Introduction to HIPE
&
HIPE Help

Carolyn M^cCoey, University of Waterloo
Adwin Boogert, NHSC/IPAC, Caltech

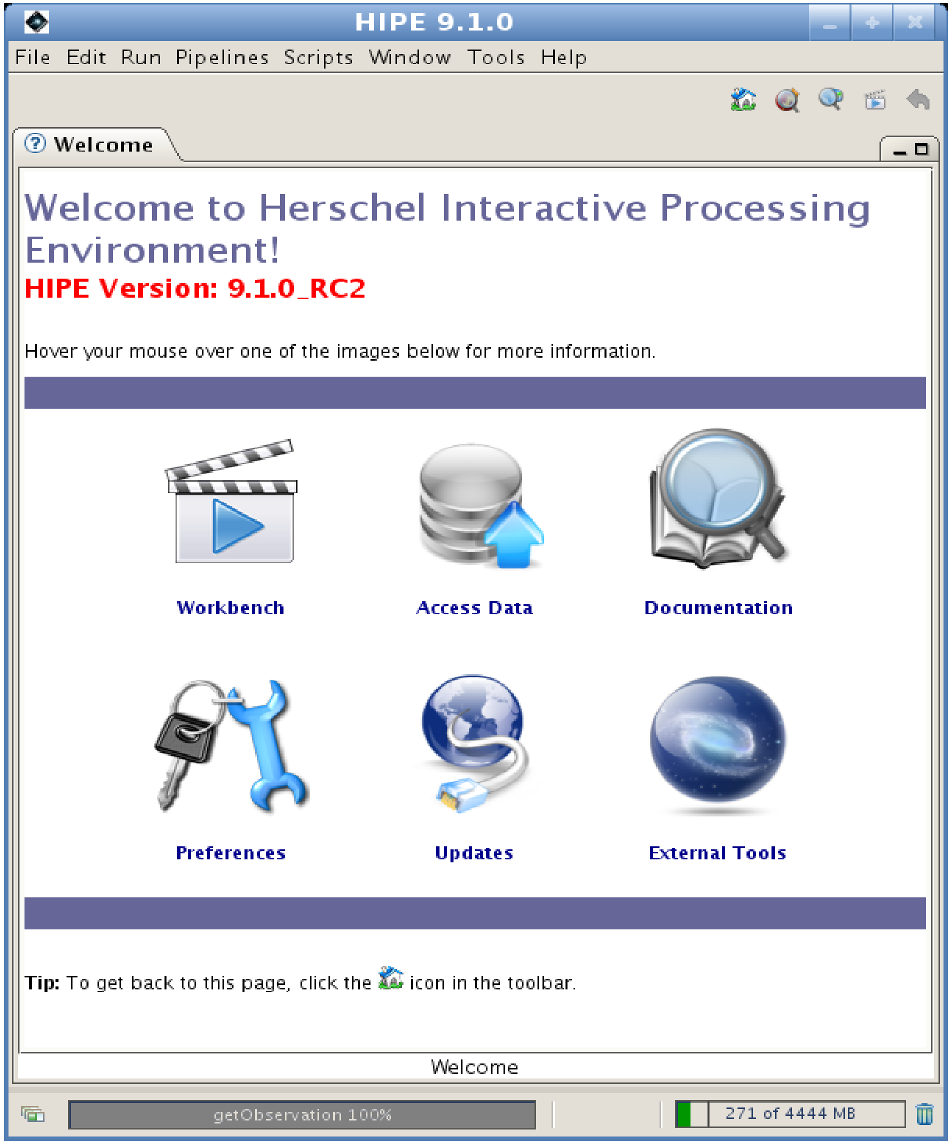
HIFI+Herschel Data Processing Editorial Board

Welcome to HIPE!

What is HIPE?

Herschel **I**nteractive **P**rocessing **E**nvironment

What is HIPE?



HIPE 9.1.0

File Edit Run Pipelines Scripts Window Tools Help

Welcome

Welcome to Herschel Interactive Processing Environment!

HIPE Version: 9.1.0_RC2

Hover your mouse over one of the images below for more information.

Tip: To get back to this page, click the icon in the toolbar.

Welcome

getObservation 100%

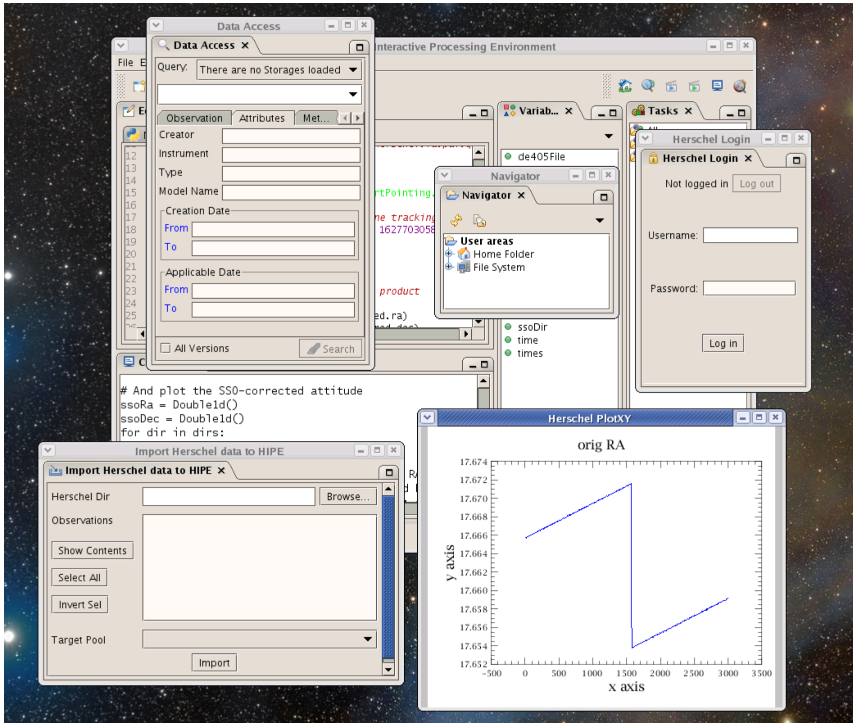
271 of 4444 MB

- HIPE is the software used by engineers, calibration scientists and astronomers to reduce, visualize and analyze Herschel data of the PACS, SPIRE and HIFI instruments:
  - Interactively
  - Automatically: jython scripting
  - In the background, without user interface: “jylaunch”
- For a high level overview of HIPE development, see the presentation by Stephan Ott this morning.

Hype about HIPE

An integrated graphical environment

- HIPE brings several applications together under a common, consistent interface. From data retrieval to plotting, from image analysis to scripting, powerful utilities are one click away.
- From raw data fresh off the Herschel Archive to publication-ready plots, all you need to get science out of your observations.



Query: There are no Storages loaded

Creator
Instrument
Type
Model Name
Creation Date
From
To
Applicable Date
From
To

de405File
ssoDir
time
times

Not logged in

Username:

Password:

Herschel Dir

Observations

Target Pool

```
# And plot the SSO-corrected attitude
ssoRa = Double1d()
ssoDec = Double1d()
for dir in dirs:
```

Hype about HIPE

The power of Java and Jython

- HIPE is based on Java, one of the most popular programming languages. The multi-platform nature of Java allows HIPE to work flawlessly under Window, Mac OS and many Linux and UNIX flavours.
- Jython is the Java-based version of Python, used worldwide for quick development of complex applications.

```
tFile = mpsDir + "[illegible]"
5File = mpsDir + "ascp2007-2020.405"
ir = mpsDir + "horizons"
erides = Ephemerides(orbitFile, de405File)
id = 2000018 # Melpomene
zons = Horizons(ssoDir, ephemerides)
 = Sso.getDirections(pp, fineTimes, horizons, nai

d plot the SSO-corrected attitude
a = Double1d()
ec = Double1d()
dir in dirs:
oRa.append(dir.raDegrees)
```

HIPE Help

A *very* powerful tool with l**ots** of documentation to help you:

•Help for new HIPE and HIFI users:

- Help-->Help Contents: Herschel Data Analysis Guide
- Help-->Help Contents: HIFI Data Reduction Guide
  - Contains launch pad for new users
- Help-->Video tutorials
- twitter.com/learnhipe

•Help on more advanced topics:

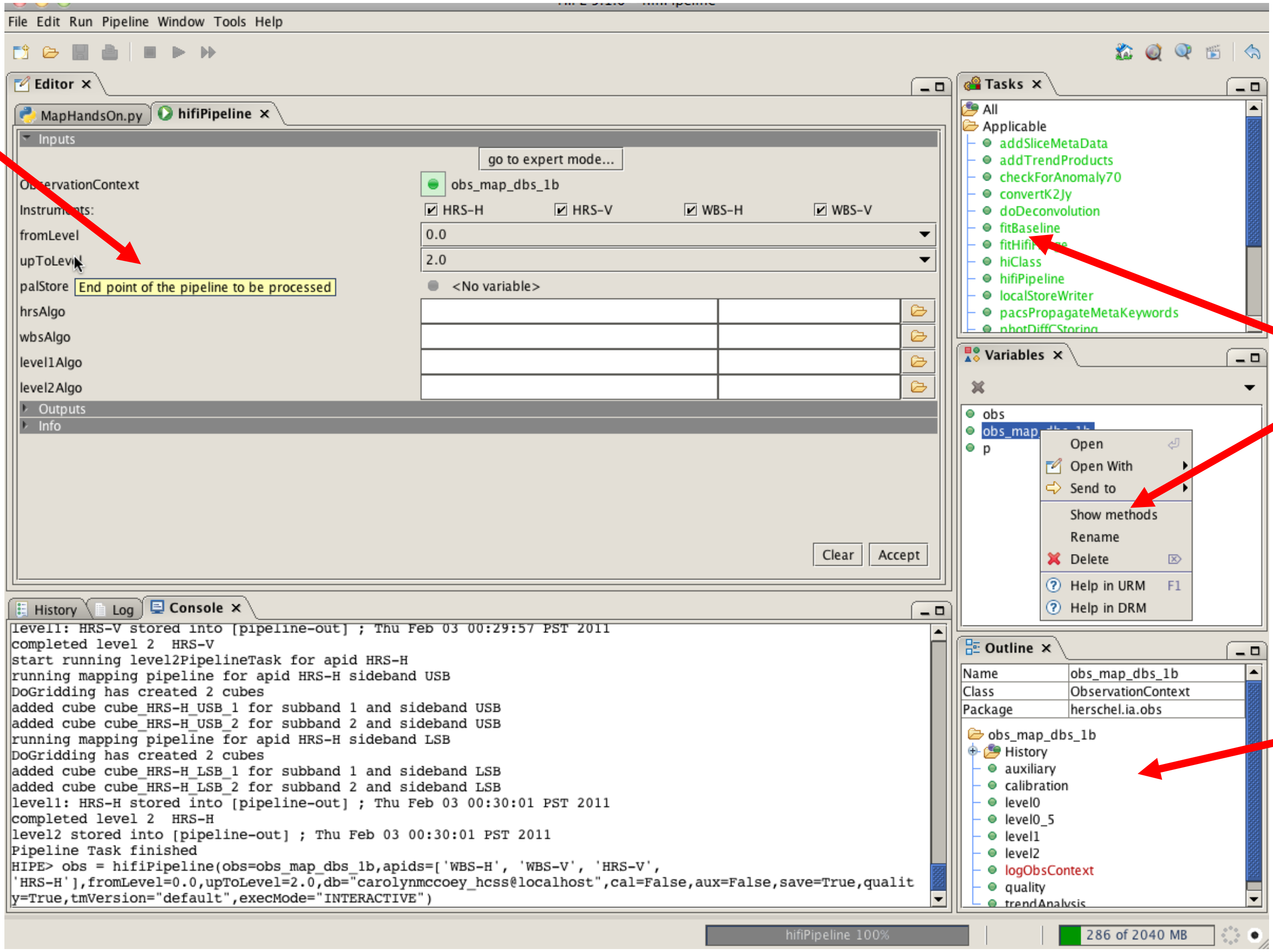
- Help-->Help Contents: HIFI Pipeline Specification
- Help-->Help Contents: Herschel and HIFI Reference Manuals

But…

HIPE Help

Nobody wants to read manuals…

Manual-free help



ObservationContext ● obs_map_dbs_1b

Instruments: ☑ HRS-H ☑ HRS-V ☑ WBS-H ☑ WBS-V

fromLevel 0.0

upToLevel 2.0

palStore End point of the pipeline to be processed ● <No variable>

hrsAlgo

wbsAlgo

level1Algo

level2Algo

```
level1: HRS-V stored into [pipeline-out] ; Thu Feb 03 00:29:57 PST 2011
completed level 2  HRS-V
start running level2PipelineTask for apid HRS-H
running mapping pipeline for apid HRS-H sideband USB
DoGridding has created 2 cubes
added cube cube_HRS-H_USB_1 for subband 1 and sideband USB
added cube cube_HRS-H_USB_2 for subband 2 and sideband USB
running mapping pipeline for apid HRS-H sideband LSB
DoGridding has created 2 cubes
added cube cube_HRS-H_LSB_1 for subband 1 and sideband LSB
added cube cube_HRS-H_LSB_2 for subband 2 and sideband LSB
level1: HRS-H stored into [pipeline-out] ; Thu Feb 03 00:30:01 PST 2011
completed level 2  HRS-H
level2 stored into [pipeline-out] ; Thu Feb 03 00:30:01 PST 2011
Pipeline Task finished
HIPE> obs = hifiPipeline(obs=obs_map_dbs_1b,apids=['WBS-H', 'WBS-V', 'HRS-V',
'HRS-H'],fromLevel=0.0,upToLevel=2.0,db="carolynmccoey_hcss@localhost",cal=False,aux=False,save=True,qualit
y=True,tmVersion="default",execMode="INTERACTIVE")
```

Name	obs_map_dbs_1b
Class	ObservationContext
Package	herschel.ia.obs

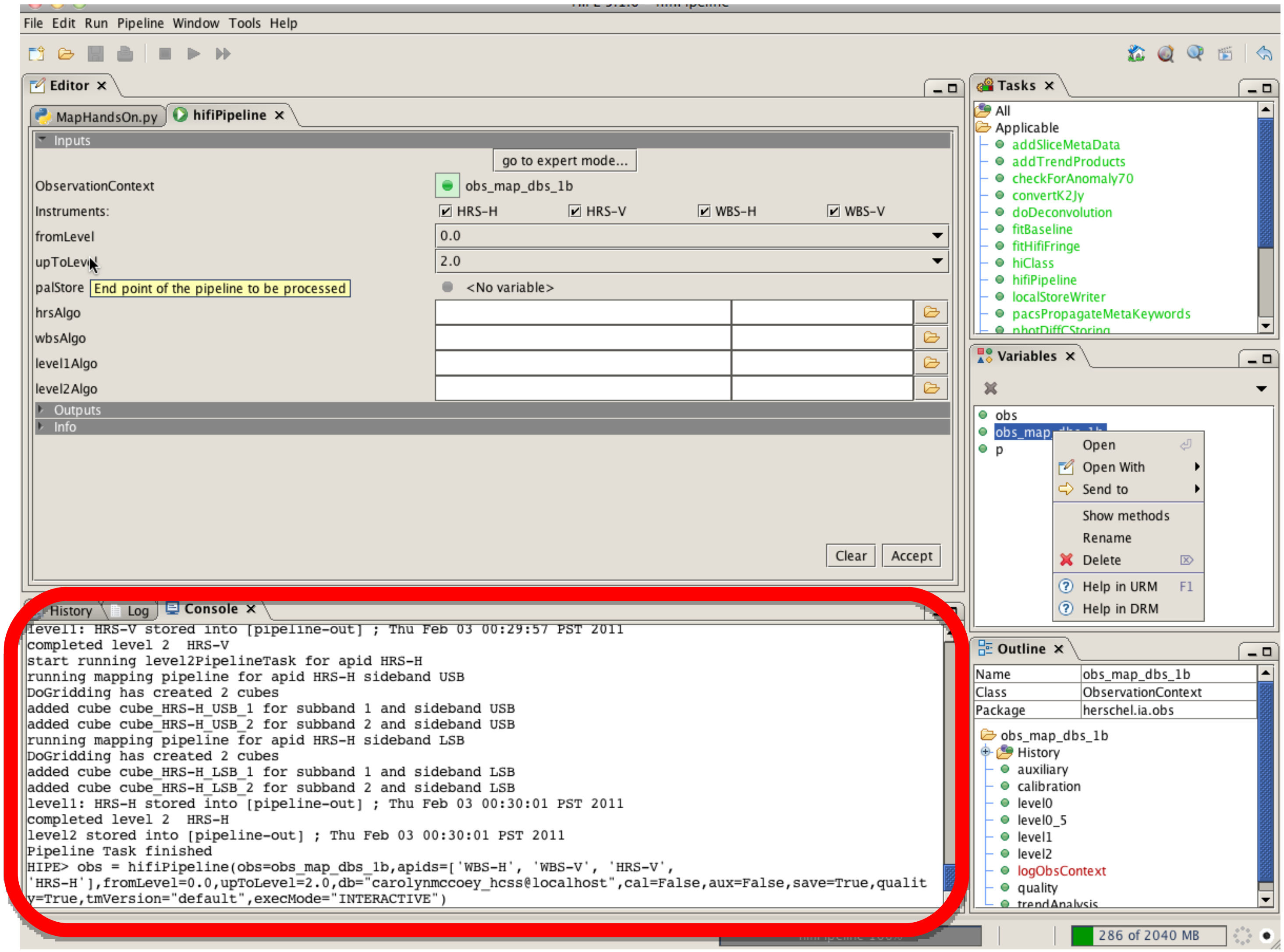
- obs_map_dbs_1b
  - History
  - auxiliary
  - calibration
  - level0
  - level0_5
  - level1
  - level2
  - logObsContext
  - quality
  - trendAnalysis

Tool Tips

Applicable Tasks

Outline

Echo to Console



ObservationContext: obs_map_dbs_1b

Instruments: ☑ HRS-H ☑ HRS-V ☑ WBS-H ☑ WBS-V

fromLevel: 0.0

upToLevel: 2.0

palStore: End point of the pipeline to be processed <No variable>

hrsAlgo

wbsAlgo

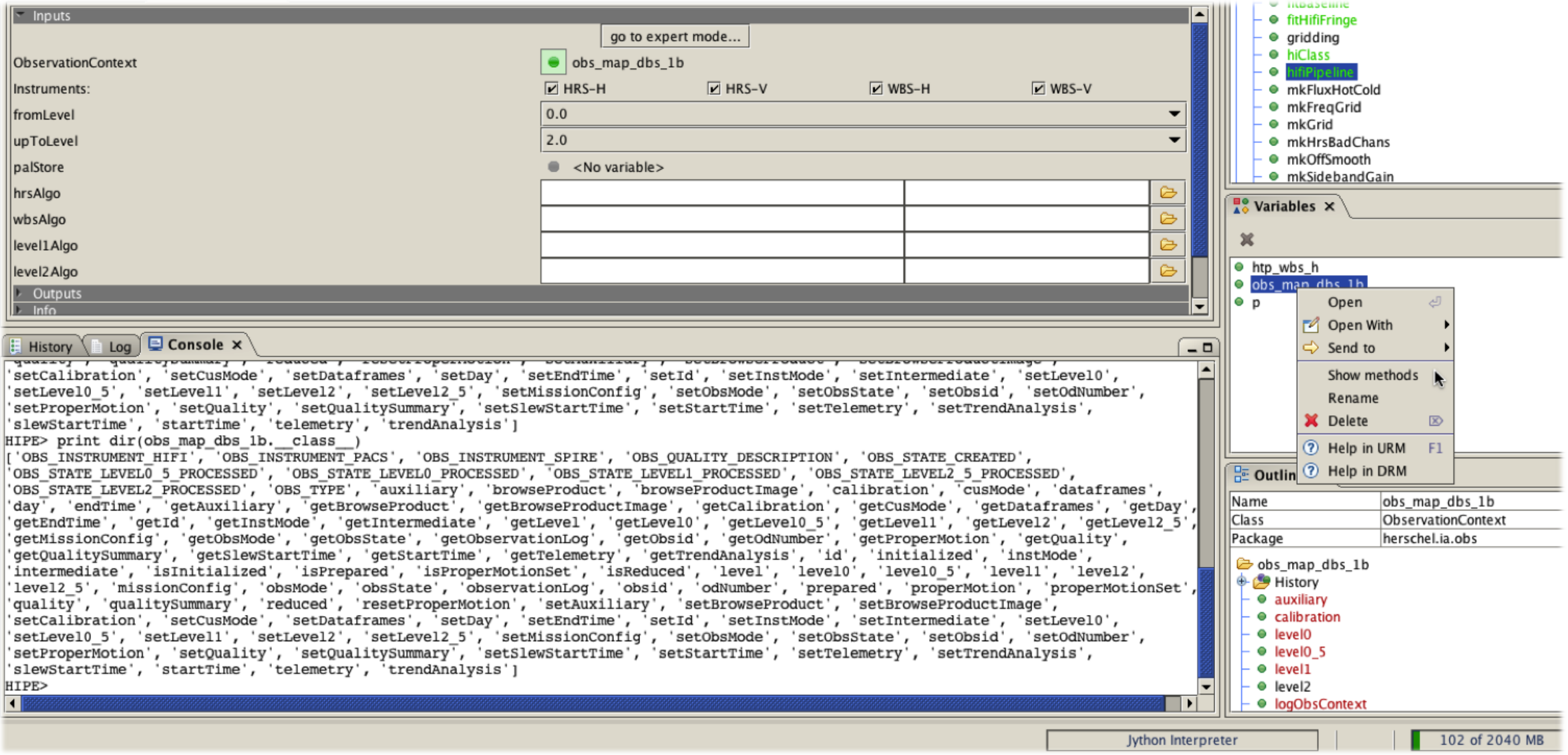
level1Algo

level2Algo

Name	obs_map_dbs_1b
Class	ObservationContext
Package	herschel.ia.obs

```
level1: HRS-V stored into [pipeline-out] ; Thu Feb 03 00:29:57 PST 2011
completed level 2  HRS-V
start running level2PipelineTask for apid HRS-H
running mapping pipeline for apid HRS-H sideband USB
DoGridding has created 2 cubes
added cube cube_HRS-H_USB_1 for subband 1 and sideband USB
added cube cube_HRS-H_USB_2 for subband 2 and sideband USB
running mapping pipeline for apid HRS-H sideband LSB
DoGridding has created 2 cubes
added cube cube_HRS-H_LSB_1 for subband 1 and sideband LSB
added cube cube_HRS-H_LSB_2 for subband 2 and sideband LSB
level1: HRS-H stored into [pipeline-out] ; Thu Feb 03 00:30:01 PST 2011
completed level 2  HRS-H
level2 stored into [pipeline-out] ; Thu Feb 03 00:30:01 PST 2011
Pipeline Task finished
HIPE> obs = hifiPipeline(obs=obs_map_dbs_1b,apids=['WBS-H', 'WBS-V', 'HRS-V',
'HRS-H'],fromLevel=0.0,upToLevel=2.0,db="carolynmccoey_hcss@localhost",cal=False,aux=False,save=True,qualit
y=True,tmVersion="default",execMode="INTERACTIVE")
```

“Show methods”



ObservationContext ● obs_map_dbs_1b

Instruments: ☑ HRS-H ☑ HRS-V ☑ WBS-H ☑ WBS-V

fromLevel 0.0

upToLevel 2.0

palStore ● <No variable>

hrsAlgo

wbsAlgo

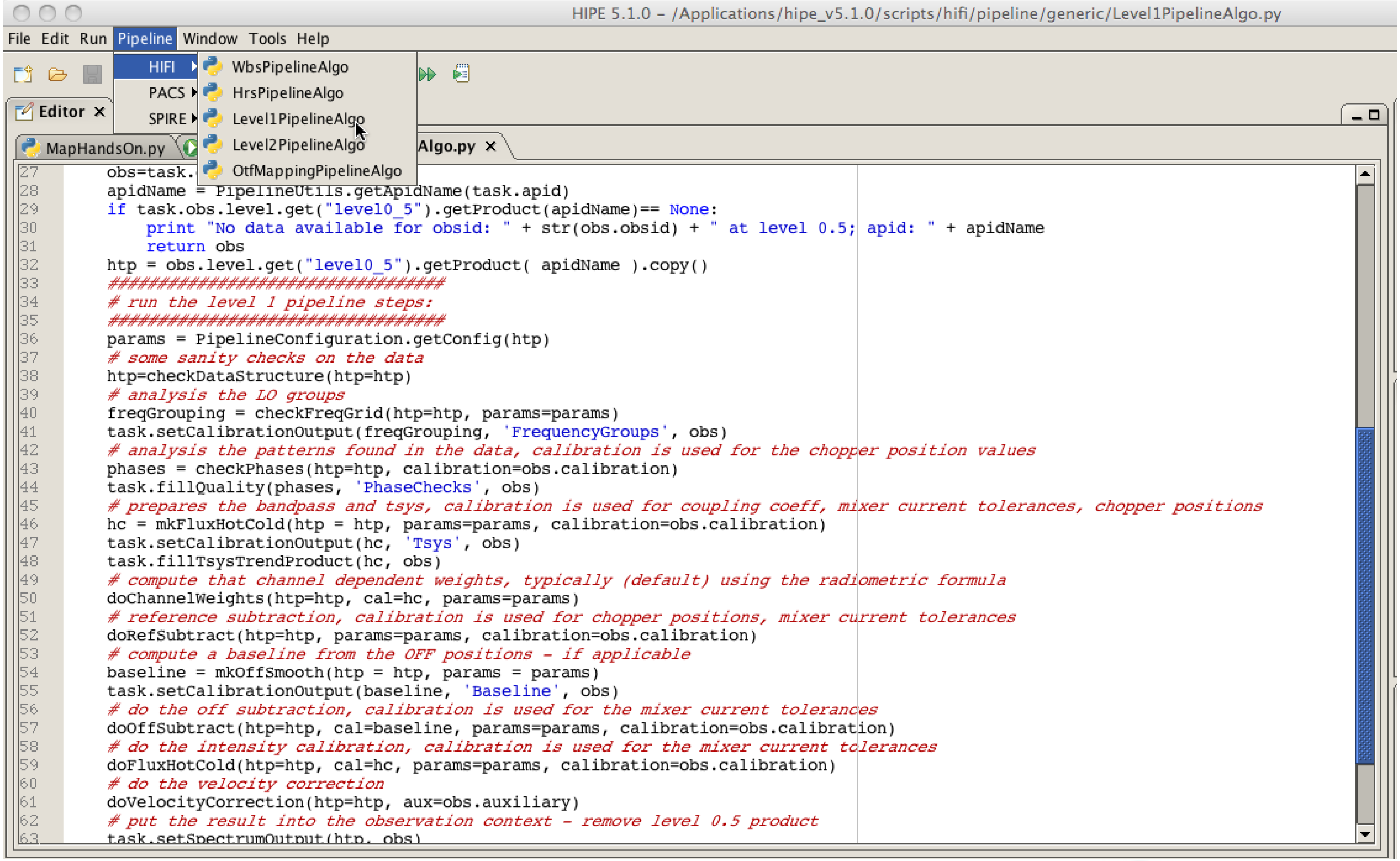
level1Algo

level2Algo

```
'setCalibration', 'setCusMode', 'setDataframes', 'setDay', 'setEndTime', 'setId', 'setInstMode', 'setIntermediate', 'setLevel0',
'setLevel0_5', 'setLevel1', 'setLevel2', 'setLevel2_5', 'setMissionConfig', 'setObsMode', 'setObsState', 'setObsid', 'setOdNumber',
'setProperMotion', 'setQuality', 'setQualitySummary', 'setSlewStartTime', 'setStartTime', 'setTelemetry', 'setTrendAnalysis',
'slewStartTime', 'startTime', 'telemetry', 'trendAnalysis']
HIPE> print dir(obs_map_dbs_1b.__class__)
['OBS_INSTRUMENT_HIFI', 'OBS_INSTRUMENT_PACS', 'OBS_INSTRUMENT_SPIRE', 'OBS_QUALITY_DESCRIPTION', 'OBS_STATE_CREATED',
'OBS_STATE_LEVEL0_5_PROCESSED', 'OBS_STATE_LEVEL0_PROCESSED', 'OBS_STATE_LEVEL1_PROCESSED', 'OBS_STATE_LEVEL2_5_PROCESSED',
'OBS_STATE_LEVEL2_PROCESSED', 'OBS_TYPE', 'auxiliary', 'browseProduct', 'browseProductImage', 'calibration', 'cusMode', 'dataframes',
'day', 'endTime', 'getAuxiliary', 'getBrowseProduct', 'getBrowseProductImage', 'getCalibration', 'getCusMode', 'getDataframes', 'getDay',
'getEndTime', 'getId', 'getInstMode', 'getIntermediate', 'getLevel', 'getLevel0', 'getLevel0_5', 'getLevel1', 'getLevel2', 'getLevel2_5',
'getMissionConfig', 'getObsMode', 'getObsState', 'getObservationLog', 'getObsid', 'getOdNumber', 'getProperMotion', 'getQuality',
'getQualitySummary', 'getSlewStartTime', 'getStartTime', 'getTelemetry', 'getTrendAnalysis', 'id', 'initialized', 'instMode',
'intermediate', 'isInitialized', 'isPrepared', 'isProperMotionSet', 'isReduced', 'level', 'level0', 'level0_5', 'level1', 'level2',
'level2_5', 'missionConfig', 'obsMode', 'obsState', 'observationLog', 'obsid', 'odNumber', 'prepared', 'properMotion', 'properMotionSet',
'quality', 'qualitySummary', 'reduced', 'resetProperMotion', 'setAuxiliary', 'setBrowseProduct', 'setBrowseProductImage',
'setCalibration', 'setCusMode', 'setDataframes', 'setDay', 'setEndTime', 'setId', 'setInstMode', 'setIntermediate', 'setLevel0',
'setLevel0_5', 'setLevel1', 'setLevel2', 'setLevel2_5', 'setMissionConfig', 'setObsMode', 'setObsState', 'setObsid', 'setOdNumber',
'setProperMotion', 'setQuality', 'setQualitySummary', 'setSlewStartTime', 'setStartTime', 'setTelemetry', 'setTrendAnalysis',
'slewStartTime', 'startTime', 'telemetry', 'trendAnalysis']
HIPE>
```

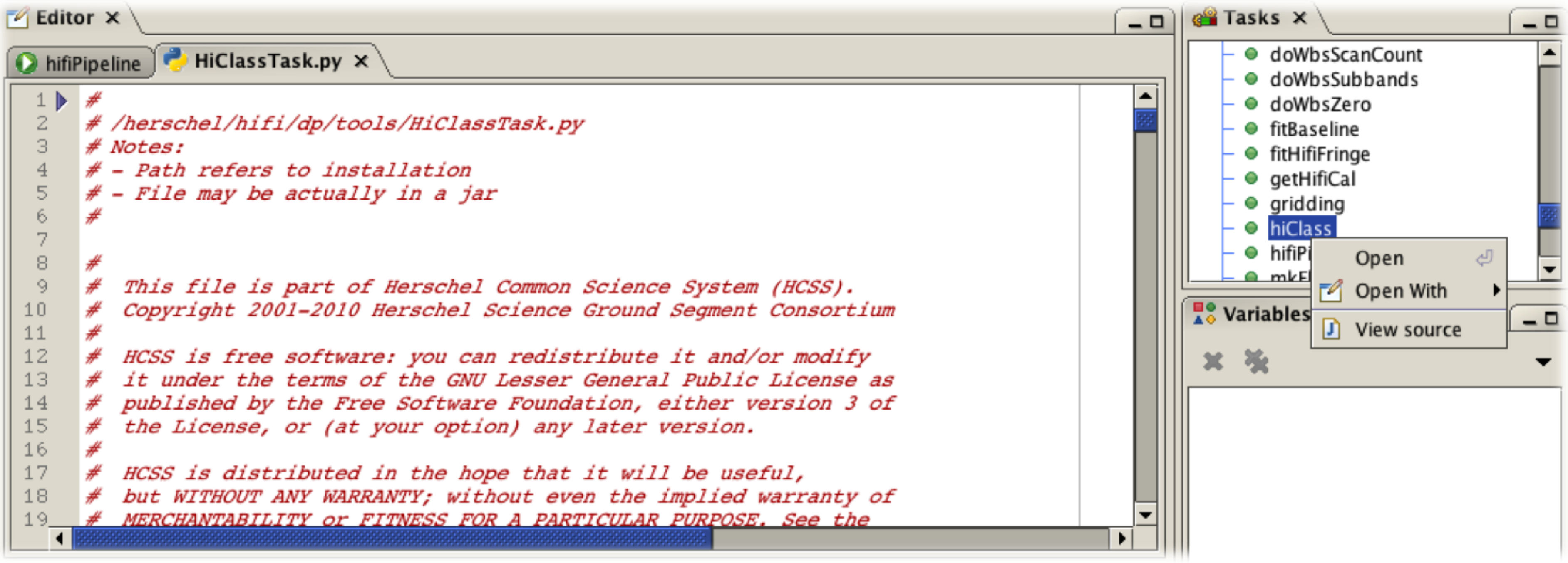
Name	obs_map_dbs_1b
Class	ObservationContext
Package	herschel.ia.obs

Commented Pipeline Scripts



```
    obs=task.
    apidName = PipelineUtils.getApidName(task.apid)
    if task.obs.level.get("level0_5").getProduct(apidName)== None:
        print "No data available for obsid: " + str(obs.obsid) + " at level 0.5; apid: " + apidName
        return obs
    htp = obs.level.get("level0_5").getProduct( apidName ).copy()
    ##################################
    # run the level 1 pipeline steps:
    ##################################
    params = PipelineConfiguration.getConfig(htp)
    # some sanity checks on the data
    htp=checkDataStructure(htp=htp)
    # analysis the LO groups
    freqGrouping = checkFreqGrid(htp=htp, params=params)
    task.setCalibrationOutput(freqGrouping, 'FrequencyGroups', obs)
    # analysis the patterns found in the data, calibration is used for the chopper position values
    phases = checkPhases(htp=htp, calibration=obs.calibration)
    task.fillQuality(phases, 'PhaseChecks', obs)
    # prepares the bandpass and tsys, calibration is used for coupling coeff, mixer current tolerances, chopper positions
    hc = mkFluxHotCold(htp = htp, params=params, calibration=obs.calibration)
    task.setCalibrationOutput(hc, 'Tsys', obs)
    task.fillTsysTrendProduct(hc, obs)
    # compute that channel dependent weights, typically (default) using the radiometric formula
    doChannelWeights(htp=htp, cal=hc, params=params)
    # reference subtraction, calibration is used for chopper positions, mixer current tolerances
    doRefSubtract(htp=htp, params=params, calibration=obs.calibration)
    # compute a baseline from the OFF positions - if applicable
    baseline = mkOffSmooth(htp = htp, params = params)
    task.setCalibrationOutput(baseline, 'Baseline', obs)
    # do the off subtraction, calibration is used for the mixer current tolerances
    doOffSubtract(htp=htp, cal=baseline, params=params, calibration=obs.calibration)
    # do the intensity calibration, calibration is used for the mixer current tolerances
    doFluxHotCold(htp=htp, cal=hc, params=params, calibration=obs.calibration)
    # do the velocity correction
    doVelocityCorrection(htp=htp, aux=obs.auxiliary)
    # put the result into the observation context - remove level 0.5 product
    task.setSpectrumOutput(htp, obs)
```

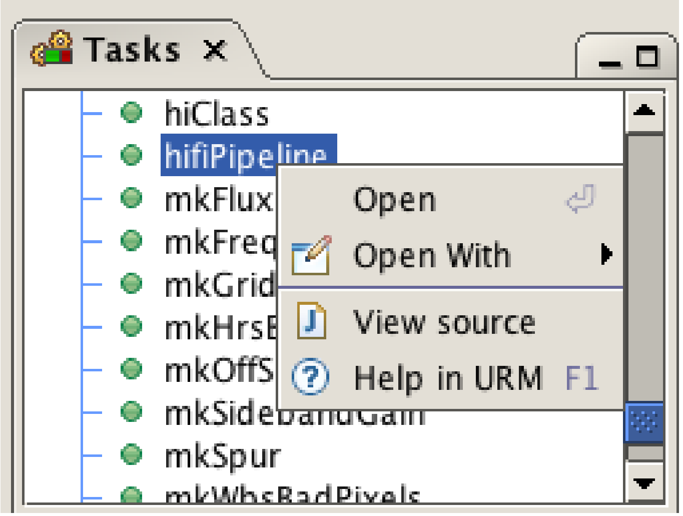
View source



```
#
# /herschel/hifi/dp/tools/HiClassTask.py
# Notes:
# - Path refers to installation
# - File may be actually in a jar
#

#
#  This file is part of Herschel Common Science System (HCSS).
#  Copyright 2001-2010 Herschel Science Ground Segment Consortium
#
#  HCSS is free software: you can redistribute it and/or modify
#  it under the terms of the GNU Lesser General Public License as
#  published by the Free Software Foundation, either version 3 of
#  the License, or (at your option) any later version.
#
#  HCSS is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

Help in URM



URM = Users Reference **Manual**

Contains information about the main classes and tasks you can use in your scripts