Course manual

Rave from Python & PRT I

Version 22

JEPPESEN
A BOEING COMPANY

Göteborg November 2015

Table of contents

General

Python Report Toolkit, **PRT**, is a toolkit for report generation. The reports are defined in Python code, using objects defined in the PRT API. The API provides classes and functions for layout of e.g. complex tables and charts using multiple fonts, styles, alignments, graphics, etc. The currently supported output formats are HTML, PDF and plain text.

PRT can get data from any source, but you often use **Rave's Python API** to get data.

In this course you learn about the Python APIs for PRT and Rave.

Rave requires a main application and Report generation is normally started from an application, but can be used stand alone.

Both PRT and Rave’s Python APC are supported by a number of Jeppesen applications. In this course we use **Studio** as main application. Studio provides its own web server and an embedded browser for display of HTML reports with support for interactivity.

Overheads

Exercises

Introduction

Log on

Step 1 Double click on the **Academy Desktop** icon 

.

Step 2 Double click on the Exceed onDemand passive icon 
.

Step 3 Double click on the **Launcher** start icon  for the PRT1 course.

If certificate warnings pop up – just press **Yes.**

Step 4 Log in as: **student## / password**

Step 5 Press the **Studio**,  , button in the Launcher.

The Special menu

The system you are using for the exercises has a **Special** menu in the top menu bar. Here you find commands that can be useful when you develop PRT reports. Examples:

- **Scripts> Python Code Manager…**
  Give you access to all Python files in CARMUSR, CARMSYS and NiceToHaveIQ.
  You can **reload** modules. That is needed if you have changed a PRT definition file.
- **Logs etc.>Tail of Studio Log**
  Gives access to Studio's log file. Error messages from Python, e.g. detailed information about raised exceptions, are written there.

- **Search> in Python Files…**
  Free text search in all Python files in CARMSYS and CARMUSR.

NiceToHaveIQ

The Special menu is defined in a "package" called `NiceToHaveIQ`.
The package is probably already available at your site. If not, you can download it from the Piazza:
https://jira.jeppesensystems.com/confluence/display/CDP/Nice+to+Have+IQ+package

In the package you also find useful Python code. You are supposed to use code from the package in some exercises. The path to it is:
`/carm/academy/NiceToHaveIQ`.

Exercises using solutions

In some cases you are supposed to continue with the solution from one exercise in another. Therefore keep your solutions. If you not have done the previous exercise you can get the solution from your teacher.

Many exercises

There is probably not enough time for you to do all exercises during the course. Pick those that look most valuable to you. Some of the exercises have been marked *(extra)* or *(maybe extra)*. They are often a bit more demanding than the others.

Use Developer Workspace

If you want, you can use DWS when you work with the exercises. You find a step-to-step guide for it in the end of the course material.

Get started

Purpose

Become familiar with the tools you will use in the course.

Start to use the documentation and create your first PRT reports.

Exercise 1

Exercise 1.1 Find information in the documentation

Step 1 Read about the `Report` class in the *API documentation*. What is the name of the method to use if you want to define the report to be in landscape format?

Step 2 `$CARMSYS/bin/publisher`.
What does the flag `-f` mean?

> **Hint:** Execute the command with the flag `-h`.

Step 3 There is an example report in CARMSYS called `helloworld.py`.
View it by using the menu entry
Special> Scripts> Python Code Manager...

Exercise 1.2 Create your first PRT report

Step 1 Create a report that is possible to generate by the command:
Planning Tools> Generate Report….
The report should display the string:
`My first PRT report`.
Verify that it works. Try both PDF and HTML.

> **Hint:** Information about Python exceptions is found in Studio's log file.

Exercise 1.3 Setup a good development environment

Step 1 When modifying a report, it is a bit inconvenient to always:

- change the report definition
- save the file
- reload from the **Python Code Manager**
- generate from the menu entry.

Can this be improved?

One way is of course to avoid using Studio. However, that is only possible as long as you don't need internal Studio functions to collect data for the report.

A better idea is to reload the module and generate the report when you execute the Python file as a script. (You run it as a script by pressing the **Run** button in the **Python Code Manager**.)

There is a function in the NiceToHaveIQ package you can use.
It is named `reload_and_display_report` and is found in the file `lib/python/nth/studio/report_generation.py`.
The module has already been copied to `lib/python` in your CARMUSR.
Use the function in the testing section of your report definition file.

Then make a minor change to the text in the report definition and verify that the change is considered when you click the **Run** button.

> **Hint:** Add code for testing in all reports you create during the course.

Step 2 Developer Workspace *(maybe extra)*

Use **DWS** instead of **Python Code Manager**. You find instructions for getting started with DWS in the end of this course material.

If you use the **Run in Studio** command in DWS you are asked to save the file. With that we have really got what we aimed for. Save, reload and generate in one single command!

Exercise 1.4 Define more simple reports *(maybe extra)*

Step 1 Create a report that is possible to generate from the object popup menu in the **Duties** window. The report should display the string:
My second PRT report.
Verify that it works.

> **Hint:** To be able to test the report you have to load a sub-plan and delete at least one trip to a duty chain.
>
> **Hint:** You must create a new module package.

Step 2 Change the report to also show the name of the module defining it, on a new row.

> **Hint:** You can make one more call to `self.add`.
>
> **Hint:** You can use the module attribute `__name__`.

Verify that it works.

Exercise 1.5 Generate without Studio *(extra)*

Start an `xterm` from the **Special** menu in Studio and generate your report from the previous exercise using the `publisher` command.
Test TXT, HTML and PDF as output format.

> **Hint:** You can use the command `evince` to display PDF files and the command `firefox` to display HTML files.

Basic objects and properties

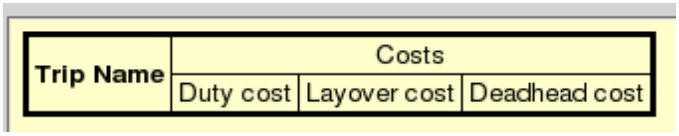
Purpose

Here we become familiar with basic objects and their properties in PRT.

Exercise 2

Exercise 2.1 Create the title area for a trip cost report

Use basic objects and properties in PRT to create this report.



Hint: The colour is LightYellow from the Studio palette. `lib/python/nth/studio/studiopalette.py` in the `NiceToHaveIQ` package has already been copied to the course CARMUSR.

Exercise 2.2 A simple table

Create a report which looks like the following table. The airport names should be left-justified and the number of departures right-justified.

Airport	Departures
LHR	102
GOT	8
AMS	34
ARN	23

Hint: You can use the file `CoursePrt1ExObj2.py` as a start.

Exercise 2.3 An old friend

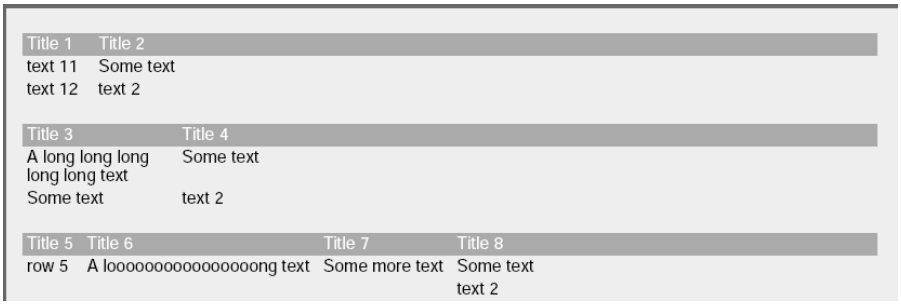
This exercise has been present in the report courses since the very beginning. Create the following report:

Make a solution where you do not define the texts more than once.

Note You can't use an object more than once in a report.

Exercise 2.4 Table layout *(maybe extra)*

This is a real case from an implementation project. The client wanted the layout below for a number of small tables in a report.



Title 1	Title 2
text 11	Some text
text 12	text 2

Title 3	Title 4
A long long long long long text	Some text
Some text	text 2

Title 5	Title 6	Title 7	Title 8
row 5	A looooooooooooooong text	Some more text	Some text text 2

As you can see the columns should be left-justified and the table header should span the entire page. The developer who worked with the report needed help to solve the problem. What he had done is available in the file `CoursePrt1Obj4.py`. Can you modify the code to get the layout requested by the client?

Hint: The trick is to use many objects with a minimum cell width of 0 pts. (Maybe there will be support for "springs" in a future version of PRT).

Hint: Do not use HTML. There are bugs.

Hint: You should not change the tables. (They are of course supposed to be replaced by real tables later on.)

Exercise 2.5 Board of checkers

The exercise is mainly interesting if you have PDL reports that will be rewritten in PRT.

Generate the report **Example.pdl** (format PDL – not PDF).

Create a report that looks the same by using PRT.

Note The purpose of this exercise is partly to learn how to transfer images from PDL reports to PRT reports. If you are not interested in that part copy the pictures from the solution CARMUSR (ask your teacher how to do it).

Hint: To convert bitmaps in PDL reports to normal X11-bitmap files you can use the script `pdl2bm.csh` (found in `NiceToHaveIQ/bin`). To convert bitmaps to jpeg you can e.g. use **xv** or **gimp**.

Pages and cross references

Purpose

To get some experience in how PRT handles pages, cross-references, bookmarks headers and footers.

Study

Read about the classes `Crossref`, `Header` and `Footer`, the `Column` methods `page`, `add_header`, `add_footer` and the `Box` property `bookmark` in the API documentation.

Look also in the section *Development> Rave Publisher PRT Reference> Tutorial> Examples> Cross- references* in the *System Help*.

Exercise 3

Exercise 3.1 A test report

In `$CARMSYS/lib/python/carmensystems/publisher/examples` you find the report `test_pagebreaks.py`.

Step 1 Enable it

Make it possible to generate this report from the menu entry **Planning Tools/Generate Report…**.

You should not copy the file and you should not create any soft link.

Step 2 Study the implementation

Make sure that you understand it.

Step 3 Generate for PDF and HTML

- Do you discover any differences?
- What happens if you click a cross-reference link?

Exercise 3.2 Common header and footer

A client normally wants all reports to get the same look. The same header and footer definitions are therefore often used for all reports. The course CARMUSR contains the module `report_sources.include.standardreport` with header and footer definitions.

Step 1 Create a variant of the report you created in Exercise 2.2 which has the standard header and footer.

Hint: You can create a subclass of the original report class.

Step 2 Change the logotype in the header of the standard header to one from your own company. Verify that it works with the report you created in the previous exercise.

> **Hint:** You can use a **WEB browser** to find the logotype and then press the **key** `Print Scrn` to get a screen dump into the clipboard buffer and finally the program **Start>Programs> Accessories> Paint** to create a jpeg file from what you have in the clipboard buffer.
>
> **Hint:** In Windows the CARMUSR is found as `H:\prt1_system\user`.

Exercise 3.3 Bookmarks

Modify the report `SimpleCrewReport.py` you find in the object popup-menu in the main area of the roster window. Add a bookmark per crew.

> **Hint:** You have to load a roster plan. E.g.:
> `Jeppesen/2012Apr/FC_737_DATED_ROSTERING/MATADOR_1/best_solution`

The Rave API – Basics

Purpose

Here you get started with the Rave API.

Study

Find and look at the sections about the Python Rave API in the documentation:

1. *Help> API Documentation> Rave Python API*

   Note that you find information about the bag interface in the first page.

2.  *> Development> Developer Guide> Modelling with Python> Rave module*

Exercise 4

Exercise 4.1 A simple evaluation

Study the code and generate the PRT report
`report_menu/CoursePrt1ExRave1.py`

Exercise 4.2 List all rules

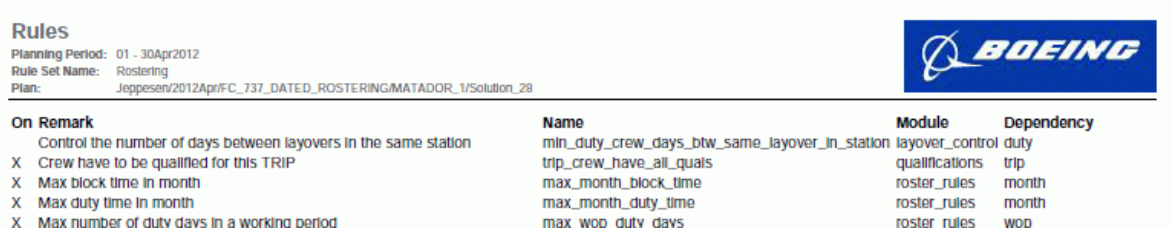
Make a report that lists all rules in the loaded rule set.

Name, **module**, **remark** and **dependency** for each rule should be presented together with information about if the rule is switched **on**.

Requirements:rule

- The report should be in landscape format.
- Standard header and footer should be used.
- There should be a header text in each column on every page.
- Output should be OK for any of the three available rule sets.
- Truncate the remark if needed.
- Sort by the remark.
- It should work also if there are rules in _topmodule (the rule set BuildRotations).

You can concentrate on PDF as output format.
This is how the top of the report can look:



Rules

Planning Period: 01 - 30Apr2012
Rule Set Name: Rostering
Plan: Jeppesen/2012Apr/FC_737_DATED_ROSTERING/MATADOR_1/Solution_28

On	Remark	Name	Module	Dependency
	Control the number of days between layovers in the same station	min_duty_crew_days_btw_same_layover_in_station	layover_control	duty
X	Crew have to be qualified for this TRIP	trip_crew_have_all_quals	qualifications	trip
X	Max block time in month	max_month_block_time	roster_rules	month
X	Max duty time in month	max_month_duty_time	roster_rules	month
X	Max number of duty days in a working period	max_wop_duty_days	roster_rules	wop

Hint: Use the `rules` iterator provided by `carmensystems.rave.api`.

Exercise 4.3 List enum parameters *(maybe extra)*

Using `enum` for Rave parameters improves usability (a pick list is then provided in the Rule parameter form). Create a report that makes it easy to find out how much this possibility is used. It should list all `enum` parameters in the loaded rule set.

The report should look something like this:

Parameters of enum type

CARMUSR: /users/stefan/work/Prt1TeacherSystem/user
Rule set: Pairing
Date: 28Nov2012
Total number: 5
List of parameters:
crg_basic.enum_constant
apc_pac.suggest_aircraft_turns_airport_selection
apc_pac.column_generation_glc_misestimate_handling
apc_pac.column_generation_block_statistics_use
crew_pos_ots.crew_category_p

Rave API – consider data

Purpose

Here we start to use data from the loaded plan in our reports.

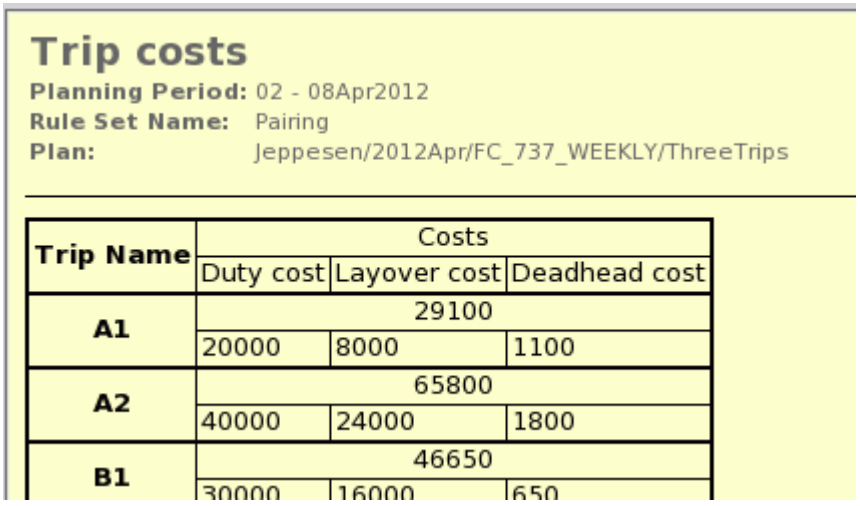
Exercise 5

Exercise 5.1 Trip cost report

Add data to the report you created in Exercise 2.1.

The report should be available in the **Planning Tools** menu and should consider all trips in the plan. Make sure you get adequate page breaks and a column header on each page (if PDF is used).

Add the standard header and standard footer to the report.



Trip costs

Planning Period: 02 - 08Apr2012
Rule Set Name: Pairing
Plan: Jeppesen/2012Apr/FC_737_WEEKLY/ThreeTrips

Trip Name	Costs		
	Duty cost	Layover cost	Deadhead cost
A1	29100		
	20000	8000	1100
A2	65800		
	40000	24000	1800
B1	46650		
	30000	16000	650

Hint: The command **Set Names** (in Trip Object pop-up menu) sets numbers to the selected trips.
You find Rave definitions for the costs in the module `trip_cost`.

Exercise 5.2 Layover information

Create a simple report that becomes available in the planning tools menu and considers all (not assigned) trips in the sub-plan. It should show all the layover stations for the trips and the number of layovers at each station. Sort the list by the number of layovers. The station with most layovers should come first.

Verify the result on any pairing plan. E.g.:
`Jeppesen/2012Apr/FC_737_DATED/APC_1/best_solution`

Hint: The iterator `iterators.layover_set` can be used. The variable `iterators.num_duties_in_bag` can also be useful.

Hint: You can skip all page handling.

Station	Layovers
GLA	75
JER	74
NAP	73
EDI	72
BLQ	62
PEQ	17

Exercise 5.3 A set

Layout and sort order should be as in the previous exercise, but instead of all airports having at least one layover, the airports in the parameterized set `crg_basic.prt_course_layover_stations` should be considered.

Note Also stations in the set without layovers should be presented in the report.

Hint: Create a dictionary with the data for each layover station.

Note It is a common requirement that you want to show a figure also when there is no data.

Exercise 5.4 Accumulated block time *(maybe extra)*

For each trip in the sub-plan, display the trip number and for each leg in the trip the accumulated block time (in the actual duty) and the arrival airport.

Example:

1005	LGW	MAN	LGW	GLA	
		1:00	2:05	3:30	
	GLA	LGW	VNO	LGW	
		1:30	4:20	7:25	
1009	LGW	TLS	LGW	BCN	LGW
		1:45	3:30	5:35	7:50
1003	MAN	LGW	ZRH	LGW	NCL
		1:10	2:50	4:30	5:45
	NCL	LGW	EDI	LGW	TLS
		1:25	2:50	4:15	6:00
	TLS	LGW	BLQ		
		1:50	4:00		
	BLQ	LGW	JER	LGW	JER
		2:20	3:20	4:15	5:15
	JER	LGW	VRN	LGW	MAN
		1:00	3:15	5:20	6:20
1011	LGW	VCE	LGW	MRS	
		2:10	4:25	6:20	
	MRS	LGW	ABZ		

Hint: You can use "`arrival - departure`" to get the block time for a leg.

More about contexts

Purpose

The purpose is to become familiar with the bag handler.

Study

Look at the code of the bag handler.

Exercise 6

Exercise 6.1 Use the bag handler

Modify the report you created in 5.1 to use the bag handler. Then generate the report without a loaded sub plan.

Exercise 6.2 Consider selected objects

Modify the report you made in 6.1 to consider all fully selected trips in the window. The report should be possible to generate from the object menu in the trip window. If there are partly selected trips in the window an adequate warning should be written in the report header.

Note Make an implementation where the two reports share as much code as possible.

Exercise 6.3 Number of legs in the “current” trip

Create a report for the roster window object pop-up. It should just show the number of legs in the “current” trip.

If you generate the report with the testing code there is maybe no current object in Studio. Make sure you display an adequate warning in this case.

Hint: Use the `bag_handler.CurrentTrip`.

To make it more interesting you are not allowed to use any definition from your Rave code.

Hint: Use the built-in iterator `atom_set`.

Exercise 6.4 List all keywords *(extra)*

You are supposed to test a new version of the product. One of the tasks is to verify that all keywords are identical for all kinds of legs.

Create a simple report that prints name, value, type and dependency of all keywords.

- Skip the **value** of keywords that are supposed to give different values.
- It should be possible to generate the report for a **single leg** in the trip window and in the roster window.

Note Some keywords require arguments. Provide adequate arguments when needed.

When the report works fine in v22, use it in v21 too and compare the results.

Hint: To start v21 you can change the carmsys link.

Hint: Produce for HTML. Copy the text to a text file.
Use e.g. `tkdiff` to compare the results.

Are all keywords identical?

Are there any new or removed keywords?

Rule failures and constraints

Purpose

To get some experience in accessing rule failures and constraints in the Rave API.

Study

- Look in the API documentation and see what is written about:
- the bag iterators:
  - `rulefailures`
  - `constraint_evals`
- The function `global_constraint_evals`
- `RuleFail` objects
- `ConstraintEval` objects

Exercise 7

Exercise 7.1 Number of rule failures per rule

Modify the report you created in 0.

The previous version of the report shows, in the first column, if a rule is on or off. Change that column to show the number of rule violations (if the rule is on) in the plan.

Hint: You can use the bag handler object `PlanCrewChains`.

Exercise 7.2 Replace PDL with PRT

Modify the menu entry **CheckLegality** in the left part of the Roster window to display a PRT report.

Hint: Modify and create a subclass of the existing `crew_window_object/CheckLegality.py`.

Hint: The menu entry is defined in `menu_scripts/Cas/StandardExtensions`.
The used Python function, `show_report_popup`, can be used for PRT reports too.
You must restart Studio if you change menu definitions.

Exercise 7.3 Constraints

Load a Pairing plan.

Take a look at the implementation of the report **Global Constraints** in the **Special> Reports** menu. Test it.

Run APC and look at the feedback about global constraints.
Note the setting of the CRS resource `apc.config.FailtextLegacyMode`.

Transforms & Performance

Purpose

To get some experience in:

- using transforms in the Rave API
- performance aspects.

Study

See what is written about transforms in the *Python Rave API* documentation.

Exercise 8

Exercise 8.1 Performance

Load a large (rostering) plan and generate the report
`report_menu/PerformanceTestRaveAPI_expr.py`

Study the report and make sure you understand what performance you get for different ways of using the API.

Exercise 8.2 List all crew working on a flight leg

Create a simple report that can be used in the object popup menu in the Roster and the Trip window. It should display the name, function and crew ID of all crew working on the leg. Something like:

Exercise 8.3 Consider best practice

Modify the report you made in 8.2. Implement the following step by step:

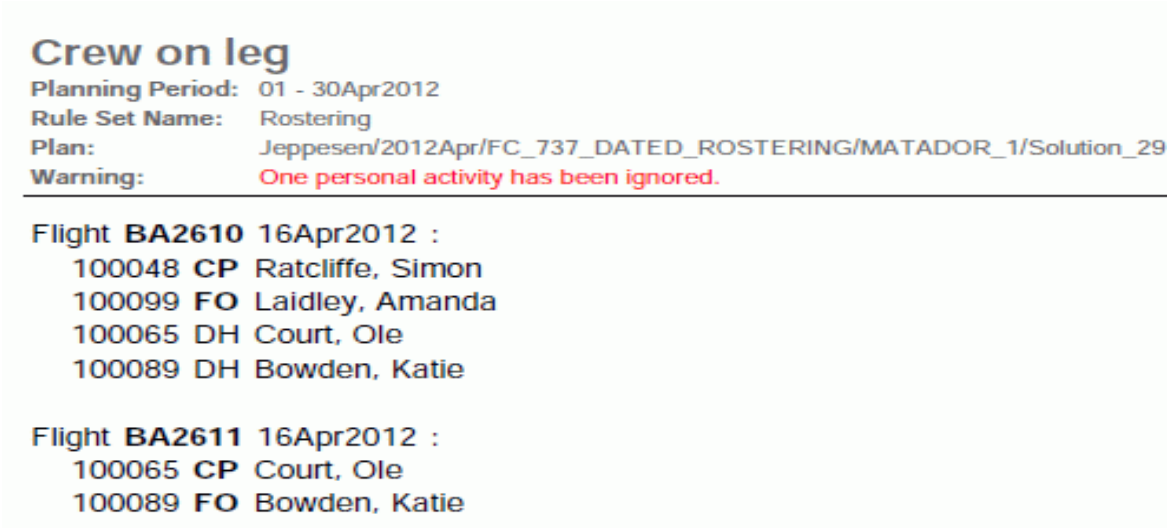
- add header and footer
- consider all selected legs in the window
  - do not duplicate information if a leg is selected twice.

Hint: `%same_leg_id%` and `same_leg_iterator` in `crg_basic` may be useful.

- ground duties should work fine
- give a warning in the header for selected personal activities
- include information about crew deadheading on the leg
- deterministic sort order
  - for the legs
    - flights before ground duties
    - flights and ground duties in alphabetic order
  - for crew:
    - active before deadhead
    - positions in alphabetic order
    - crew id
- Only id and position with bold text.

It is still enough to consider legs in rosters.

The report should look something like this:



Crew on leg

Planning Period: 01 - 30Apr2012
Rule Set Name: Rostering
Plan: Jeppesen/2012Apr/FC_737_DATED_ROSTERING/MATADOR_1/Solution_29
Warning: One personal activity has been ignored.

Flight **BA2610** 16Apr2012 :
100048 **CP** Ratcliffe, Simon
100099 **FO** Laidley, Amanda
100065 DH Court, Ole
100089 DH Bowden, Katie

Flight **BA2611** 16Apr2012 :
100065 **CP** Court, Ole
100089 **FO** Bowden, Katie

Solutions

Exercise 1 Get started

Exercise 1.1 Use the documentation

Step 1 `self.setpaper(orientation=p.LANDSCAPE)`

How:

Help> API Documentation
Click on **Python Report Toolkit**
Click on the class **carmensystems.publisher.api.Report**
Click on and read about **setpaper**

Step 2 You select output format.
"`-f html`", "`-f pdf`" and "`-f txt`" are supported

How:

Execute the Linux command:
`publisher -h`.

Alternative:

Take a look in the manual.

In **System Help,** select:

Development>
Rave Publisher PRT Reference>
Introduction to Python Report Toolkit>
Using a command line script

> **Hint:** The easiest way to find the information you need in the manuals is often to use the search command. In this case it does not work very well. The string **publisher** is very common. **Publisher script** works better, but is not obvious to find out.

Step 3 View example report.

Special> Scripts> Python Code Manager…

Select the directory:

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

(presented as `<SYS>/carmensystems/publisher/examples`).
Select the file:
`helloworld.py`.

Click **View**

Exercise 1.2 Create your first PRT report

Step 1 In CARMUSR`/lib/python/report_sources/report_menu` add the file `ExIntro2_1.py`:

```
import carmensystems.publisher.api as p

class Report(p.Report):
    def create(self):
        self.add(p.Text('My first PRT report'))
```

Exercise 1.3 Setup a good development environment

Step 1 To reload and generate by the **Run** button add the following to the end of the report definition file:

```
if __name__ == "__main__":
      import report_generation as rg
      rg.reload_and_display_report()
```

Step 2 `DWS`

-

Exercise 1.4 Define more simple reports

Step 1 Create a new module package:

```
cd $CARMUSR/lib/python/report_sources
mkdir rtd_window_object
touch rtd_window_object/__init__.py
```

In the new module package add the file `ExIntro4_1.py`:

```
import carmensystems.publisher.api as p
class Report(p.Report):
    def create(self):
        self.add(p.Text('My second PRT report'))
```

Step 2 Add one row in the end of the report definition file:

```
self.add(p.Text(__name__))
```

or

```
        self.add(__name__)    # A Text object is created automatically
```

Exercise 1.5 Generate the report from xterm

```
> cd
> publisher report_sources.rtd_window_object.ExIntro4_2
> evince report_sources.rtd_window_object.ExIntro4_2.pdf
> publisher -f html report_sources.rtd_window_object.ExIntro4_2
> firefox report_sources.rtd_window_object.ExIntro4_2.html
> publisher -f txt report_sources.rtd_window_object.ExIntro4_2
> more report_sources.rtd_window_object.ExIntro4_2.txt
```

Exercise 2 Basic objects and properties

Exercise 2.1 Create the title area for a cost report

Create the report file:

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

class Report(p.Report):
    def create(self):
        self.set(background=sp.LightYellow)
        self.add(p.Row(p.Text("Trip Name",
                              valign=p.CENTER,
                              font=p.font(weight=p.BOLD)),
                       p.Column(p.Text("Costs",
                                       align=p.CENTER),
                                p.Row(p.Text("Duty cost"),
                                      p.Text("Layover cost"),
                                      p.Text("Deadhead cost"))),
                       border=p.border(top=2, bottom=2, left=2, right=2,
                                       inner_wall=1, inner_floor=1)))
```

Exercise 2.2 A simple table

Report definition file:

```
import carmensystems.publisher.api as p

data = [("LHR", 102), ("GOT", 8), ("AMS", 34), ("ARN", 23)]

class Report(p.Report):
    def create(self):
        self.add(p.Row(p.Text("Airport",
                              border=p.border(right=1)),
                       p.Text("Departures"),
                       font=p.font(weight=p.BOLD),
                       border=p.border(bottom=2)))
        for a, n in data:
            self.add(p.Row(p.Text(a, border=p.border(right=1)),
                           p.Text(n, align=p.RIGHT),
                           border=p.border(bottom=1)))
```

Exercise 2.3 An old friend

One possible solution:

```
import carmensystems.publisher.api as p
class Report(p.Report):
   def create(self):
       for i in xrange(9):
           if not i % 3:
               r = self.add(p.Row())
           if i % 2:
               r.add("")
           else:
               r.add(p.Column("The", "Report", "Generator",
                              border=p.border_frame(1)))
```

Note `Text` objects are created automatically for all ordered arguments to `Row` and `Column`.

Note The add method returns the added object.

Exercise 2.4 Table layout

One possible solution:

```
<             self.add(p.Isolate(table, width=w))
---
>             for _ in xrange(50):
>                 table.add(TT("", rowspan=1,
>                              padding=p.padding(0, 0, 0, 0)))
>             self.add(p.Isolate(p.Column(table, width=w)))
```

Exercise 2.5 Board of checkers

Create the jpeg files.

1. Copy the bitmaps from the PDL file into temporary files, for example called `black.pdl` and `white.pdl`. Example `white.pdl`:

```
bitmap
25x13
    <
    00 1c 00 00 00 63 00 00 80 80 00 00
    40 1c 01 00 40 22 01 00 20 49 02 00
    20 5d 02 00 20 49 02 00 40 22 01 00
    40 1c 01 00 80 80 00 00 00 63 00 00
    00 1c 00 00
    >
end
```

2. Execute the commands:
   ```
   /carm/academy/NiceToHaveIQ/bin/pdl2bm.csh black.pdl > black.bm
   /carm/academy/NiceToHaveIQ/bin/pdl2bm.csh white.pdl > white.bm
   ```

3. To convert `black.bm` and `white.bm` to jpeg files with **xv**:

   Command:
   ```
   xv white.bm
   ```

   In xv:

   Right "**Save**".
   Change the suffix to "jpg".
   Change format to "JPEG".

   Press **OK**

   Click **Quit**.

4. Put the jpg files in `$CARMUSR/images`.

5. Create the report. A possible report definition:

```
import carmensystems.publisher.api as p
CELLSIZE = 19
class Report(p.Report):
    def create(self):
        r = self.add(p.Row())
        for s in " ABCDEFGH":
            r.add(p.Text(s, align=p.CENTER))
        for row in xrange(8):
            r = self.add(p.Row())
            for col in xrange(9):
                if col == 0:
                    r.add(p.Text(row + 1,
                                 valign=p.CENTER))
                elif row < 3 and (row + col) % 2:
                    r.add(p.Image("white.jpg",
                                  width=CELLSIZE,
                                  border=p.border_frame(1),
                                  valign=p.CENTER))
                elif row > 4 and (row + col) % 2:
                    r.add(p.Image("black.jpg",
                                  width=CELLSIZE,
                                  border=p.border_frame(1),
                                  valign=p.CENTER))
                else:
                    r.add(p.Text("",
                                 padding=p.padding(0, 0, 0, 0),
                                 width=CELLSIZE,
```

```
                                height=CELLSIZE,
                                border=p.border_frame(1)))
```

Exercise 3 Pages and cross references

Exercise 3.1 A test report

Step 1 Enable

Create a one line report definition file:

```
from carmensystems.publisher.examples.test_pagebreaks import Report
```

Step 2 -

Step 3 Differences HTML – PDF

- Calls to `page` and `page0` are ignored in HTML.
- Pages breaks are not visible in the HTML browser.
- The cross-reference links work fine in both HTML and PDF, but look differently.

Exercise 3.2 Common header and footer

Step 1 Add header and footer

One possible solution:

```
import carmensystems.publisher.api as p
import report_sources.report_menu.ExObj2 as e2
import report_sources.include.standardreport as std

class Report(e2.Report):
    def create(self):
        self.add(std.standard_header(self, "Exercise Page 2"))
        self.add(std.standard_footer(self))
        super(self.__class__, self).create()

if __name__ == "__main__":
    reload(e2)
    reload(std)
    import report_generation as rg
    rg.reload_and_display_report("PDF")
```

Note It is convenient to reload all modules (you still are modifying) the report is depending on in the testing section.

Step 2 New logotype

Put the new JPEG file in `$CARMUSR/images` and change the file name on the row

```
p.Image('fake_logo.jpg', align=p.RIGHT)
```

in

`CARMUSR/lib/python/report_sources/include/standardreport.py`

Exercise 3.3 Bookmarks

```
self.add(p.Text(" ",
                font=p.font(size=2),
                padding=p.padding(0, 0, 0, 0)))

self.add(p.Row(p.Text("%s:" % roster_info[0],
                      bookmark=p.bookmark(roster_info[0])),
               "%s %s " % roster_info[1:3]))

fs = "Signature: %s   Base: %s   Seniority: %s"
self.add(p.Row("", fs % roster_info[3:6]))
```

Exercise 4 Rave API – Basics

Exercise 4.1 A simple evaluation

-

Exercise 4.2 List all rules

One possible solution:

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

import report_sources.include.standardreport as std


class Report(p.Report):

    def create(self):
        self.setpaper(orientation=p.LANDSCAPE)

        self.add(std.standard_header(self, "Rules"))
        self.add(std.standard_footer(self))

        mc = self.add(p.Column())
        mc.add_header(p.Row("On", "Remark", "Name", "Module", "Dependency",
                      font=p.font(weight=p.BOLD)))

        for rule in sorted(r.rules(),
                           key=r.Rule.remark):

            module_name, rule_name = (rule.name().split(".")
                                      if "." in rule.name()
                                      else ("_topmodule", rule.name()))
            level = rule.level().name().split(".")[-1]
            remark = rule.remark()
            if len(remark) > 75:
                remark = remark[:71] + " ..."
            mc.add(p.Row("X" if rule.on() else "",
                         remark,
                         rule_name,
                         module_name,
                         level))
            mc.page()


if __name__ == "__main__":
    import report_generation as rg
    reload(std)
    rg.reload_and_display_report("PDF")
```

Note PRT lacks support for a more sophisticated solution than setting a hard coded limit for the length of the remark to not exceed the maximum page width.

Exercise 4.3 List enum parameters

One possible solution:

```
import os
import time

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

class Report(p.Report):

    def create(self):

        self.add(p.Text("Parameters of enum type",
                        font=p.font(weight=p.BOLD)))
        self.add(p.Row("CARMUSR:", os.getenv("CARMUSR")))
        self.add(p.Row("Rule set:", r.eval("rule_set_name")[0]))
        self.add(p.Row("Date:",
                       Dates.FDatInt2Date(Dates.FDatUnix2CarmTimeLT(time.time()))))

        enum_params = [param.name()
                       for param
                       in r.parameters()
                       if isinstance(param.value(), r.enumval)]

        self.add(p.Row("Total number:", len(enum_params)))
        if len(enum_params):
            self.add("List of parameters:")
            for enum_param in enum_params:
                self.add("  %s" % enum_param)
```

Exercise 5 Rave API – Consider data

Exercise 5.1 Trip cost report

One possible solution:

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

import report_sources.include.standardreport as std
from report_sources.include.studiopalette import studio_palette as sp

class Report(p.Report):

    def create(self):

        def create_one_section(na, c, c1, c2, c3):
            return p.Row(p.Text(na,
                                valign=p.CENTER, align=p.CENTER,
                                font=p.font(weight=p.BOLD)),
                         p.Column(p.Text(c, align=p.CENTER),
                                  p.Row(c1, c2, c3)),
                         border=p.border(top=2, bottom=2, left=2, right=2,
                                         inner_wall=1, inner_floor=1))

        self.set(background=sp.LightYellow)
        self.add(std.standard_header(self, "Trip costs"))
        self.add(std.standard_footer(self))
        mcol = self.add(p.Column())
        mcol.add_header(create_one_section("Trip Name",
                                           "Costs",
                                           "Duty cost",
                                           "Layover cost",
                                           "Deadhead cost"))
        cntx_bag = r.context("sp_crrs").bag()
        for trip_bag in cntx_bag.iterators.trip_set():
            da = trip_bag.trip_cost.trip_cost_days()
            al = trip_bag.trip_cost.trip_cost_allowances()
            ho = trip_bag.trip_cost.trip_cost_hotel()
            pa = trip_bag.trip_cost.trip_cost_passive()
            pd = trip_bag.trip_cost.trip_cost_per_diem()

            mcol.add(create_one_section(trip_bag.trip.name(),
                                        da + al + ho + pa + pd,
                                        da,
                                        al + ho + pd,
                                        pa))
            mcol.page()
```

Exercise 5.2 Layover information

Python module:

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

class Report(p.Report):

    def create(self):

        self.set(border=p.border_all())
        self.add(p.Row("Station", "Layovers",
                       font=p.font(weight=p.BOLD)))

        cntx_bag = r.context("sp_crrs").bag()
        for layover_bag in cntx_bag.iterators.layover_set(
                where="not duty.%is_last_in_trip%",
                sort_by="-iterators.%num_duties_in_bag%"):
            self.add(p.Row(layover_bag.duty.end_station(),
                           layover_bag.iterators.num_duties_in_bag()))
```

Exercise 5.3 A set

A possible solution:

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

class Report(p.Report):

    def create(self):
        cntx_bag = r.context("sp_crrs").bag()
        d = dict((layover_bag.duty.end_station(),
                  layover_bag.iterators.num_duties_in_bag())
                 for layover_bag
                 in cntx_bag.iterators.layover_set(
                     where="not duty.%is_last_in_trip%"))

        s = r.set("crg_basic.prt_course_layover_stations")
        l = [(d.get(a, 0), a) for a in s.members()]

        self.set(border=p.border_all())
        self.add(p.Row("Station", "Layovers",
                       font=p.font(weight=p.BOLD)))
        for num, stn in sorted(l, reverse=True):
            self.add(p.Row(stn, num))
```

Exercise 5.4 Accumulated block time

One possible solution:

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

class Report(p.Report):

    def create(self):

        cntx_bag = r.context("sp_crrs").bag()
        for trip_bag in cntx_bag.iterators.trip_set():

            dc = p.Column()
            tr = p.Row(trip_bag.trip.name(),
                       dc,
                       border=p.border_frame())
            self.add(tr)
            self.page()

            for duty_bag in trip_bag.iterators.duty_set():
                dr = p.Row(p.Column(duty_bag.duty.start_station()))
                dc.add(dr)

                sblock = RelTime(0)
                for leg_bag in duty_bag.iterators.leg_set():
                    block = leg_bag.arrival() - leg_bag.departure()
                    sblock += block
                    dr.add(p.Column(leg_bag.arrival_airport_name(),
                                    sblock))
```

Exercise 6 More about contexts

Exercise 6.1 Use bag handler

One possible solution:

```
import carmensystems.publisher.api as p
import report_sources.include.standardreport as std
from carmstd import bag_handler
from report_sources.include.studiopalette import studio_palette as sp

class Report(p.Report):

    def create(self):

        def create_one_section(na, c, c1, c2, c3):
            return p.Row(p.Text(na,
                                valign=p.CENTER,
                                align=p.CENTER,
                                font=p.font(weight=p.BOLD)),
                         p.Column(p.Text(c,
                                         align=p.CENTER),
                                  p.Row(c1, c2, c3)),
                         border=p.border(top=2, bottom=2,
                                         left=2, right=2,
                                         inner_wall=1, inner_floor=1))

        self.set(background=sp.LightYellow)

        tbh = bag_handler.PlanTrips()

        self.add(std.standard_header(self, "Trip costs - ExContext1", tbh.warning))
        self.add(std.standard_footer(self))

        mcol = self.add(p.Column())
        mcol.add_header(create_one_section("Trip Name",
                                           "Costs",
                                           "Duty cost",
                                           "Layover cost",
                                           "Deadhead cost"))

        if not tbh.bag:
            return

        for trip_bag in tbh.bag.iterators.trip_set():
            da = trip_bag.trip_cost.trip_cost_days()
            al = trip_bag.trip_cost.trip_cost_allowances()
```

```
ho = trip_bag.trip_cost.trip_cost_hotel()
pa = trip_bag.trip_cost.trip_cost_passive()
pd = trip_bag.trip_cost.trip_cost_per_diem()

mcol.add(create_one_section(trip_bag.trip.name(),
                            da + al + ho + pa + pd,
                            da,
                            al + ho + pd,
                            pa))

mcol.page()
```

Exercise 6.2 Consider selected objects

a)

Modify the existing report.
1. Create class attributes for what you want to change in the new report:

```
class Report(p.Report):

    tbh_class = bag_handler.PlanTrips
    title = "Trip costs - ExContext1"
```

2. Use them:

```
        self.set(background=sp.LightYellow)
        tbh = self.tbh_class()
        self.add(std.standard_header(self,
                                     self.title,
                                     tbh.warning))
        self.add(std.standard_footer(self))
```

b)

Then create `report_sources/crr_window_object/ExContext2.py`:

```
from carmstd import bag_handler
import report_sources.report_menu.ExContext1 as ParentReportModule

class Report(ParentReportModule.Report):
    tbh_class = bag_handler.MarkedTripsMain
    title = "Trips cost - ExContext2"

if __name__ == "__main__":
    reload(ParentReportModule)
    import nth.studio.report_generation as rg
    rg.reload_and_display_report("HTML")
```

Exercise 6.3 Number of legs in the “current” trip

One possible implementation:

```
import carmensystems.publisher.api as p

from carmstd import bag_handler

class Report(p.Report):

    def create(self):

        tbh = bag_handler.CurrentTrip()
        if tbh.warning:
            self.add(tbh.warning)
        if tbh.bag:
            n = 0
            for _ in tbh.bag.atom_set():
                n += 1
            self.add('Number of legs in "current" trip: %s' % n)
```

Exercise 6.4 List all keywords

No keyword differences.

One possible solution:

```
import os

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

from carmstd import bag_handler

class Report(p.Report):

    def create(self):

        self.set(font=p.font(face=p.MONOSPACE))
        self.add(os.path.expandvars("Release=$CarmRelease"))

        always_diff = set(["crr_identifier", "leg_identifier",
                           "minute_time", "minute_time_homebase"])

        tbh = bag_handler.CurrentLeg()
        if tbh.warning:
            self.add("Warning: %s" % tbh.warning)
        if not tbh.bag:
            return
```

```
        for lbag in tbh.bag.atom_set():
            base, = r.eval(lbag, "sp_homebase_1")
            args = ('',
                    '(1)',
                    '("%s")' % (base,),
                    '("%s","")' % (base,),
                    '("%s",01Jun2011 10:00)' % (base,),
                    '(01Jun2011 10:00, 0, 0)')

            for kw in r.keywords():
                for arg in args:
                    try:
                        expr = "%s%s" % (kw.name(), arg)
                        v, = r.eval(lbag, expr)
                        break
                    except (r.UsageError, r.ParseError, TypeError):
                        if arg == args[-1]:
                            raise

                txt = "%-40s %-28s %-15s %s" % (expr,
                                                expr in always_diff and "---" or v,
                                                v.__class__.__name__,
                                                kw.level().name().split(".")[-1])

                self.add(txt)
                self.page()

if __name__ == "__main__":
    import report_generation as rg
    rg.reload_and_display_report("HTML")
```

Exercise 7 Rule failures & constraints

Exercise 7.1 Number of rule failures per rule

Here is one possible solution. The changes are indicated with **bold text**.

```
from collections import defaultdict

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

from carmstd import bag_handler
import report_sources.include.standardreport as std

class Report(p.Report):

    def create(self):
        tbh = bag_handler.PlanCrewChains()
        violations = defaultdict(int)
        if tbh.bag:
            for chain_bag in tbh.bag.chain_set():
                for _, fail in chain_bag.rulefailures():
                    violations[fail.rule.name()] += 1

        self.setpaper(orientation=p.LANDSCAPE)

        self.add(std.standard_header(self, "Rules", tbh.warning))
        self.add(std.standard_footer(self))

        mc = self.add(p.Column())
        mc.add_header(p.Row("Violations", "Remark", "Name", "Module", "Dependency",
                      font=p.font(weight=p.BOLD)))
        for rule in sorted(r.rules(), key=r.Rule.remark):
            module_name, rule_name = (rule.name().split(".")
                                      if "." in rule.name()
                                      else ("_topmodule", rule.name()))
            level = rule.level().name().split(".")[-1]
            remark = rule.remark()
            if len(remark) > 70:
                remark = remark[:66] + " ..."
            mc.add(p.Row(violations[rule.name()] if rule.on() else "-",
                         remark,
                         rule_name,
                         module_name,
                         level))
            mc.page()
```

Exercise 7.2 Replace PDL with PRT

1. Modify the existing `CheckLegality.py` report:

```
class Report(p.Report):
    tbh_class = bag_handler.MarkedRostersMain
    title = "Check Legality"

    def create(self):
        self.tbh = self.tbh_class()

        # Define header and footer
        title = "%s%s" % (self.title,
                          " - one rule" if self.arg('rule') else "")
        self.add(standardreport.standard_header(self, title,
                                                self.tbh.warning))
```

2. Create a new report `report_sources/hidden/CheckLegalityCrewLeft.py`:

```
from carmstd import bag_handler
import report_sources.crew_window_object.CheckLegality as BaseReportModule

class Report(BaseReportModule.Report):
    tbh_class = bag_handler.MarkedRostersLeft
    title = "Check Legality (Left Selected Crew)"
```

3. Modify the menu source file `menu_scripts/Cas/StandardExtensions`

Just replace `Check_Legality_Crew_Left` with `CheckLegalityCrewLeft.py`

Exercise 7.3 Global Constraints

You find the code in:

```
/carm/academy/NiceToHaveIQ/lib/python/nth/studio/prt/global_constraints.py
```

Exercise 8 Transforms & Performance

Exercise 8.1 Performance

-

Exercise 8.2 Show all crew working on a flight leg

One possible solution:

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

from carmstd import bag_handler

def gui_date(d):
    return Dates.FDatInt2Date(d.getRep())

class Report(p.Report):

    def create(self):

        tbh = bag_handler.CurrentLeg()
        if tbh.warning:
            self.add(tbh.warning)
        if not tbh.bag:
            return
         for cleg_bag in tbh.bag.atom_set():
            if not cleg_bag.flight_duty():
                self.add("The report only works for flight legs")
                return

            self.add(p.Text("Crew working on flight %s, %s:" %
                            (cleg_bag.flight_number(),
                             gui_date(cleg_bag.leg.start_date())),
                            font=p.font(weight=p.BOLD)))

            cond = r.expr("not deadhead and not void(crr_crew_id)")
            for leg_bag in cleg_bag.equal_legs().iterators.leg_set(where=cond):
                self.add(p.Row(leg_bag.crr_crew_id(),
                               leg_bag.crew.main_func(),
```

```
                                "%s, %s" % (leg_bag.crew.surname(),
                                            leg_bag.crew.firstname())))
```

Exercise 8.3 Consider best practice

One possible solution:
(An alternative could be to create and use a new bag handler class)

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

import report_sources.include.standardreport as std
from carmstd import bag_handler

def gui_date(d):
    return Dates.FDatInt2Date(d.getRep())

class Report(p.Report):

    def create(self):

        tbh = bag_handler.MarkedLegsMain()

        npa = 0
        if tbh.bag:
            for _ in tbh.bag.atom_set(where="personal_activity"):
                npa += 1

        if npa == 1:
            warning = "One personal activity has been ignored. "
        elif npa:
            warning = "%s personal activities have been ignored. " % npa
        else:
            warning = ""

        self.add(std.standard_header(self, "Crew on leg", warning + tbh.warning))
        self.add(std.standard_footer(self))

        if not tbh.bag:
            return

        for sleg_bag in tbh.bag.crg_basic.same_leg_iterator(where="not personal_activity",
                                                            sort_by=("ground_duty",
                                                                     "ground_duty_code",
```

```
                                                        "flight_carrier",
                                                        "flight_number")):
        for n, leg_bag in enumerate(sleg_bag.atom_set()):
            if n:
                break

            if leg_bag.flight_duty():
                info = ("Flight", "%s%s" % (leg_bag.flight_carrier(),
                                            leg_bag.flight_number()))
            else:
                info = ("Ground duty", "%s" % leg_bag.ground_duty_code())

            row = p.Row(info[0],
                        p.Text(info[1],
                               font=p.font(weight=p.BOLD)),
                        gui_date(leg_bag.leg.start_date()),
                        ":")

            self.add(p.Isolate(row))

            cond = r.expr("not void(crr_crew_id)")
            for segm_bag in leg_bag.equal_legs().atom_set(where=cond,
                                                          sort_by=("deadhead",
                                                                   "crew.%main_func%",
                                                                   "crr_crew_id")):
                if segm_bag.deadhead():
                    func_info = p.Text("DH")
                else:
                    func_info = p.Text(segm_bag.crew.main_func(),
                                       font=p.font(weight=p.BOLD))

                self.add(p.Row("   ",
                               segm_bag.crr_crew_id(),
                               func_info,
                               "%s, %s" % (segm_bag.crew.surname(),
                                           segm_bag.crew.firstname())))
            self.add("")
            self.page()
```

Quick Reference

Where to put the PRT report definitions in Studio:

`$CARMUSR/lib/python/report_sources/<DIR>`

`<DIR>` could be:

`(crr|rtd|leg|crew|acrot)_window_(object|general)`

`batch_menu`

`hidden`

`report_menu`

A simple report:

```
import carmensystems.publisher.api as p


class Report(p.Report):
    def create(self):
        self.add("Hello World")
```

Objects you could `add` to the Report in `create`:

```
Text(width, height, name, border, padding,
     align, valign, colspan, rowspan, background,
     bookmark, link, action)
Image(path, width, height, name, border, padding,
      align, valign, colspan, rowspan, background,
      bookmark, link, action)
Row(*members, height, border, font, colour,
    rowspan, background)


Column(*members, width, border, font, colour,
       colspan, background)
Isolate(box, width, height, border, padding, align,
        valign, font, colour, colspan, rowspan, background)
```

How to define properties (when not trivial):

```
font : font(face=None,     # None, SERIF, SANSSERIF, MONOSPACE
            size=None,     # pt
            weight=None,   # BOLD, None
            style=None)    # ITALIC, None

border : border(left, top, right, bottom,          # pt
                inner_floor, inner_wall,           # pt
                colour)                            # RGB code
         border_all(w, colour)                     # pt, RGB
         border_frame(w, colour)                   # pt, RGB

padding: padding(left, top, right, bottom)         # pt
align: LEFT, CENTER, RIGHT
valign: TOP, CENTER, BOTTOM
bookmark: bookmark(text,       # string
                   level,      # int
                   open)       # bool
```

Rave API

Definition Objects and functions for creation

```
Constraint     : constraint(name)
Context        : context(name)
                 selected(level)
                 buffer2context(CpmBuffer)
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)
```

Methods of Definition Objects

```
name, remark          : all object
level                 : most objects
value, set_value
default_value, reset,
maxvalue, minvalue    : Parameter
on, setswitch,
resetswitch           : Rule, Constraint
members               : Group, Set, ParameterSet
add, clear, remove
reset, defaultmembers : ParameterSet
```

Evaluate

`eval([bag], *evaluators)` : returns a tuple

evaluator:

`Variable, Expression, Keyword..`

Other

Traversers: `first|last(level, computable)¨`

Predefined **levels**: `Level.atom(), Level.chain(), Level.const()`

Bag interface

```
bag = context.bag()
for iter_bag in bag.<iterator_name>(where, sort_by):
    iter_bag.arrival()
```

Rule failures

```
for ledob_bag, fe in chain_bag.rulefailures(where, sort_by):
     fe.rule.remark()
     fe.limitvalue
```

Vertical constraints

```
for ledob_bag, ce in chain_bag.constraint_evals():
     ce.constraint.remark()
     ce.limitvalue
```

Global constraints

```
for empty_bag, ce in  chain_bag.global_constraint_evals():
     ce.constraint.remark()
     ce.limitvalue
```

DWS – Get started

DWS (**Developer Workspace –** Eclipse with PyDev and CARMUSR plug-ins) is an advanced tool with many possibilities. There is much to learn. Below you just find the basics to get started. There is more to find in the command **Help> Help Contents**> **Developer Workspace User Guide** inside DWS.

Get DWS up and running

Step 1 Start Studio.

Step 2 Start DWS (**Admin Tools> Developer Workspace**).

It takes some time. An Eclipse workspace with a project for your CARMUSR is created automatically.

Note You can start DWS from the **Launcher** or from the **Special** menu too.

Interact with Studio from DWS.

Step 1 Open and run a Python file.

1. Double click on a file, e.g.
   `Carmusr Resources/`
   `python/report_sources/report_menu/HelloWorld.py`,
   in the **Project Explorer** window to open it in the editor window.
2. Now press the **Run in Studio** button  in the toolbar to run the file in Studio.
3. In the tab 

in the bottom area you have access to Studio's log file.

Start and use a Studio debug session

Step 1 Enter a port number for the debug server.
(This you just do once for the workspace).

1. Open the **Preference** window (**Window> Preferences**).

2. Select **PyDev> Run/Debug** in the left field.
3. Each debug server needs a unique debug port. For this purpose use the number associated with your user account. Compose the port number as: 500 + User account number (e.g. 50011) and enter this number in the **Port for remote debugger** field.
4. Press **OK**.

Step 2 Start debugger in Studio

1. Press the **Start debug in Studio** button  in the toolbar.

   This action starts the PyDev debug server in Eclipse and debugging of Python code in Studio. Studio also connects to the debug server.
2. Switch to the Debug perspective. It is already opened. You find it in the tab to the right of the toolbar.

Step 3 Set a break point

1. Select a Python file, e.g. `report_sources/report_menu/CoursePrt1ExRave1.py`.
2. Set a breakpoint at e.g. line 29:

   ```
   self.add( ..
   ```

   This is easiest performed by left double click in front of the line (in the grey area) in the editor. (It is also possible to right click in front of the row and select **Add breakpoint** from the pop-up menu.)
3. Generate the report (from **Studio** or with the **Run in Studio** command in DWS).
4. The execution stops at the break point.
5. Look around in the debug windows.
   e.g. **Debug**, **Variables**, **Breakpoints** and **Outline**.

   General information about the PyDev plug-in debugger can be found at: http://pydev.org/manual_adv_debugger.html
6. Try the debugging buttons  **Resume** (**F8**), **Terminate**, **Step Into** (**F5**), **Step Over** (**F6**) and **Step Return** (**F7**), located in the header of the **Debug** window.

Note It is currently not possible to quit a debug session and then start a new debug session, without first restarting Studio and DWS.

End the DWS session

Step 1 If you are stepping a file in the debugger, run this file to the end by pressing the **Resume** (**F8**) button in the debug window. Then terminate the debug session by pressing **Terminate** (**Ctrl+F2**) in the debug window

Step 2 Exit DWS (**File> Exit**).

Note The next time you start DWS the workspace and the project will still be there. The information is stored on file.

Use the command **Special> DWS> Remove DWS files…** to get information about where these files are stored. The command can also be used to remove the files. That is the first thing to do if DWS has problems to start up correctly.

Code completion and tool tips in DWS

Start DWS again. We will take a look at some useful stuff.

Step 1 Make sure you are in the **Carmusr** perspective.

Step 2 Select e.g. the file `report_sources/report_menu/CoursePrt1ExRave1.py` in the editor.

Step 3 Watch the tool tips for e.g. p.`Report`, p.`Text`, `p`, `r` and `r.eval`, by moving the mouse over the code in the editor window. Also note that all usage of the object in the file is highlighted when you double click at the code.

Step 4 DWS provides code completion. To try this functionality add e.g. the code sequence below to the end of the method `create` in an existing report definition. Do not copy and paste, instead type and watch the suggested code completion while typing and select from the drop-down list, by double click (or press Enter). Also note the tool tips for the selected entity.

```
self.add(p.Text("Text Field",
                border=p.border_all(),
                tooltip="text field"))
```

Hint: To force code completion press **Ctrl-Space**.

Short-cuts in DWS

Knowing short-cuts is essential for efficient usage of DWS.

Step 1 Inspect the definition of keyboard short-cuts.
It is good to know where you find them. Here you can also redefine them if you want.

Use **Windows> Preferences**, select **General> Keys** in the left field.

Step 2 Some very useful short-cuts

Ctrl Shift L – List all short-cuts

F3 – Jump to the definition of the selected object.

Ctrl Shift F3 – Jump to a Rave definition from Python code.

Note Changed from **F7** to **Shift F3** in CARMSYS 21.12 to the current value in CARMSYS 22.2.

Alt Left/Right Arrow – Jump back and forth in history.

Ctrl Q – Jump to last the change.

Ctrl H – Search

Ctrl Shift G – Find where the selected object is defined or used.

Ctrl E – List all open editors

Ctrl 1 – Quick code fix

Shift Ctrl F – Improve Python code format

Ctrl 3 – Comment Python code

Shift Ctrl 3 – Uncomment Python code

Ctrl I – Correct the indentation of Python code.

Ctrl D – Delete a row.

Ctrl Shift Delete – Delete to end of line

Ctrl Space – Normal code completion

Shift Space – Completion of Rave definition in Python code.