python`, `CARMSYS/lib/python` and `CARMSYS/lib/python/x86_64_linux` added to `sys.path`

- Studio looks for report definitions in module packages below `CARMUSR/lib/python/report_sources`
  e.g.: `crr_window_general` General pop-up in trip window> **Generate Report**
  `crr_window_object` Object pop-up in trip window> **Generate Report**
  `report_menu` **Planning Tools> Generate Report**

See the
Quick Reference
for a complete list

Make sure that there is a file called **`__init__.py`** in the directories.

- You have to `reload` the report module when you have made changes
  Use e.g.: **Special> Scripts> Python Code Manager…> Reload**

- Error messages are written to Studio's log file.
  Use e.g. **Special> Logs etc.> Tail of Studio Log**

Show example

Developer Workspace (DWS)

Step-by-step guide to get started in the end of the course material.

Demo:

- start
- run code in Studio
- debug.

Exercise

Now it is time for exercise 1

PRT – basic objects and properties

In this section we will look at:

Objects:
- Text
- Image
- Column
- Row
- Isolate

Properties:
- size
- alignment
- padding
- border
- spanning
- colour
- font

Text objects

Used to display texts.

`Text(*texts, **properties)`

Examples:

```
def create(self):
    self.add(p.Text("text 1"))
    self.add(p.Text("text ", 2))
    self.add(p.Text([1, 2, 3]))
```

`Report` is a subclass of `Column`. The `add` method puts an object at the bottom of a column.

Concatenate

Any object could be used.
The `__str__` method is called.

```
text 1
text 2
[1, 2, 3]
```

SlideObj1.py

Image objects

Used to display JPEG pictures.

`Image(path, **properties)`

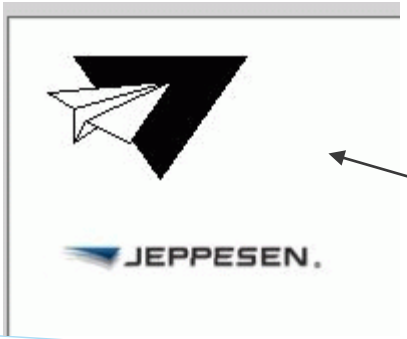
In `$CARM[USR|SYS]/images`

Example:

```
import carmensystems.publisher.api as p
class Report(p.Report):
    def create(self):
        self.add(p.Image("examples/customer_logo.jpg"))
        self.add(p.Image("jepplogo.jpg"))
```

in CARMSYS

In the course CARMUSR



Default: No scaling

SlideObj2.py

Column & Row objects

Container objects.

Used to order objects.

prt objects

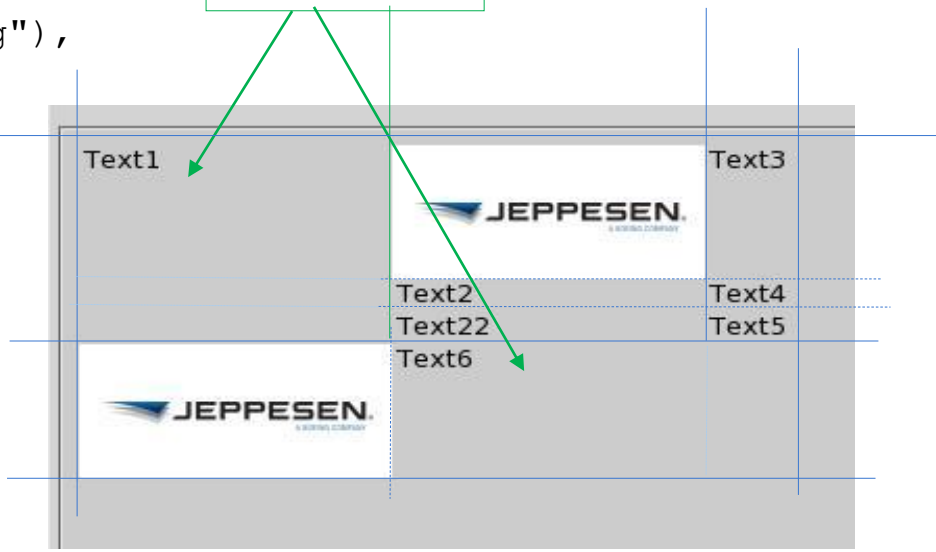
```
Column/Row(*components, **properties)
add(self, component)
```

Example:

```
self.set(background=sp.Grey)
r = p.Row(p.Text("Text1"),
          p.Column(p.Image("jepplogo.jpg"),
                   p.Text("Text2"),
                   p.Text("Text22")))
r.add(p.Column(p.Text("Text3"),
               p.Text("Text4"),
               p.Text("Text5")))
self.add(r)
self.add(p.Row(p.Image("jepplogo.jpg")),
               p.Text("Text 6")))
```

The last object in a row/column spans more than one cell in the grid

All objects are put in a **grid**.



SlideObj4.py

Isolate objects

Container object.

Creates an independent grid for all objects in a row/column.

```
Isolate(object, **properties)
```

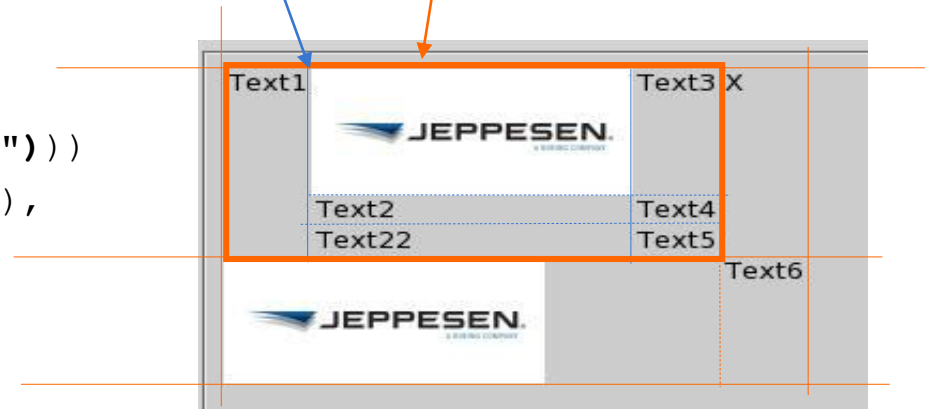
A row or a column

Example:

```
r = p.Row(p.Text("Text1"),
          p.Column(p.Image("jepplogo.jpg"),
                   p.Text("Text2"),
                   p.Text("Text22")))
r.add(p.Column(p.Text("Text3"),
               p.Text("Text4"),
               p.Text("Text5")))
self.add(p.Row(p.Isolate(r), p.Text("X")))
self.add(p.Row(p.Image("jepplogo.jpg")),
               p.Text("Text 6")))
```

One cell in the report grid

Independent grid



SlideObj5.py

Properties – general

- Properties could be defined:
  - when an object is created
  - by the `set` method.
- The supported properties vary from object to object.
- Properties can have different meaning for different objects.
- Documentation:
  - the supported properties for each kind of object are found in the API documentation
  - in the section *Development> Rave Publisher PRT Reference> Geometrical model* of ***System Help*** you find detailed descriptions of many properties.

Properties – colours

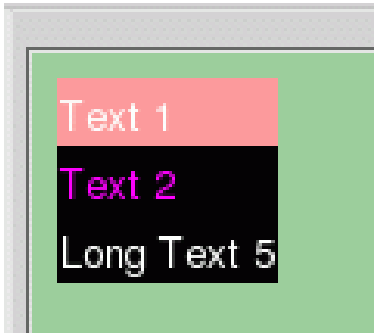
- Argument names: `colour`, `background`
- May be:
  - An RGB code.
    - Example: `"#FF0000"` (red)
  - A colour palette attribute.
    - Colour palettes:
      - Class name: `ColourPalette`
      - You can create and use colour palettes. Example:
        ```
        c = p.ColourPalette(blue='#0000FF', green='#00FF00')
        c.green == '#00FF00'  # This is true
        ```
      - There are two predefined palettes
        - Colours used in Studio's drawing areas: `NiceToHaveIQ/lib/python/nth/studio/prt/studiopalette.py`
        - Nordic Light `$CARMSYS/lib/python/carmensystems/publisher/nordiclight.py`
- If set on container objects – default for all objects inside.

Colours – example

```
from report_sources.include.studiopalette import studio_palette as sp
import carmensystems.publisher.api as p

class Report(p.Report):
    def create(self):
        self.set(background=sp.Green,
                 colour=sp.White)
        c = p.Column(background=sp.Weekday)
        c.add(p.Text("Text 1",
                     background=sp.Red))
        c.add(p.Text("Text 2",
                     colour="#FF00FF"))
        c.add(p.Text("Long Text 5"))
        self.add(c)
```

For the entire report (although the "report column" is smaller)

Default for objects in the container



SlideObj3.py

Properties – font

- Argument name: `font`
- A font object. Created by:

```
font(face=None, size=None, weight=None, style=None)
```

```
size   - points or None
face   - SERIF, SANSSERIF, MONOSPACE, None
weight - BOLD, None
style  - ITALIC, None
```

`None` gives default

- If set on container objects – default for all objects inside.
- Example:

```
import carmensystems.publisher.api as p
my_fonts =  [("None", None),
             ("sansserif", p.SANSSERIF),
             ("serif", p.SERIF),
             ("monospace", p.MONOSPACE)]
class Report(p.Report):
    def create(self):
        for fsize in (None, 40):
            for fname, fface in my_fonts:
                self.add(p.Text(fname, " ", fsize,
                                font=p.font(size=fsize, face=fface)))
```

SlideObj6.py

Font – example

None None
sansserif None
serif None
monospace None

`Size 10` is default

None 40
sansserif 40
serif 40
monospace 40

`Face p.SANSSERIF` is default

Properties – border

- Argument name: `border`
- Drawn round grid cells.

An **object** is often smaller than the grid cell

- A border object from:

```
border(left=None, top=None, right=None, bottom=None,
       inner_floor=None, inner_wall=None,    I
       colour='#000000')
```

`left, top, right, bottom`: width in points

`inner_*` : default border width for objects inside Row / Column

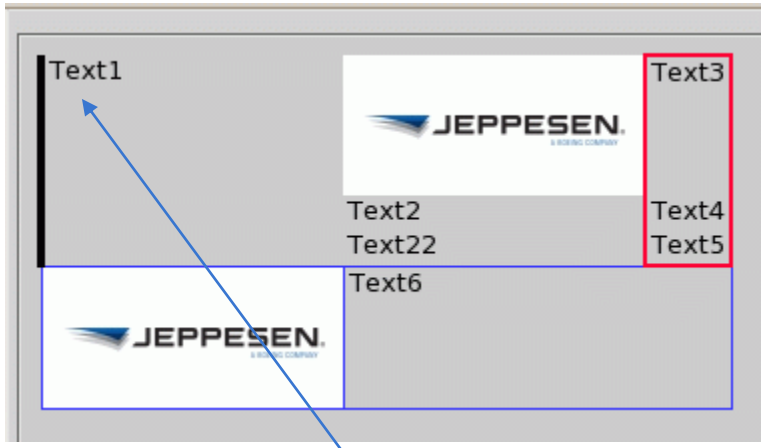
Not inside `Isolate`

- For convenience also:

```
border_frame(w, colour='#000000'):
    return border(w, w, w, w, None, None, colour)
border_all(w, colour='#000000'):
    return border(w, w, w, w, w, w, colour)
```

Properties – border example

```
class Report(p.Report):
    def create(self):
        self.set(background=sp.Grey)
        r = p.Row(p.Text("Text1",
                         border=p.border(left=3)),
                  p.Column(p.Image("jepplogo.jpg"),
                           p.Text("Text2"),
                           p.Text("Text22")))
        r.add(p.Column(p.Text("Text3"),
                       p.Text("Text4"),
                       p.Text("Text5"),
                       border=p.border_frame(2, colour=sp.BrightRed)))
        self.add(r)
        self.add(p.Row(p.Image("jepplogo.jpg"),
                       p.Text("Text6"),
                       border=p.border_all(1, colour=sp.DarkBlue)))
```



The Text object is much smaller than the grid cell.

SlideObj7.py

Properties – size

- Argument names: `width`, `height`
- Points (pt) (1 pt = 1/72 inch)
- Size of the **object** (not the cell in the grid)
- For containers:
  - specifies minimum size
  - `height` not available for `Column`
  - `width` not available for `Row`
- For `Text`:
  - Default – **one row** containing the entire string
  - `width` – if the string is longer you get row breaks
  - `height` – minimum

`\n` is not supported

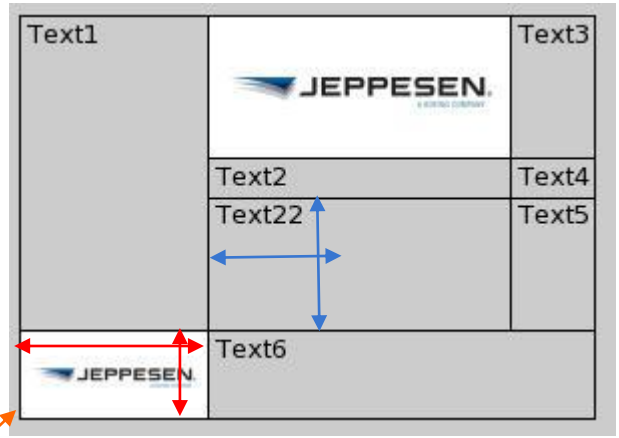
Note: Words are never split.

We have discussed the need for the `Text` property `maxwidth`.

Properties – size example

```
class Report(p.Report):
    def create(self):
        self.set(border=p.border_all(1), background=sp.Grey)
        r = p.Row(p.Text("Text1"),
                  p.Column(p.Image("jepplogo.jpg"),
                           p.Text("Text2"),
                           p.Text("Text22",
                                  width=50,
                                  height=50)))
        r.add(p.Column(p.Text("Text3"),
                       p.Text("Text4"),
                       p.Text("Text5")))
        self.add(r)
        self.add(p.Row(p.Image("jepplogo.jpg",
                               width=70),
                       p.Text("Text6")))
```

Note: Scaled images sometimes cause problems for HTML. **Avoid!**



Proportionally scaled picture

Properties – padding

- Argument name: `padding`
- Points (1 point = 1/72 inch)
- Minimum distance between object and cell grid
- Default is `2,2,2,2` for `Text`
- Defined by:

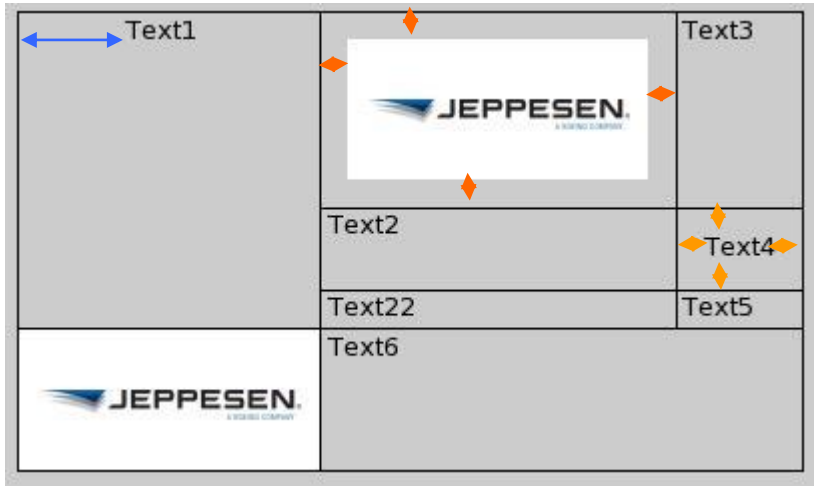
  `padding``(left=None, top=None, right=None, bottom=None)`
- Must be set on each atomic object. No default handling.

  You can use a factory function instead.

You can not create subclasses of e.g. `Text`.

Properties – padding example

```
p10 = p.padding(10, 10, 10, 10)

def TextP10(*args, **kw):
    if not "padding" in kw:
        kw["padding"] = p10
    return p.Text(*args, **kw)

class Report(p.Report):
    def create(self):
        self.set(border=p.border_all(1),
                 background=sp.Grey)
        r = p.Row(p.Text("Text1",
                         padding=p.padding(left=40)),
                  p.Column(p.Image("jepplogo.jpg",
                                   padding=p10),
                           p.Text("Text2"),
                           p.Text("Text22")))
        r.add(p.Column(p.Text("Text3"),
                       TextP10("Text4"),
                       p.Text("Text5")))
        self.add(r)
        self.add(p.Row(p.Image("jepplogo.jpg"),
                       p.Text("Text6")))
```



Minimum size of a cell is **object size + padding.**

SlideObjA

Properties – alignments

- Argument names: `align, valign`
- Where to put the object in the grid cell.
- Possible values:

  `align:` `LEFT`, `CENTER` and `RIGHT`

  `valign:` `TOP`, `CENTER` and `BOTTOM`

`TOP`/`LEFT` is default

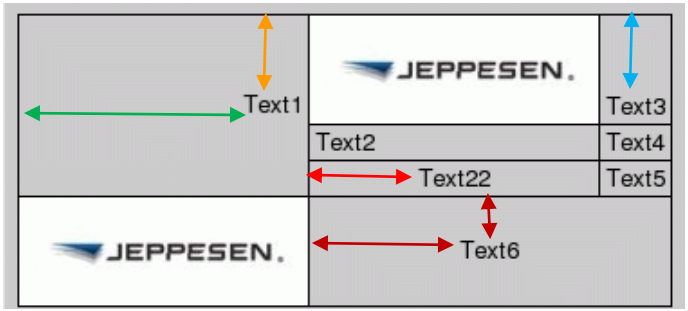
- No default values propagated from the container – you must specify per atomic object
  - Use factory functions. Example:

```
def TextC(*args, **kw):
    kw["align"] = kw.get("align", p.CENTER)
    kw["valign"] = kw.get("valign", p.CENTER)
    return p.Text(*args, **kw)
```

Properties – alignments example

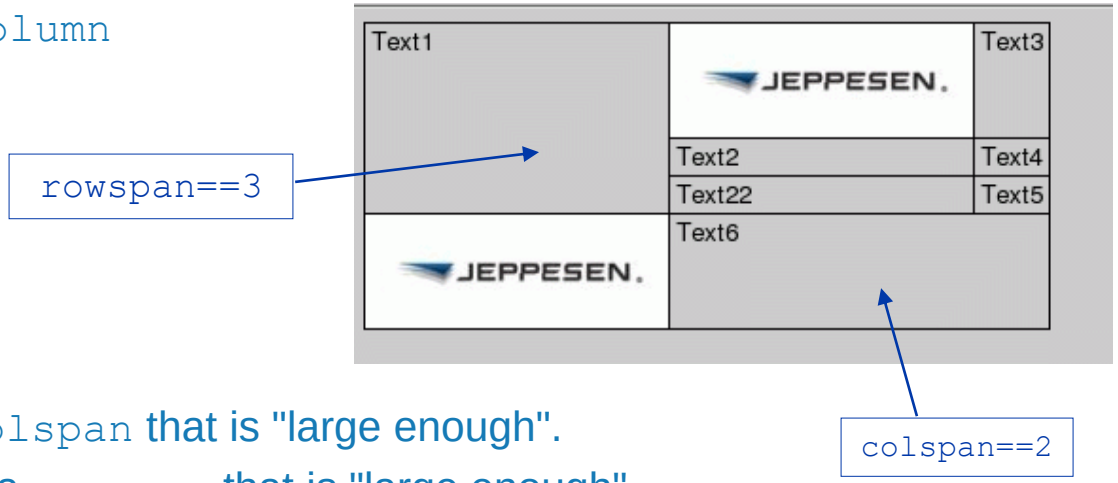
```
class Report(p.Report):
    def create(self):

        self.set(border=p.border_all(1), background=sp.Grey)),
        r = p.Row(TextC("Text1", align=p.RIGHT),
                  p.Column(p.Image("jepplogo.jpg"),
                           p.Text("Text2"),
                           TextC("Text22")))
        r.add(p.Column(p.Text("Text3", valign=p.BOTTOM),
                       p.Text("Text4"),
                       p.Text("Text5")))
        self.add(r)
        self.add(p.Row(p.Image("jepplogo.jpg"),
                       TextC("Text6")))
```

Only matters if there is "extra space" in the cell.



Properties – spanning

- Argument names: `rowspan`, `colspan`
- Specifies number of cells in the grid for an object
- Provided by all objects. Exceptions:
  - `colspan` is not supported for `Row`
  - `rowspan` is not supported for `Column`

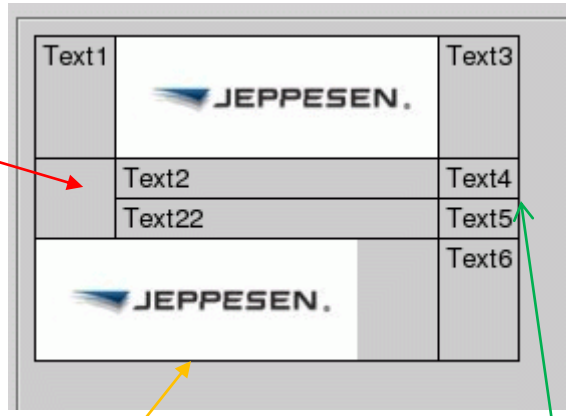


- Default values:
  - normally 1
  - the last object in a `Row` get a `colspan` that is "large enough".
  - the last object in a `Column` get a `rowspan` that is "large enough".
  - a `row/column` get the values from the objects inside (largest value).

Property – spanning example

```
class Report(p.Report):
    def create(self):
        self.set(border=p.border_all(1),
                 background=sp.Grey)
        r = p.Row(p.Text("Text1", rowspan=1),
                  p.Column(p.Image("jepplogo.jpg"),
                           p.Text("Text2"),
                           p.Text("Text22")))
        r.add(p.Column(p.Text("Text3"),
                       p.Text("Text4", rowspan=2),
                       p.Text("Text5")))
        self.add(r)
        self.add(p.Row(p.Image("jepplogo.jpg", colspan=2),
                       p.Text("Text6")))
```

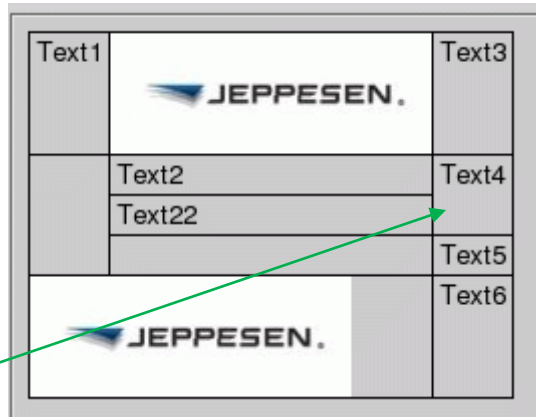
empty cell



ignored – there must be rows available

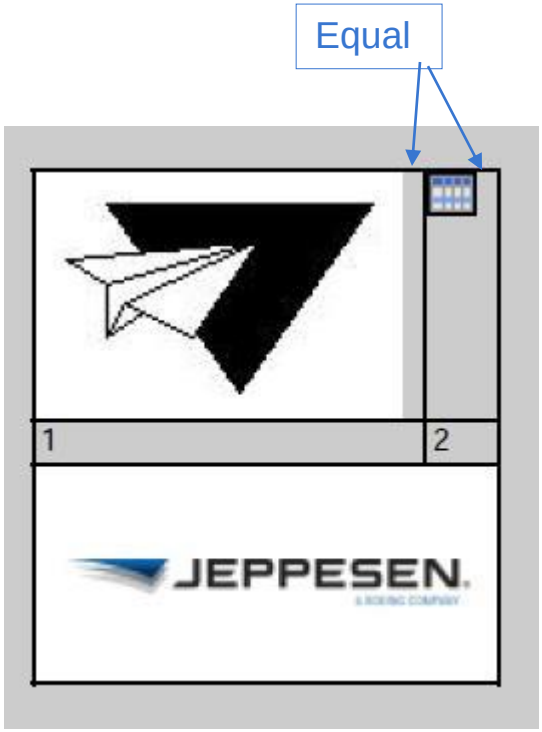
Property – spanning example II

```
class Report(p.Report):
    def create(self):
        self.set(border=p.border_all(1),
                 background=sp.Grey)
        r = p.Row(p.Text("Text1", rowspan=1),
                  p.Column(p.Image("jepplogo.jpg"),
                           p.Text("Text2"),
                           p.Text("Text22"),
                           p.Text("")))
        r.add(p.Column(p.Text("Text3"),
                       p.Text("Text4", rowspan=2),
                       p.Text("Text5")))
        self.add(r)
        self.add(p.Row(p.Image("jepplogo.jpg", colspan=2),
                       p.Text("Text6")))
```



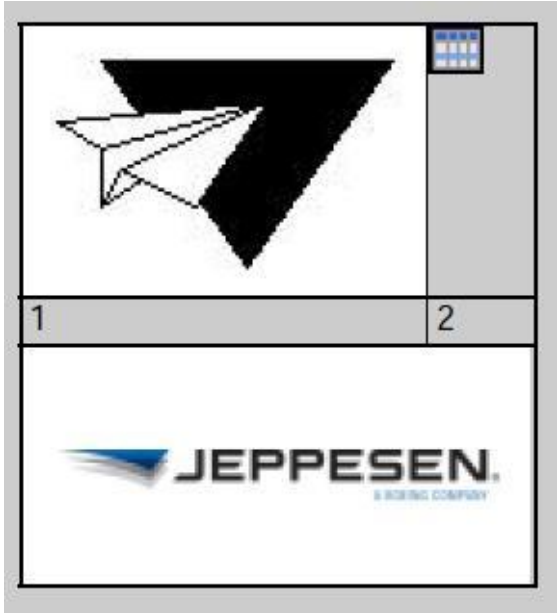
Distribution of ‘extra’ space

```
class Report(p.Report):

    def create(self):
        self.set(border=p.border_all(1),
                 background=sp.Grey)
        r = p.Row(p.Column(p.Image("fake_logo.jpg",
                           p.Text(1)),
                  p.Column(p.Image("table.jpg",
                           p.Text(2)))
        self.add(r)
        self.add(p.Image("jepplogo.jpg"))
```



Distribution of ‘extra’ space II

What to do if you want all the extra space in the right column?



We don’t

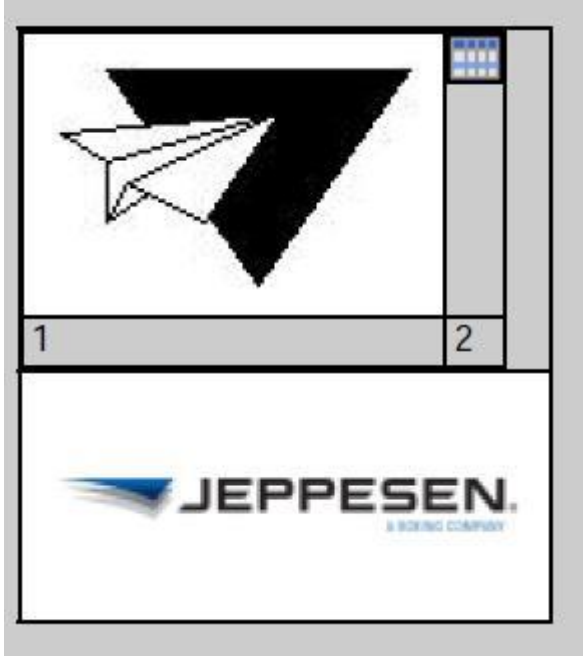
- have **springs** as in PDL
- know the **size** of objects involved

but we have Isolate...?!

Distribution of ‘extra’ space III

.. but Isolate does not do what we want:

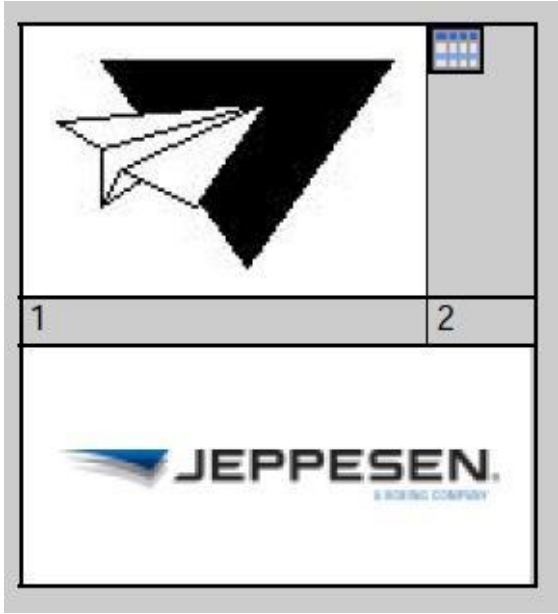
```
class Report(p.Report):

    def create(self):
        self.set(border=p.border_all(1),
                 background=sp.Grey)
        r = p.Row(p.Column(p.Image("fake_logo.jpg"),
                           p.Text(1)),
                  p.Column(p.Image("table.jpg"),
                           p.Text(2)),
                  border=p.border_all(1))
        self.add(p.Isolate(r))
        self.add(p.Image("jepplogo.jpg"))
```



Distribution of ‘extra’ space IV

Use “padding” objects

```
def Pad():
    return p.Text("",
                  rowspan=1,
                  padding=p.padding(0, 0, 0, 0))

class Report(p.Report):
    def create(self):
        self.set(background=sp.Grey)
        r = p.Row(p.Column(p.Image("fake_logo.jpg"),
                           p.Text(1),
                           border=p.border_all(1)),
                  p.Column(p.Image("table.jpg"),
                           p.Text(2)),
                  border=p.border(top=1, bottom=1, left=1,
                                  right=1, inner_floor=1))
        for _ in xrange(50):
            r.add(Pad())
        self.add(r)
        self.add(p.Image("jepplogo.jpg",
                         border=p.border_frame(1)))
```



Performance ☹

Exercise

Now it is time for exercise 2

PRT – Pages and cross-references

Let's look at:

- paper
- margins
- page breaks
- cross-references
- header / footer
- page width
- bookmarks

Paper

- Defined by the `Report` method:

`setpaper(orientation, size)`

orientation: `PORTRAIT` (default), `LANDSCAPE`

size: `A4` (default), `A5`...

```
...
class Report(p.Report):

    def create(self):
        self.setpaper(orientation=p.LANDSCAPE, size=p.A3)
        ...
```


Margins & page_width

- Defined by the `Report` property `margins`.
  - `padding()` is used as value.
- Default values (left =42, 36, 36, 36)
- Ignored in HTML.

Normally OK

- The `Report` method `page_width()` returns the available width considering margins and paper.

Page breaks

Remember:
`Report` is a subclass of `Column`.

Methods of `Column`:

The `Column` may be inside `Isolate` and other `Columns`, but not inside a `Row`.

`newpage()`	always page break
`page()`	page break if needed
`page0()`	page break if needed, lower priority than `page`.

`page` and `page0` are ignored in HTML.

You don't see any page breaks in the HTML browser, but they are considered when you print.

Cross-references

- Used as argument to `Text`
- Class: `Crossref(name, size, format)`

  `name` can be:

  - "current_page"
  - "last_page"
  - the name of an object

  `format:`

  - format string with the page (an integer) as optional argument
  - Example: `"page %d"`

Should not be used

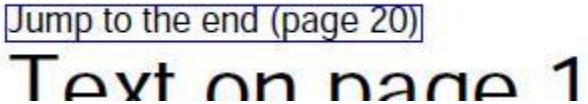
The property `name` defines the name of an object.

You get a clickable link.

Cross-references – simple example

```
import carmensystems.publisher.api as p

class Report(p.Report):
    def create(self):
        self.add(p.Text(p.Crossref("LAST",
                                   format="Jump to the end (page %d)"),
                        name="FIRST"))
        for i in xrange(20):
            if i:
                self.newpage()
            self.add(p.Text(p.Crossref("current_page", format="Text on page %d"),
                            font=p.font(size=25)))
        self.add(p.Text(p.Crossref("FIRST",
                                   format="Jump to the beginning"
                        name="LAST"))
```

pdf:



html:

Headers and Footers

- Appear on each page.
- Two types
  - for `Column`
    - Members of the **`Column`** from a layout perspective
  - for `Report`
    - Independent of the report body

Header / footer for `Column`

`Column` methods:

`add_header(box, column)`

`add_footer(box, column)`

The `Column` may not be inside a `Row`.

`box` : content

`column`: column this header/footer belongs to.
Default is `self`.

Example:

```
col.add_header(p.Text("My Header"))
```

Header / footer for `Report`

- Classes: `Header` and `Footer`
  - Independent of the rest of the report.
- Displayed at the top and the bottom of each **page**.
- How: `col.add(p.Header(…))`

The `Column` may not be inside a `Row`.

- You can **change** the header/footer in a report
  - Just make a new call to `add`. The last one "wins".

Example

```
class Report(p.Report):
    def create(self):
        self.setpaper(p.LANDSCAPE)
        self.set(font=p.font(size=30))
        rh = p.Row(p.Text("Report Header"),
                   p.Text("Page",
                          p.Crossref("current_page",
                                     format=" %i"),
                          p.Crossref("last_page",
                                     format=" (%i)"),
                          align=p.RIGHT))
        self.add(p.Header(rh, background=sp.LightGrey,
                          width=self.page_width()))
        self.add(p.Footer(p.Text("Report Footer", align=p.CENTER),
                          background=sp.LightBlue, width=self.page_width()))

        mc = self.add(p.Column(border=p.border_all()))
        mc.add_header(p.Row(p.Text("C-Header 1"), p.Text("C-Header 2"),
                            colour=sp.Red, border=p.border_all()))
        mc.add_footer(p.Text("Column Footer", colour=sp.Green,
                             align=p.CENTER, border=p.border_all()))

        for i in xrange(1, 40):
            mc.add(p.Row("data on 1 row %d" % i, "data on 2 row %d" % i))
            mc.page()
```

Report Header Page 2 (4)

C-Header 1	C-Header 2
data on 1 row 13	data on 2 row 13
data on 1 row 14	data on 2 row 14
data on 1 row 15	data on 2 row 15
data on 1 row 16	data on 2 row 16
data on 1 row 17	data on 2 row 17
data on 1 row 18	data on 2 row 18
data on 1 row 19	data on 2 row 19
data on 1 row 20	data on 2 row 20
data on 1 row 21	data on 2 row 21
data on 1 row 22	data on 2 row 22
data on 1 row 23	data on 2 row 23
data on 1 row 24	data on 2 row 24
Column Footer	

Report Footer

PDF

The `Report` (object) can **not** handle **both** column and page header/footer.

SlidePage1

Bookmarks

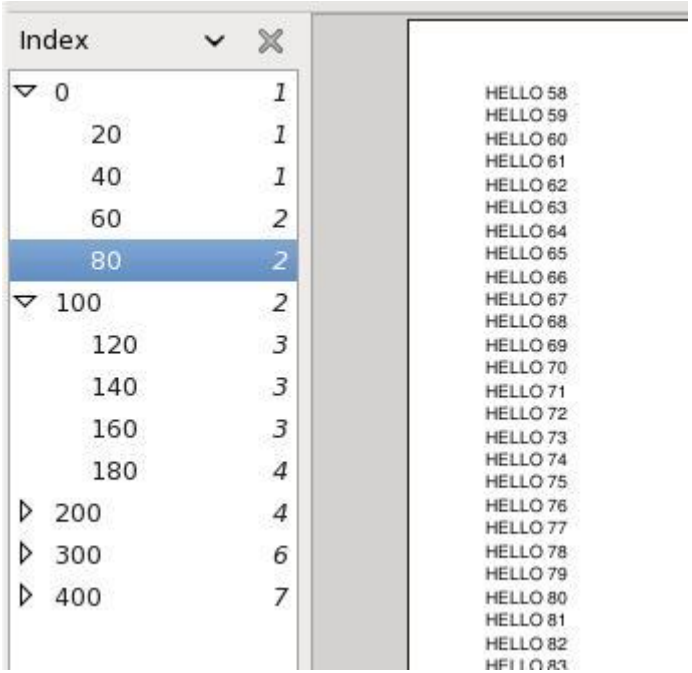
`bookmark` is a property of atomic objects

Value: `p.bookmark(text, level=1, open=True)`

- Only considered in PDF

Example:

```
import carmensystems.publisher.api as p

class Report(p.Report):
    def create(self):

        for i in xrange(500):
            self.page()
            t = self.add(p.Text('HELLO %s' % (i,)))
            if not i % 100:
                t.set(bookmark=p.bookmark(str(i), 1,
                                          i == 100))
            elif not i % 20:
                t.set(bookmark=p.bookmark(str(i), 2))
```



Exercise

Now it is time for exercise 3

Rave API – introduction

Slowly replacing *application specific APIs. E.g.* ***CuiCrc*** *in Studio.*

- History
  - v12: first version
  - v13: faster
  - v14: bag interface
- Requires a Rave enabled application

  (Studio, APC, Matador, Mave etc.)
- An object-oriented API that lets you:
  - investigate definitions of the loaded rule set
  - change settings
    - switch rules/constraints on/off
    - change parameter values
  - calculate values considering data.
- Module: `carmensystems.rave.api`

This section – basics

- Documentation.
- Python classes for definitions in the Rule Set.
- Python classes for Rave Expressions.
- Data types in Rave/Python.
- Evaluate constant values using `eval`.
- Iterate over Rave definitions.

Documentation

- **API**

  Help> API Documentation> Rave Python API

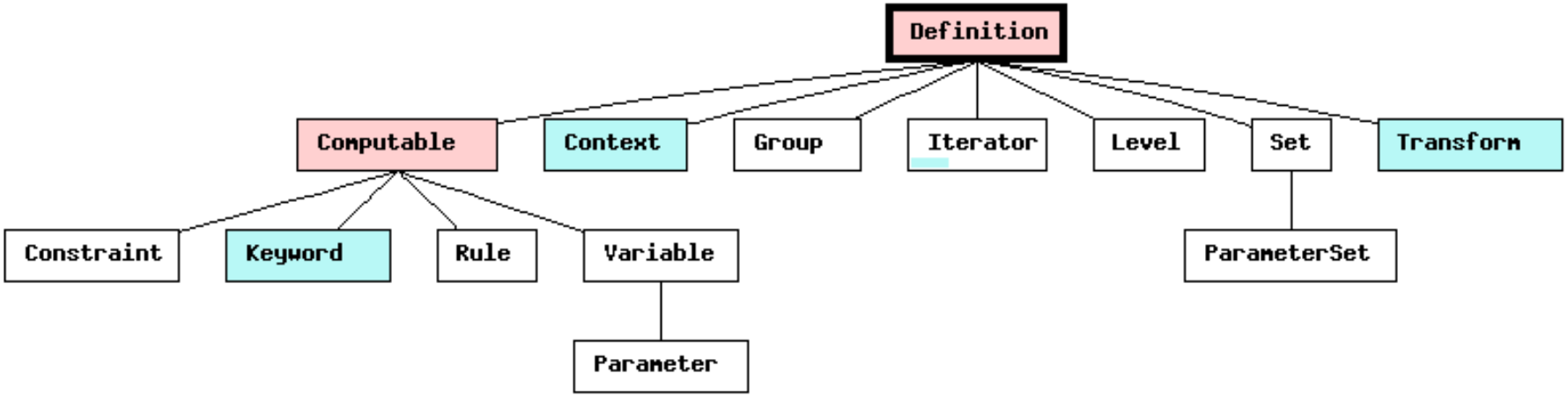
- **General**

  Development> Developer Guide> Modelling with Python> Rave module

  in *System Help*

Python classes for definitions

The Python Rave API contains one class for each type of definition in Rave



Defined by the application (Studio)

Base class - no instances

Classes and factory functions

Definition Objects and functions for creation

```
Constraint   : constraint(name)
Context      : context(name)
               selected(level)
               buffer2context(Buffer)
Group        : group(name)
Iterator     : iterator(name)
Keyword      : keyw(name)
Level        : level(name)
Parameter    : param(name)
ParameterSet: paramset(name)
Rule         : rule(name)
Set          : set(name)
Transform    : transform(name)
Variable     : var(name)
```


Python classes for definitions – some methods

All Rave definitions are available in the API – not only the exported ones.

All classes:

- `name`, `remark`

Most classes:

- `level` Dependency. Returns a `Level` object

`Set` and `Group`:

- `members` ← **Does not work for external sets.**

`Rule` and `Constraint`:

- `on`, `setswitch`

`Parameter`:

- `value`, `setvalue`

`ParameterSet`:

- `clear`, `add`, `remove`

See the API documentation for a complete list.

Python classes for definitions – example

```
import carmensystems.rave.api as r

# Toggle a rule on/off
arule = r.rule("trip_rules_exp.exp_no_middle_passive_legs")
arule.setswitch(not arule.on())

# Toggle a boolean parameter True/False
aparam = r.param("studio_config.rudob_show_ac_change")
aparam.setvalue(not aparam.value())

# Print the name of the dependency level for a variable
print r.var("crg_hotel.earliest_check_in").level().name()
```

`levels.duty` is written to the log file.

dependency

SlideRaveB1.py

Python class for Rave expressions

- The API supports Rave expressions
- The class `expr` is used

  One public attribute, `code`

- The *Rave Interpreter* is used to evaluate expressions
  - Slower than compiled Rave code
  - Limited. Not supported:
    `traversers`, `aggregates`, `contexts`, `transforms`, `sets`, `index`
- Example:

```
import carmensystems.rave.api as r
ex = r.expr("arrival-departure")
print ex.code
print ex
print type(ex)
```

In the log file:

```
arrival-departure
expr(arrival-departure)
<type 'api.expr'>
```

Data types

The data types in Rave have corresponding data types in Python:

Rave	Python	
`Int`	`int`	`built-in`
`Bool`	`bool`	`built-in`
`String`	`string`	`built-in`
`Abstime`	`AbsTime`	`AbsTime`
`Reltime`	`RelTime`	`RelTime`
`Enum`	`enumval`	`carmensystems.rave.api`

The classes `AbsTime` and `RelTime`

- Basic data types in Jeppesen applications exported to Python.
- Found in the modules `AbsTime` and `RelTime`
  - Also found in the modules:
    `carmensystems.basics.abstime`
    `carmensystems.basics.reltime`
- Defined in the module `BSIRAP.py`.
- Useful methods:

`split()`	A `tuple` ([year, mon, day,] hour, min)
`__str__()`	Non-localized string format
`getRep()`	Integer

Localized date (+ time) string:

```
a = AbsTime()
Dates.FDatInt2Date(a.getRep())
Dates.FDatInt2DateTime(a.getRep())
```

Localized "now":

```
Dates.FDatInt2DateTime(Dates.FDatUnix2CarmTimeLT(time.time()))
```

- See the manual pages, *`AbsTime(3)`* and *`RelTime(3)`* for details.

Note:
The manual pages describe the corresponding **C++** classes.

Python class `enumval` – example

Rave code (in module `crg_basic`):

```
enum enumeration_constants =
    max_roster;
    production;
    reserve;
end
%enum_constant% = parameter production;
```

It is **not** possible to ask for the set of all allowed values of an `enum`.

Python code:

```
p = r.param("crg_basic.enum_constant")
print type(p.value())
p.setvalue(r.enumval("crg_basic.max_roster"))
print p.value()
```

In the log file:

```
<type 'api.enumval'>
crg_basic.max_roster
```

Evaluation with `eval`

`eval([bag,] *args)`

- Evaluates the arguments in a given data set, specified by the bag.
- Produces a `tuple`.
- Parameters:
  - `bag`
    - when constants are evaluated, this argument is not needed.
  - `*args` can be:
    - computable definitions (object or name)
    - expressions (object or string)
    - traversers: `first`, `last`
    - `foreach`

Next section

before v14

Evaluation – example

```
import carmensystems.rave.api as r
res = r.eval('rule_set_name', '9:00 - 0:01')
print res
print [str(r) for r in res]
print [type(r) for r in res]
```

Constant Rave definitions → No **bag** is needed.

In the log file:

```
('Pairing',  <C RelTime instance at _508.. >)
['Pairing', '8:59']
[<type 'str'>, <class 'BSIRAP.RelTime'>]
```

`Abstime` and `Reltime` in Rave give `BSIRAP` types from the API.

Get all objects in the rule set

There is an iterator for each kind of definition in Rave:

```
variables(), parameters(), modules(), sets(), keywords(),
constraints() , contexts(), transforms(), levels(), rules(),
groups(), iterators()
```

Get all objects in the rule set – example

Print the 20 longest remarks of rules and parameters in the loaded rule set to the log file.

```
import carmensystems.rave.api as r

for it in sorted(list(r.parameters()) + list(r.rules()),
                 key=lambda a: -len(a.remark() or a.name()))[:20]:
    print "%s '%s':\n %s '%s'" % (it.__class__.__name__,
                                  it.name(),
                                  " " * 6,
                                  it.remark())
```

Log file:

```
Rule 'trip_rules_exp.exp_duty_max_nr_of_passive_legs':
        '(EXP) Max nr of passive(OAG, deadhead or ground) legs in a duty (CAN be broken by APC)'
Parameter 'tor.tor_general_deadhead_instead_of_onduty_ok':
        'TOR.general.1: Leg changed from onduty to deadhead is considered as an identical leg'
```

SlideRaveB2.py

Outside the scope of the API

There are no functions in the Rave API for saving or loading:

- rule set
- parameter sets.

In Studio there are functions in the `Cui` module for these purposes.

Covered in “Python in Studio”.

Exercise

Now it is time for exercise 4

Consider data in calculations

Recap Rave and some new terms

- `Level`, ledob, bag
- `Dependency`
- `Context`
- `Iterator`

`Bag` in Python

Rave iterator in Python

Simple report

Level Dependency requirements in the API

Level

(should be well known from the Rave course)

- You define them in the Rave code.
- Often in a rule set: `leg`, `duty`, `trip`, `wop`, `roster`
  - Normally found in the Rave module `levels`

Example:

```
global export level duty =
    is_last(leg)
    when(%levels_leg_is_last_in_duty%);
end
```

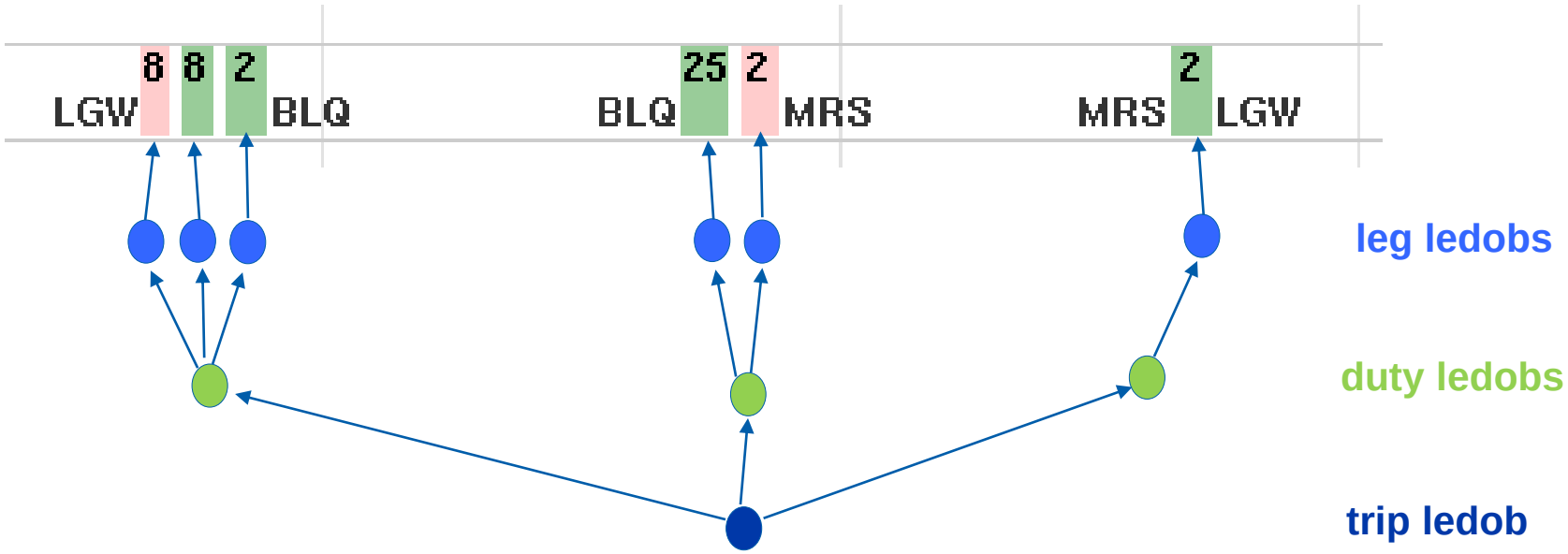
Predefined level objects in the API:
`Level.atom(), Level.chain(), Level.const()`

Deprecated synonyms:
`atomlevel chainlevel constlevel`

Ledob

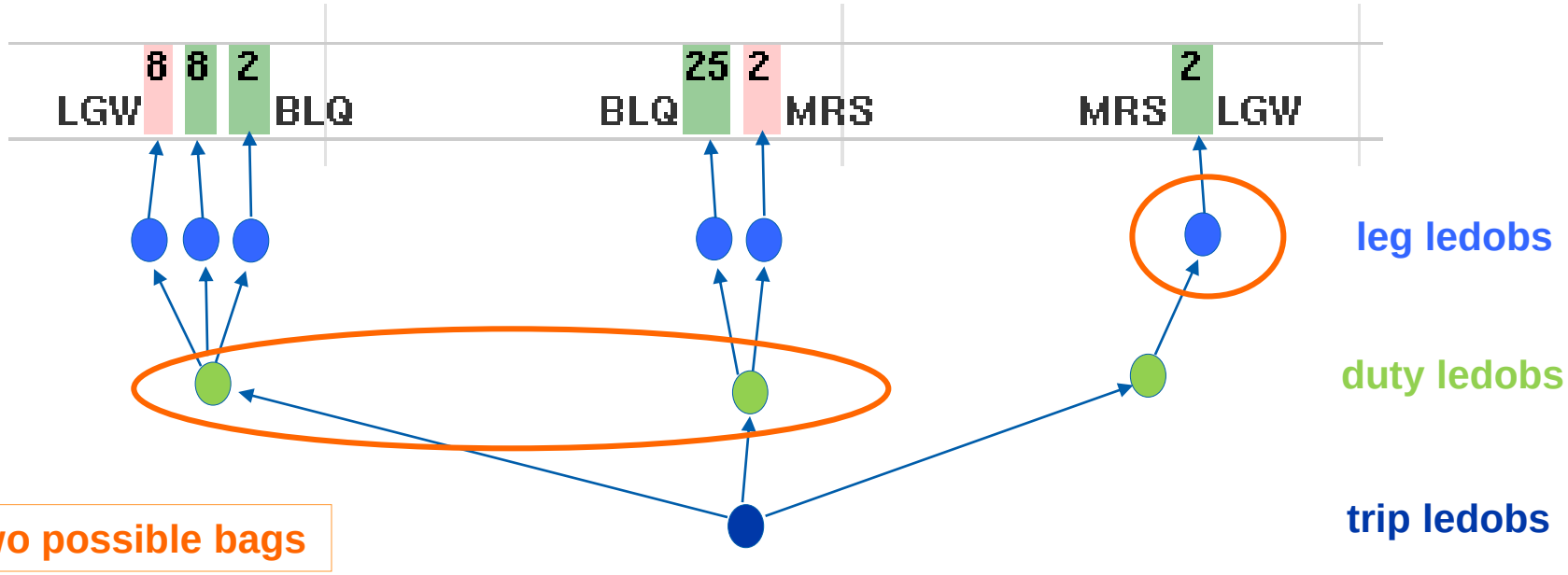
"**l**evel-**de**fined-**ob**ject".

- Called **planning object** in the API documentation.
- Example:



Bag

A **bag** contains a **set of ledobs** (having the same level).



Dependency

Dependency

Derived by Rave for each definition.

Examples:

- Duty dependant variable:

```
%layover_station% = last(leg(duty), arrival_airport_name);
```

- Atom/leg dependant keyword:

```
crr_name
```

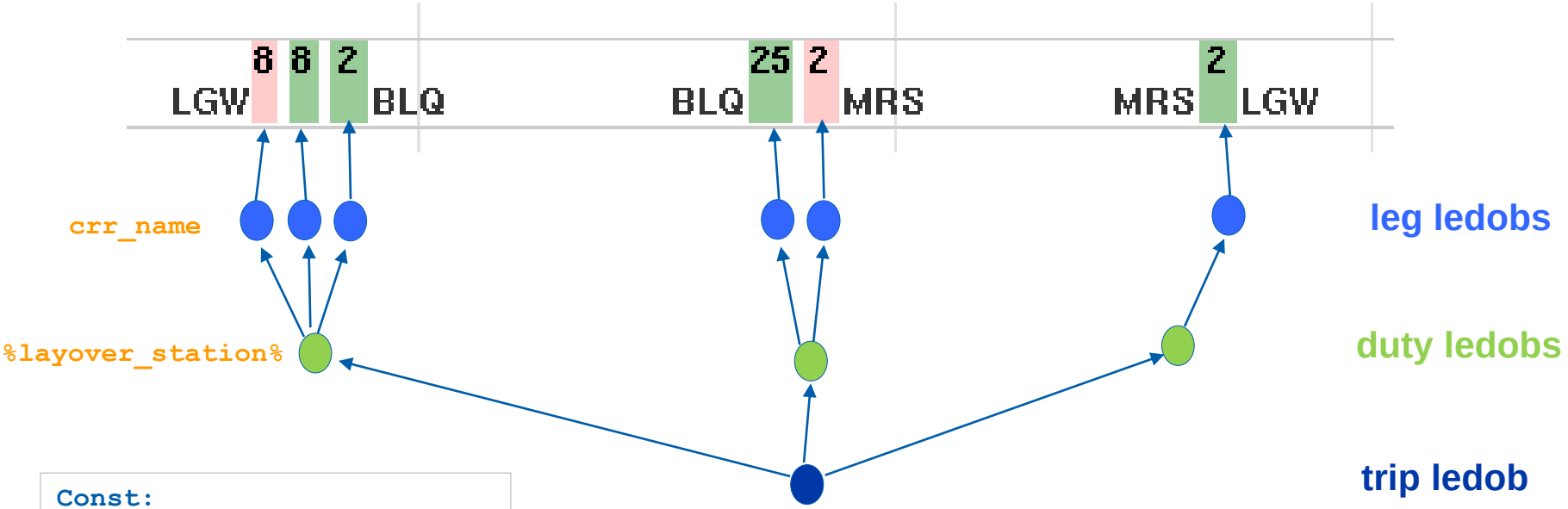
Dependency of Keywords may be:
atom (leg), **chain** or **const**

- Constant dependency:

```
%leg_cost_dh_hour% = parameter 10000;
```

Dependency – Graph

Values are calculated for ledobs with the correct dependency.



Const:

%leg_cost_dh_hour%

Contexts – recap

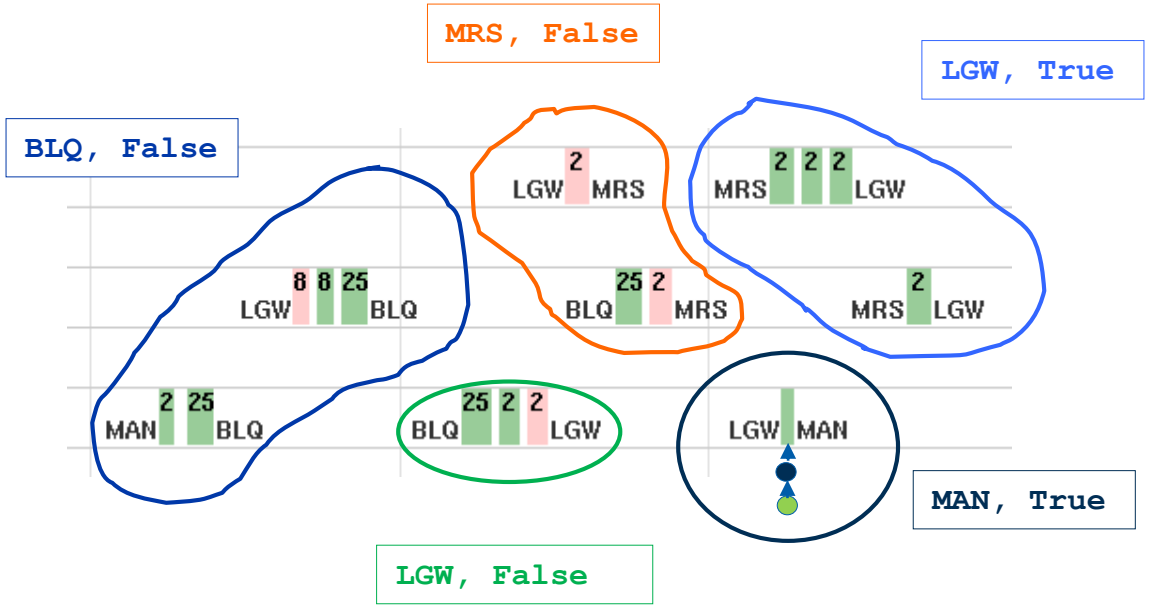
- A context **normally** contains a set of **chain** ledobs
- The contexts are defined by the application that enables Rave.
    - You **cannot** define your own contexts.
- Run ***Help> Keywords etc.*** to get a complete list.
- Some often used Studio contexts:
    - `ac_rotations`
    - `lp_activity`
    - `sp_crrs`
    - `sp_crew_chains`
    - `sp_crew`
    - `sp_free_legs`

Iterators – recap

- A Rave **iterator** considers all ledobs in the current bag and puts them into a number of new **bags**.
- There are some predefined iterators in CARMSYS, but you create most of them yourself in Rave.

Iterators recap – example

```
export iterator layover_set =
    partition(duty)
    by(duty.%end_station%, duty.%is_last_in_trip%);
end
```



Note:
We skip the drawing of the ledobs, but they are still there.

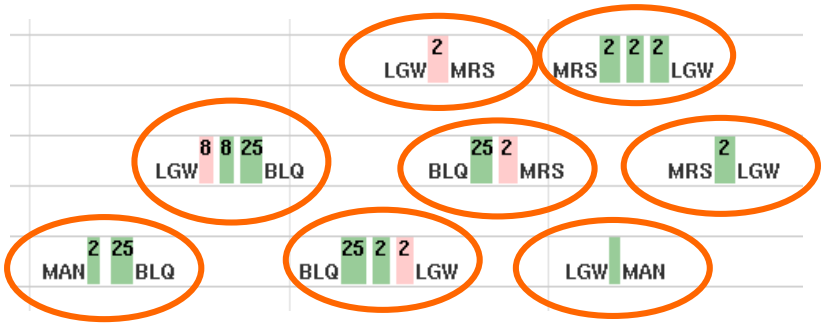
Iterators – recap more

One ledob per bag.

- A rule set normally contains one atomic iterator per level
  e.g.: `leg_set`, `duty_set`, `trip_set`, `wop_set` and `roster_set`.

```
global export iterator duty_set =
    partition(duty);
end
```

Predefined atomic iterators in Rave:
`chain_set`
`atom_set`.



Iterators – recap more

You can use iterators in Rave definitions. Examples:

Common

```
%num_duties_in_bag% = count(duty_set);
```

8

Dependency is **const**.

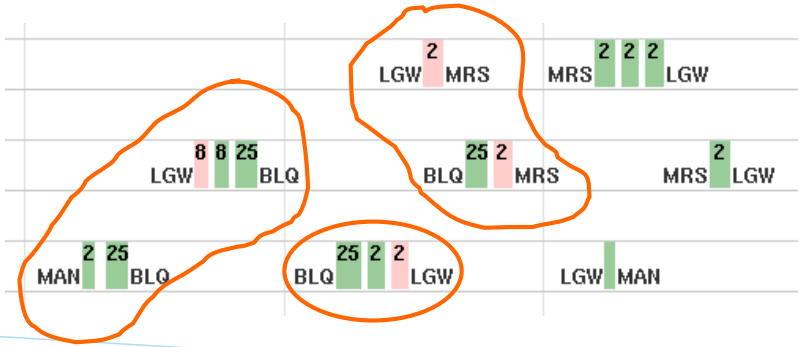
The value depends on the "active" bag, but not the "active" chain.

Seldom useful

```
%num_layover_stations% = count(layover_set) where(not duty.%is_last_in_trip%);
```

3

The number of created **bags** is counted.



End of recap

- Level
- Ledob
- Bag
- Dependency
- Context
- Iterator

`Bag` in Python

New in v14

- A `Bag` is an object which:
  - contains a number of ledobs
  - can be used as first argument to `eval`.
  - has **callable attributes** corresponding to all: `keywords`, `variables`, `rules`, `constraints`, `iterators`, `parameters` and `transforms` in the loaded Rule set.

As we have seen

Described some pages down in the main section of the Rave API documentation

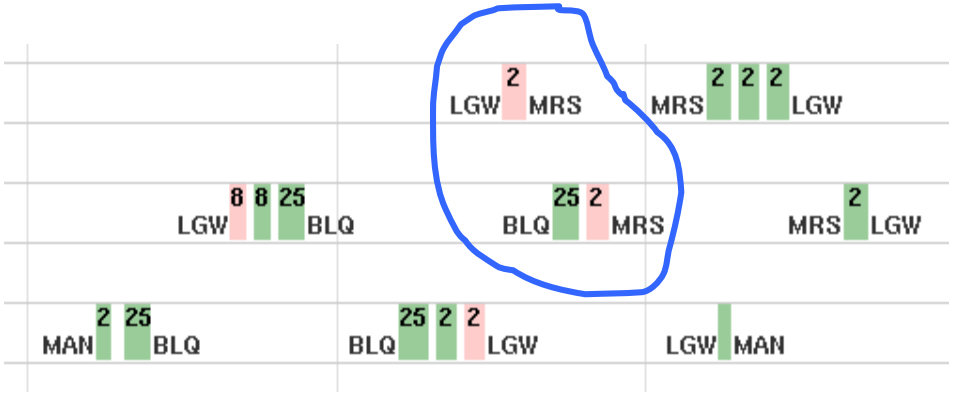
Bag in Python – example

Rave module

Rave variable

```
bag.duty.end_station()
"MRS"

r.eval(bag, "duty.end_station")
("MRS",)

bag.iterators.num_duties_in_bag()
2
```

Assume we have a bag with two **duty-ledobs** ending at the same airport.

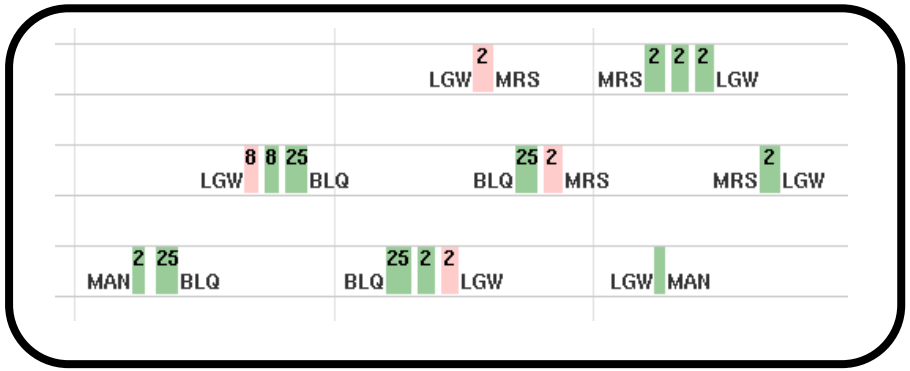


Create `Bag` – from `Context`

- Initially you must create a `Bag` from a `Context`.
- Use the `bag` method.
  - It returns a `Bag` containing chain-ledobs for all chains in the context.

Example:

```
cbag = r.context("sp_crrs").bag()
```

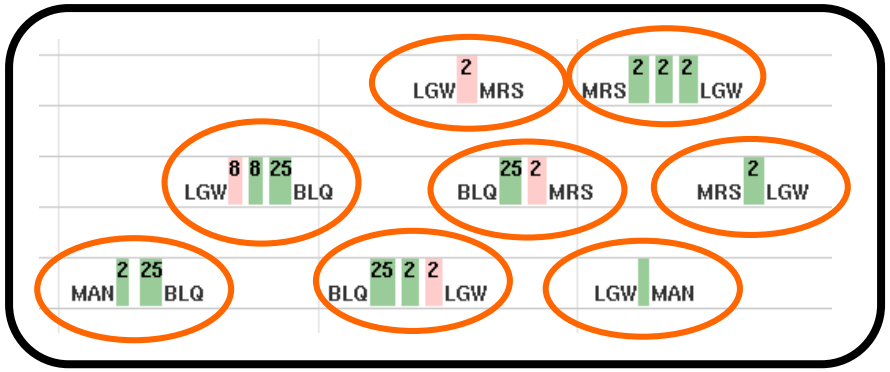


The sub-plan `(weekly/ThreeTrips)` only contains these trips.

Use the bag

```
cbag = r.context("sp_crrs").bag()
cbag.iterators.num_duties_in_bag()
8
r.eval(cbag, "iterators.num_duties_in_bag")[0]
8
r.eval("sp_crrs", "iterators.num_duties_in_bag")[0]
8
```

The `Context` object is created and the `bag` method is called automatically by `eval`.



Create Bag – using iterators

As we have seen

Bag has a method for each Rave iterator

- The methods create Python iterators, which you can use in for loops.
- In each loop you get a new Bag from the iterator.

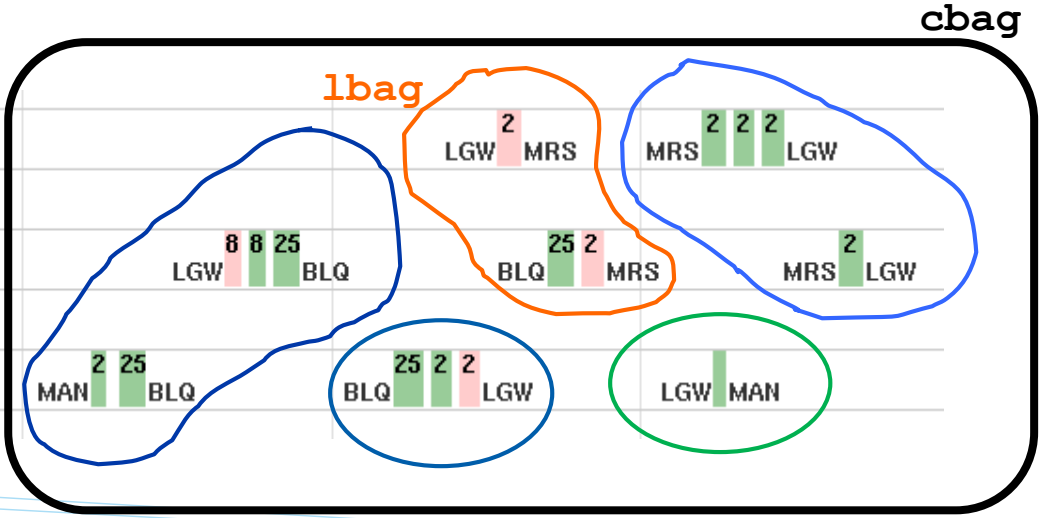
Example:

```
cbag = r.context("sp_crrs").bag()
for lbag in cbag.iterators.layover_set():
    print lbag.iterators.num_duties_in_bag()
```

Note
Bags from iterators are implemented by changing **a global state**. For these:
- There are no real Bag objects in Python.
- You can only use the "active" Bag.

In the log file

2
2
2
1
1



Bag iterators – where and sort_by

```
cbag = r.context("sp_crrs").bag()
for lbag in cbag.iterators.layover_set(where="not duty.%is_last_in_trip%",
                                       sort_by="duty.end_station"):

    print lbag.iterators.num_duties_in_bag(),
    print lbag.duty.end_station()
```

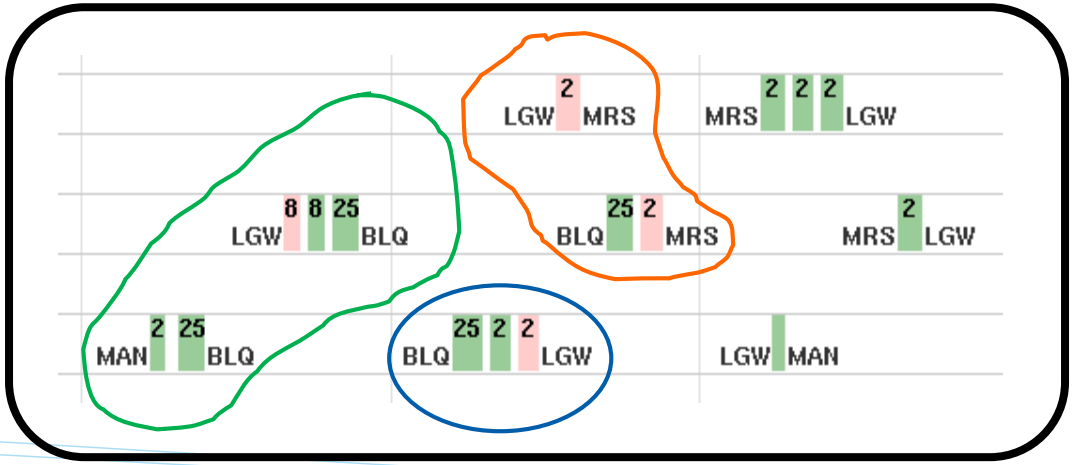
May be one **single** or a **tuple** of **variable/ keyword / expression** (**string** or **object**)

Examples:

```
("deadhead", "leg.%in_pp%")
"not deadhead"
r.keyw("deadhead")
```

In the log file

2 BLQ
1 LGW
2 MRS



SlideRaveD1.py

The same – before v14

- Values from all iterator bags in one call
- `Iterator` as argument to the functions `foreach` and `iter`.

Example:

```
res = r.eval('sp_crrs',
             r.foreach(r.iter('iterators.layover_set',
                              where="not duty.%is_last_in_trip%",
                              sort_by="duty.end_station"),
                       'iterators.num_duties_in_bag',
                       'duty.end_station'))
self.add(res)
```

`eval` produces a tuple, `foreach` a list of tuples.

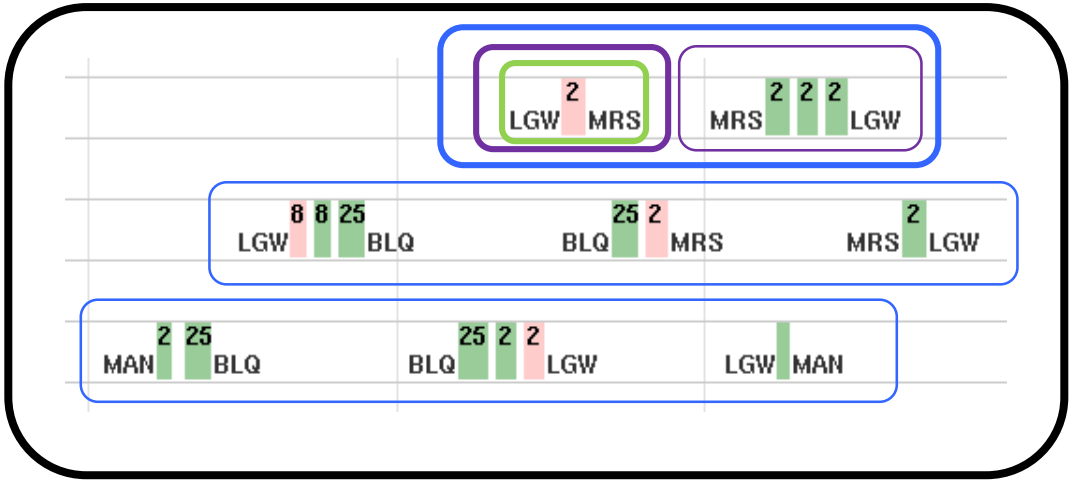
([(2, 2, 'BLQ'), (3, 1, 'LGW'), (1, 2, 'MRS')],)

index number – useless

SlideRaveD4.py (report)

Nested iterators – example

```
class Report(p.Report):
    def create(self):
        cbag = r.context('sp_crrs').bag()
        for trip_bag in cbag.iterators.trip_set():
            for duty_bag in trip_bag.iterators.duty_set():
                for leg_bag in duty_bag.iterators.leg_set():
                    …
```



SlideRaveD3.py

Example – simple trip report

```
class Report(p.Report):

    def create(self):

        cntx = r.context('sp_crrs')
        for trip_bag in cntx.bag().iterators.trip_set():
            dc = p.Column()
            self.add(p.Row(trip_bag.trip.name(),
                           dc,
                           border=p.border_frame()))
            for duty_bag in trip_bag.iterators.duty_set():
                dr = dc.add(p.Row(duty_bag.duty.start_station()))
                for leg_bag in duty_bag.iterators.leg_set():
                    dr.add(p.Column(leg_bag.flight_number(),
                                    leg_bag.arrival() - leg_bag.departure()))
                    dr.add(p.Column(leg_bag.arrival_airport_name()))
```

A1	LGW	2796	MRS				
		1:50					
	MRS	2795	LGW	2940	EDI	2941	LGW
		1:50		1:25		1:25	
A2	LGW	2762	AMS	2763	LGW	2564	BLQ
		1:10		1:10		2:05	
	BLQ	2563	LGW	2798	MRS		
		2:10		1:50			
	MRS	2799	LGW				
		1:50					
B1	MAN	2901	LGW	2560	BLQ		
		1:10		2:05			
	BLQ	2559	LGW	2760	AMS	2761	LGW
		2:15		1:15		1:05	
	LGW	2902	MAN				
		1:00					

add returns the argument.

SlideRaveD5.py

Dependency checking

The Rave API has a rather strict dependency checking

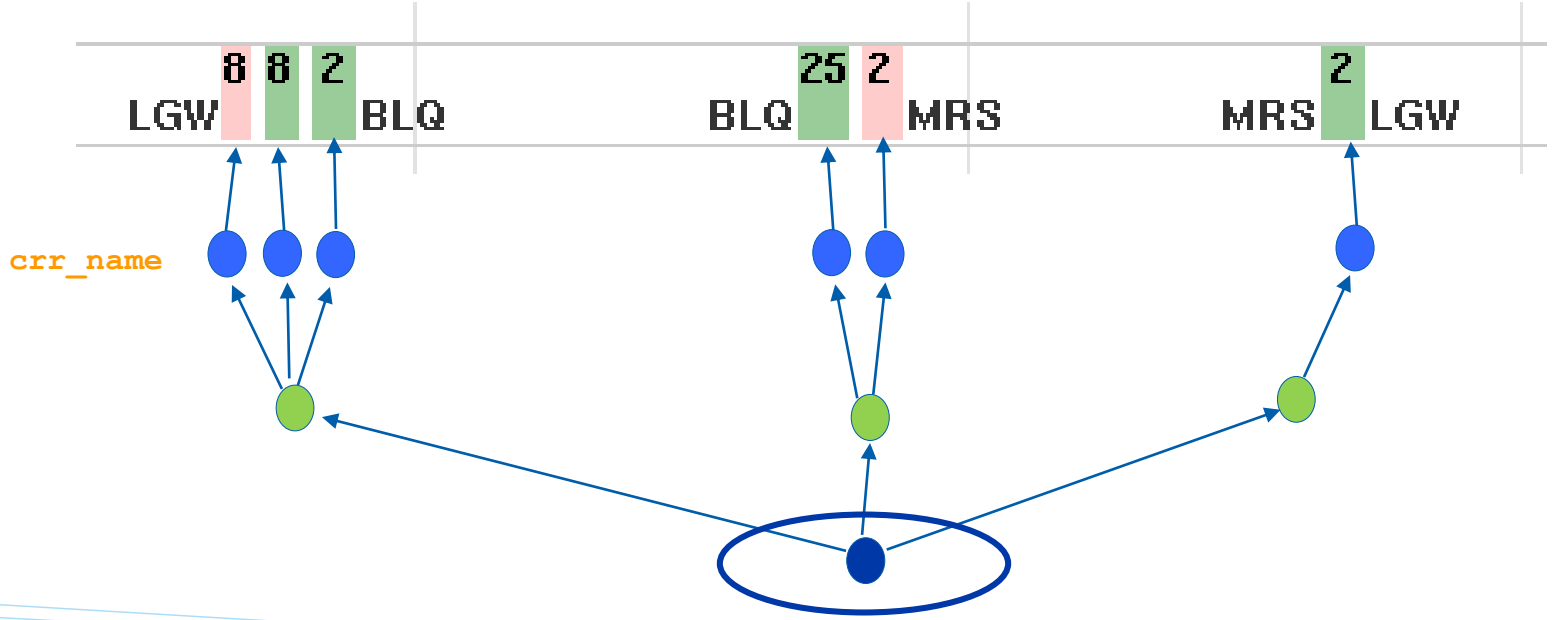
- Rave definitions you evaluate must have the **same or "bigger" dependency** than the Ledobs in a bag.
  - Example:
    You can't ask for a leg dependent value with duty-ledobs in the bag.

- Normally a `bag` from a `Context` **only** supports evaluation of Rave definitions with `const` dependency.

Dependency checking – example (1)

```
import carmensystems.rave.api as r
for trip_bag in r.context("sp_crrs").bag().iterators.trip_set():
    print trip_bag.crr_name()
```

In the logfile:
api.UsageError: 'crr_name' must be computed on level levels.leg or lower (now applied on level levels.trip)



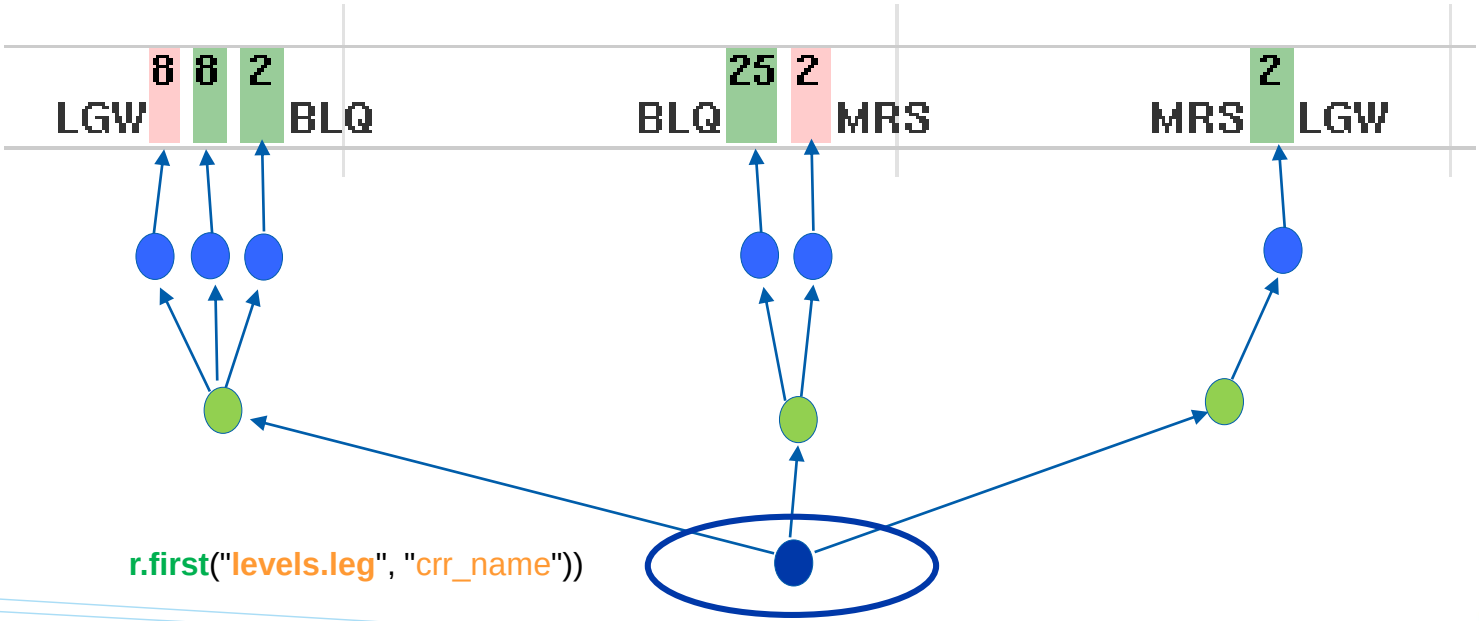
SlideRaveD6.py

Dependency checking – example (2)

```
import carmensystems.rave.api as r
for trip_bag in r.context("sp_crrs").bag().iterators.trip_set():
    # print trip_bag.crr_name()
    print r.eval(trip_bag, r.first("levels.leg", "crr_name"))[0]
```

Gets the same dependency as the ledobs in the current bag.

Traversers in the API: `first` and `last`



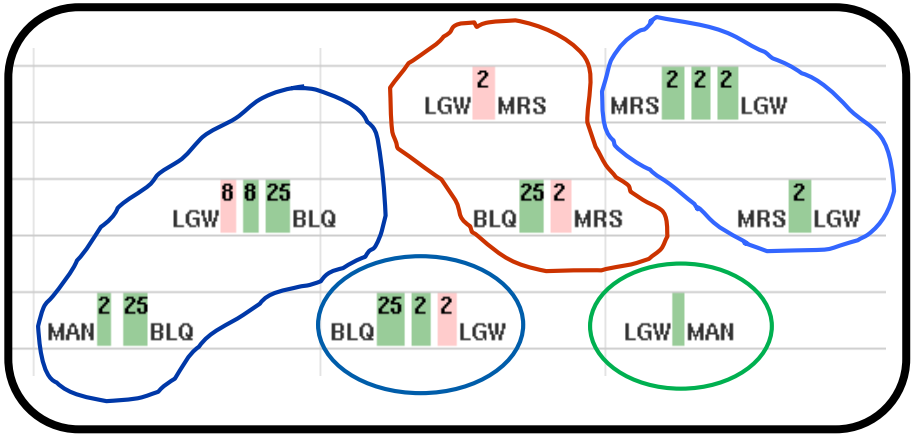
Dependency – warning

- The dependency checking does not guarantee that you get deterministic answers.

Example:

```
cbag = r.context("sp_crrs").bag()
for lbag in cbag.iterators.layover_set():
    print lbag.duty.start_station(),
    print lbag.duty.num_legs()
```

LGW 1
MRS 3
LGW 3
BLQ 3
LGW 1



One ledob in the bag is picked **randomly.**

Exercise

Now it is time for exercise 5

More about Contexts

We will take a look at:

- More interesting contexts:
  - other useful Rave contexts
  - context for current object
  - contexts from buffers
- Best Practice:
  - select-and-operate
  - standard

Other useful Rave contexts

```
default_context
marked_in_window_main
marked_in_window_left
```

default_context

- The content of `default_context` depends on how you start the report:
  - **General pop-up**: Everything in the window.
  - **Object pop-up**: The chain you point at.
  - **Plan menu**: Pairing system - All trips in the sub-plan.
    Rostering system - All rosters in the sub-plan.

Note:

You must initialize `default_context` yourself, if you use it in the Rave API "outside" report generation.

How you do this is described in the course **Python in Studio**.

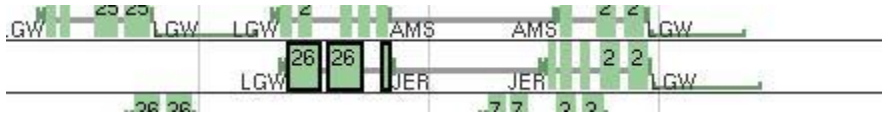
Rave API – more about contexts

marked_in_window_main

Contains **leg-ledobs.**

All other contexts we have seen contain **chain-ledobs**.

Why should you care?



Example

```
class Report(p.Report):

    def create(self):
        cbag = r.context("marked_in_window_main").bag()
        self.add(p.Row("Num legs 1:",
                       cbag.iterators.num_legs_in_bag()))
        self.add(p.Row("Num legs 2:",
                       sum(tbag.trip.num_legs()
                           for tbag
                           in cbag.iterators.trip_set())))
```

Num legs 1: 3
Num legs 2: 8

A `leg` level dependant iterator is used.

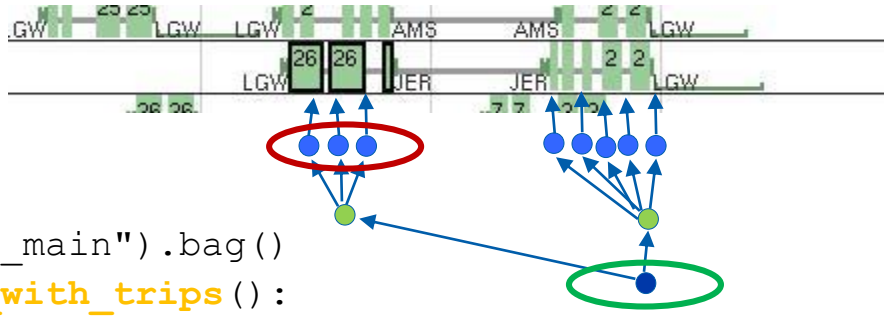
A `trip` level dependant iterator is used.

To avoid problems, put ledobs with the correct level in the ”top bag”.

Crew/SlideContext1.py

Correct ledob-level in the ”top bag” – How?

Example




```
class Report(p.Report):

    def create(self):
        cbag = r.context("marked_in_window_main").bag()
        for topbag in cbag.sno.single_bag_with_trips():

            self.add(p.Row("Num legs 1:",
                           topbag.iterators.num_legs_in_bag()))
            self.add(p.Row("Num legs 2:",
                           sum(tbag.trip.num_legs()
                               for tbag
                               in topbag.iterators.trip_set())))
```

Num legs 1: 8
Num legs 2: 8

```
iterator single_bag_with_trips =
    partition(trip)
    by(true);
end
```

Crew/SlideContext2.py

Current object

Not all marked

Task:

Consider only the **single** object (e.g. leg) the user has generated the report from?

Problem:

`default_context` from “object popup” contains a **chain** ledob. There is no keyword for the current object.

Solution:

`r.selected(level)`

returns a `Context` with **one** ledob of the specified level.

Note:

Rave "knows" that there is always just one ledob in the bag.

A `bag` from a `Context` created with `selected` supports calculations of Rave definitions with **not** `const` dependency.

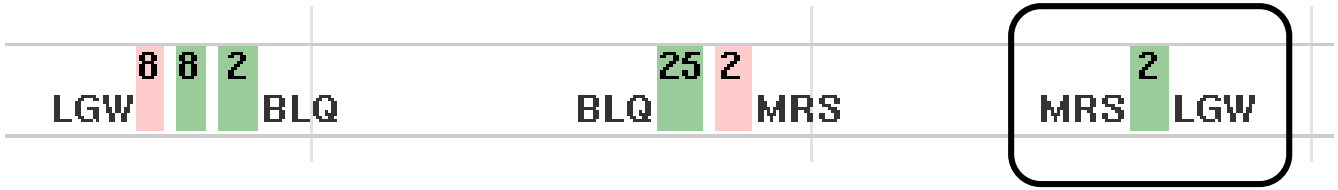
Current object – example

Show the flight number of the current leg.

Run the report from object pop-up menu.

```
import carmensystems.publisher.api as p
import carmensystems.rave.api as r

class Report(p.Report):
    def create(self):
        cbag = r.selected("levels.leg").bag()
        self.add(p.Text(cbag.flight_number()))
```

You must specify the module. **`global export`** is only considered in the Rave code.



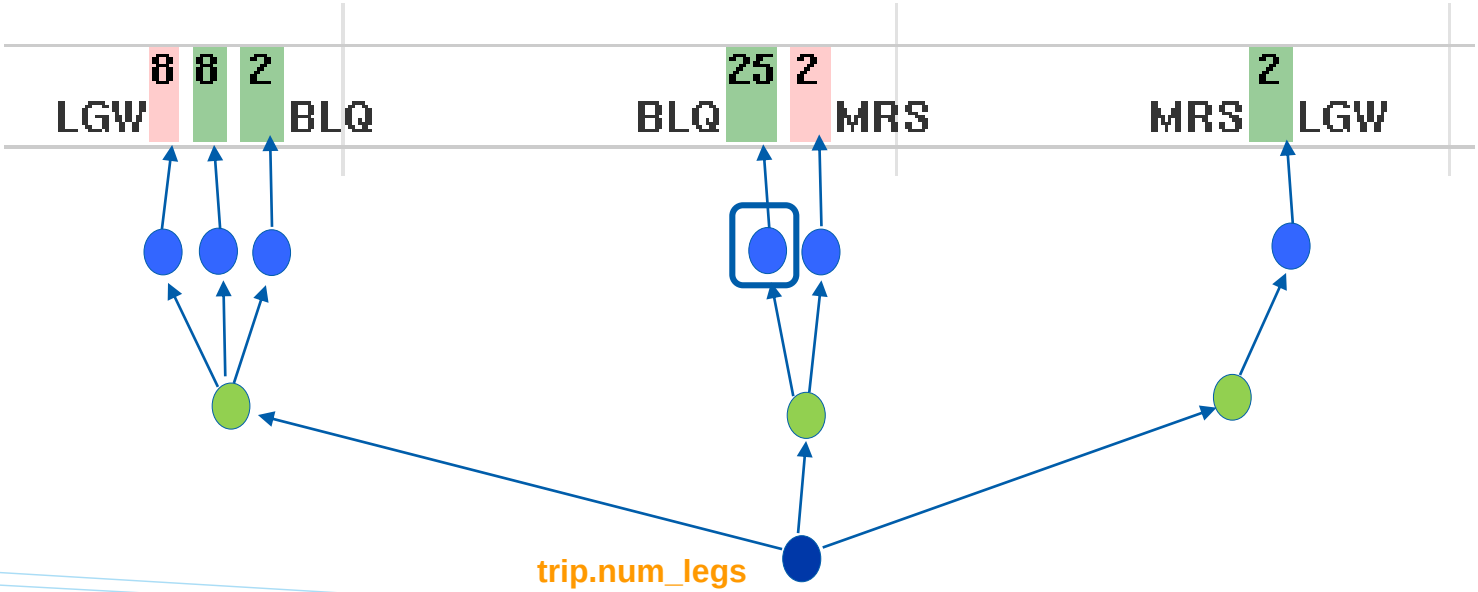
crr_object/SlideContext3.py

Note – dependency “bigger" than the ledob

The **entire chain** is always considered in calculations of level-dependent Rave definitions, no matter the kind of ledobs in the bag. Example:

```
class Report(p.Report):

    def create(self):
        sbag = r.selected("levels.leg").bag()
        self.add(sbag.trip.num_legs())
```

A trip-dependent variable:
`%num_legs% = count(leg(trip));`



6

SlideContext4.py

Context from Buffer

We have seen two ways to create a `Context` in Python

```
r.context("sp_crrs")
r.selected("levels.leg")
```

There is one more:

```
r.buffer2context(buffer)
```

- `buffer` is a `CuiBuffer` or a `CpmBuffer`

Cpm/Cui buffers

- `CpmBuffer` and `CuiBuffer`:
  - Found in the Python modules:

    `carmensystems.studio.cpmbuffer`

    `carmensystems.studio.cuibuffer`
  - `CuiBuffer` is a subclass of `CpmBuffer`
- A buffer contains a number of ledobs (any level).
- Documentation:

System Help> Development> Developer Guide> Modelling with Python> Studio modules>
carmensystems.studio.cpmbuffer
carmensystems.studio.cuibuffer

Create a Cpm buffer

```
CpmBuffer()
CpmBuffer(buffer)
CpmBuffer(plan_constant)
```

Leg sets

Rotations

Some of the constants: `SUB_PLAN`, `REF_PLAN_1`, `LOCAL_PLAN_CFC`, `LOCAL_PLAN_CARC`

```
CpmBuffer(buffer, rave_expr)
```

Must be a string

Example:

```
import carmensystems.studio.cpmbuffer as cpmb
import carmensystems.rave.api as r

buf = cpmb.CpmBuffer(cpmb.SUB_PLAN)
cbag = r.buffer2context(buf).bag()
print cbag.iterators.num_chains_in_bag()
```

SlideContext5.py

Create a Cui buffer

CuiBuffer(area, scope)

Useful constants for **area**:

CuiWhichArea

CuiArea0..3

Useful constants for **scope**:

ObjectScope ← The **complete** ”current” chain

MarkedScope ← All marked **legs** in the active window

WindowScope

Example. Number of chains in the active window:

```
import carmensystems.studio.cuibuffer as cuib
import carmensystems.rave.api as r

buf = cuib.CuiBuffer(cuib.CuiWhichArea, cuib.WindowScope)
cbag = r.buffer2context(buf).bag()
print cbag.iterators.num_chains_in_bag()
```

SlideContext6.py

Cpm/Cui buffers – modify

There are some methods which modify the content of a buffer:

```
fill(buffer, rave_expr)
fill(buffer)
fillAnyLevel(buffer, rave_expr)
empty()
```

Must be level chain

Example:

```
import carmensystems.studio.cpmbuffer as cpmb
import carmensystems.rave.api as r

buf1 = cpmb.CpmBuffer(cpmb.SUB_PLAN)
buf2 = cpmb.CpmBuffer()
buf2.fill(buf1, "is_crr")
cbag = r.buffer2context(buf2).bag()
print cbag.iterators.num_chains_in_bag()
```

Chain dependant Rave expression for kind of chain in sub-plan:

Trip : `is_crr`

Leg: `is_free_leg`

Roster: `not void(crr_crew_id)`

Duty: `not (is_crr or is_free_leg or not void(crr_crew_id))`

Cpm/Cui buffers – good to know

There are bugs and traps. Advice to avoid problems:

- **do not modify** a buffer you have created a context for
- **do not change the plan** while a buffer is used
- put ledobs with the **correct level** in the “top-bag”
- keep **your own reference** to the buffer as long as the context is used

OK:

```
buf = cuib.CuiBuffer(cuib.CuiArea1, cuib.WindowScope)
cntx = r.buffer2context(buf)
```

not OK:

```
cntx = r.buffer2context(cuib.CuiBuffer(cuib.CuiArea1, cuib.WindowScope))
```

DnD – best practice

Main window reports should:

- be generated from the **object** menu
- only consider **fully selected** objects
  - exception for roster reports – one selected leg in the roster is enough
- contain a warning if there are (ignored) partly selected objects in the window.

Plan reports should:

- be generated from the **Planning tools** menu
- consider **all objects in the plan**.

Left window reports should:

- only consider chains selected in the left window.

Dynamic reports should:

Covered in PRT2

- only consider the current object

Standard – best practice

Main & Left window reports should contain information about

- the **current** object when generated from the **object** menu
- **all** objects in the window when generated from the **general** menu.

Plan reports & **Dynamic** reports:

- same as for DnD

Best practice – how to implement I

It is complicated to get it right. Example: a Trip report

Select-and-operate

In `crr_window_object`

```
import carmensystems.publisher.api as p
import carmensystems.rave.api as r
import carmensystems.studio.cuibuffer as cuib
import carmensystems.studio.cpmbuffer as cpmb

from report_sources.include import standardreport

class Report(p.Report):

    def create(self):
        win_buf = cuib.CuiBuffer(cuib.CuiWhichArea, cuib.WindowScope)
        cbag = r.buffer2context(win_buf).bag()
        warning = cbag.studio_sno.marked_trips_warning_txt()
        self.add(standardreport.standard_header(self, "Trip Report", warning))
        self.add(standardreport.standard_footer(self))
        m_buf = cpmb.CpmBuffer(win_buf, 'studio_sno.%trip_is_marked%')

        cbag = r.buffer2context(m_buf).bag()

        for trip_bag in cbag.iterators.trip_set():
            self.add("TRIP START: %s" % trip_bag.trip.start_hb())
```

One warning-variable per level Needed in Rave

A selection-variable that gives a **correct set** of ledobs with **right level**.

SlideContext9.py

Best practice – how to implement II

Idea:

- put all the complexity in one place
- use the same pattern in all reports.

Solution:

Used by OTS etc.

`carmstd.bag_handler`

- always used to create context-bags
- one class for each context-bag used in CARMUSR-python-code.
- a class instance provides the attributes:

`bag` : a Rave-bag or `None`

`warning` : Always a message when `bag` is `None`.
Sometimes a message when `bag` is a Rave-bag.

`warning_eng` : The same message, but always in English.

Take a look

Best practice – how to implement III

The trip report again:

```
import carmensystems.publisher.api as p
import carmensystems.rave.api as r

from carmstd import bag_handler
from report_sources.include import standardreport

class Report(p.Report):

    def create(self):

        tbh = bag_handler.MarkedTripsMain()
        self.add(standardreport.standard_header(self, "Trip Report", tbh.warning))
        self.add(standardreport.standard_footer(self))
        if not tbh.bag:
            return

        for trip_bag in tbh.bag.iterators.trip_set():
            self.add("TRIP START: %s" % trip_bag.trip.start_hb())
```

top bag handler

The **same pattern** can be used in **all reports**.

SlideContext9b.py

Exercise

Now it is time for exercise 6

Content of this section

- Rule failures in the Rave API
- Recap of constraints
- Constraints in the Rave API

Introduction

`Bag` method:

```
rulefailures(where=None, sort_by=None)
```

- Requirement:

  Only one ledob in the active bag

- Returns an iterator – normally used in for loops

  In each loop a tuple with two objects is returned:

  - a `Bag` with one ledob

    level from the failed rule

  - a `RuleFail` object

    large number of attributes:
    `rule`, `limitvalue`, `actualvalue`, `failtext`...

Old obsolete possibility:

the function `rulefailure` as argument to `foreach`.

Example – Check legality report for current trip

Rule	Duty	Leg	Actual	Limit	Diff
(EXP) Min connection time	3	2	0:40	1:00	0:20
(EXP) Min connection time	2	1	0:40	1:00	0:20
(EXP) No rest periods at trip homebase	3	-	-	-	-
(EXP) No rest periods at trip homebase	2	-	-	-	-
(JAR) Max duty time when more than 3 early starts/late ends between weekly rests	4	-	13:30	10:00	3:30
(JAR) Max duty time when more than 3 early starts/late ends between weekly rests	3	-	11:15	10:00	1:15
(JAR) Max duty time when more than 3 early starts/late ends between weekly rests	2	-	10:10	10:00	0:10

Example – code

Rule	Duty	Leg	Actual	Limit	Diff
(EXP) Min connection time	3	2	0:40	1:00	0:20
(EXP) Min connection time	2	1	0:40	1:00	0:20
(EXP) No rest periods at trip homebase	3	-	-	-	-
(EXP) No rest periods at trip homebase	2	-	-	-	-
(JAR) Max duty time when more than 3 early starts/late ends between weekly rests	4	-	13:30	10:00	3:30
(JAR) Max duty time when more than 3 early starts/late ends between weekly rests	3	-	11:15	10:00	1:15
(JAR) Max duty time when more than 3 early starts/late ends between weekly rests	2	-	10:10	10:00	0:10

```
class Report(p.Report):
    def create(self):
        tbh = bag_handler.CurrentChain()
        - the normal stuff -
        self.set(border=p.border_all(1))
        self.add(p.Row("Rule", "Duty", "Leg", "Actual", "Limit", "Diff",
                       font=p.font(weight=p.BOLD)))
        for chain_bag in tbh.bag.chain_set():
            for f_bag, fail in chain_bag.rulefailures():
                leg_num_text = duty_num_text = "-"
                try:
                    duty_num_text = f_bag.duty.duty_num_in_trip()
                    leg_num_text = f_bag.leg.leg_num_in_duty()
                except r.UsageError:
                    pass
                self.add(p.Row(fail.rule.remark(),
                               duty_num_text,
                               leg_num_text,
                               "-" if fail.actualvalue is None else fail.actualvalue,
                               "-" if fail.limitvalue is None else fail.limitvalue,
                               "-" if fail.overshoot is None else fail.overshoot))
```

Exception if incorrect level

Do not apply iterators. The bag always contains all legs in the **chain**.

SlideRaveRC1.py

Recap

Some of the syntax:

```
constraint [(on|off)] constr_name =
  (foreach ;)*
  [valid ;]
  constr_condition ;
  [nonadditive(formalargs) = expr ;]
  [cost [(expr)] = expr ;]
  [failtext ;]
  [remark "text" ;]
end
```

Example of vertical constraint:

```
constraint max_inexperienced =
    valid leg.%is_flight_duty%;
    count(equal_legs)
    where(crew.%is_inexperienced%)
    <= 1;
end
```

transform -> Vertical constraint

Example of global constraint:

Formats the loop variable

```
constraint bc_glc_max_daily =
    foreach row in set(1, %rows_in_bc_table%) alias (%bc_daily_output_one%(row));
    foreach day in set(%bc_lp_period_start_day%, %bc_lp_period_end_day%, 24:00)
    alias (%bc_output_day_name%(day))
    where (%bc_max_daily_prod_active%(row, day));
    sum(sp_crrs, %bc_daily_prod_constr%(row, day)) <= %bc_max_daily_prod%(row);
    cost = %bc_excess_cost%(row);
    remark "Max daily base prod";
end
```

context -> Global constraint

Rave API

For vertical constraints:

`constraint_evals(where, sort_by)`

Method of `bag` (only one ledob in the bag)

For global constraints:

`global_constraint_evals(where, sort_by)`

Global function in the API

They are both very similar to `rulefailures()`.

Returns (`Bag, ConstraintsEval`) when used in a `for` loop

Note:
The **`Bag`** returned by **`global_constraint_evals`** is empty.

APC feedback about constraints

A new resource in v18:

`apc.config.FailtextLegacyMode`

True (default in v18)

1. `failtext` (if failtext defined)
2. `remark(loopvariables) | type | Capacity | Value`

Use False!

False (default in V19):

`remark(loopvariables) | Type | Capacity | Value | Overshoot | Cost`

Constraint	Type	Capacity	Value	Overshoot	Cost
GLC1 - max long (>= 4 days) trips from base(LGW)	<=	10	0	0	0
GLC1 - max long (>= 4 days) trips from base(MAN)	<=	5	1	0	0

Get the same information in Studio:

Use **Special> Reports> Global Constraints**
Special> Reports> Violated Vertical Constraints

Take a quick look at the source code

Rave API – constraints

Limitations

- Not supported by Matador/APC.
- The ”bag” to consider for the calculation is defined to a context/transform in the Rave definition, not in the API.

Exercise

Now it is time for exercise 7

Transforms & Performance

Transforms

- Recap
- Example

Performance

What to avoid?

Introduction / Recap

- Defined by the application
- Documentation, ***Help> Keywords etc.***
- Often used:
  - `ac_rotations_ref`
  - `equal_legs`
- Mainly used in (compiled) Rave code

- There is a `Bag` method for each transform

  returns a new `Bag` (supports only `Const` Rave definitions)

- Restriction

  There must be just ONE object in the bag when you use a transform.

  For the transforms above it must be a ONE-leg-ledob.

Example – all crew deadheading on a leg

```
class Report(p.Report):
    def create(self):
        tbh = bag_handler.CurrentLeg()
        ... Normal stuff ...
        for leg_bag in tbh.bag.atom_set():
            self.add(p.Text("Crew deadheading on flight %s" % leg_bag.flight_number(),
                            font=p.font(weight=p.BOLD)))
            tot = 0

            for dh_bag in leg_bag.equal_legs().atom_set(where="deadhead"):
                if dh_bag.crr_crew_id():
                    self.add("Crew %s" % dh_bag.crr_crew_id())
                    tot += 1
                else:
                    ns = sum([dh_bag.assigned_crew_position(ix)
                              for ix in xrange(1, 13)])
                    if ns:
                        self.add("Unassigned trip %s for %d crew" %
                                 (dh_bag.crr_name(), ns))
                        tot += ns

            self.add("Total: %d" % tot)
```

Crew deadheading on flight 2908
Unassigned trip 3020 for 1 crew
Crew 100161
Total: 2

Performance

If **performance** is an **issue**:

- do calculations in compiled Rave code.
  - avoid Rave expressions in the API.
  - do not calculate in Python

- `bag` or `eval` does not matter

Look at the report:
`PerformaceTestRaveAPI_expr.py`

Exercise

Now it is time for exercise 8

Last slide

Courses:

- PRT2
- Python in Studio

Q / A

Evaluation

Welcome back to Jeppesen Crew Academy!

