

PYP

API Documentation

August 5, 2011

Contents

Contents	1
1 Script script-pyp	2
1.1 Variables	2
1.2 Class PypCustom	2
1.3 Class PowerPipeListCustom	2
1.4 Class PypStrCustom	2
1.5 Class PypListCustom	2
1.6 Class PypFunctionCustom	2
1.7 Class Colors	2
1.7.1 Methods	3
1.7.2 Properties	3
1.7.3 Class Variables	3
1.8 Class PowerPipeList	3
1.8.1 Methods	4
1.8.2 Properties	6
1.8.3 Class Variables	7
1.9 Class PypStr	7
1.9.1 Methods	7
1.9.2 Properties	8
1.10 Class PypList	8
1.10.1 Methods	9
1.10.2 Properties	9
1.10.3 Class Variables	9
1.11 Class Pyp	9
1.11.1 Methods	10
1.11.2 Properties	19
1.11.3 Instance Variables	19

1 Script script-pyp

1.1 Variables

Name	Description
usage	Value: ' \npyp is a python-centric command line text manipula...
parser	Value: optparse.OptionParser(usage)
manual	Value: ' \n=====...'
unmodified_config	Value: '\n\n#!/usr/local/bin/python\n# This must be saved in sam...
__package__	Value: None
args	Value: []
options	Value: <Values at 0x101bf38: {'macro_save_name': None, 'execute'...

1.2 Class PypCustom

1.3 Class PowerPipeListCustom

Known Subclasses: script-pyp.PowerPipeList

1.4 Class PypStrCustom

Known Subclasses: script-pyp.PypStr

1.5 Class PypListCustom

Known Subclasses: script-pyp.PypList

1.6 Class PypFunctionCustom

1.7 Class Colors

object ┌
└─ script-pyp.Colors

defines basic color scheme

1.7.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

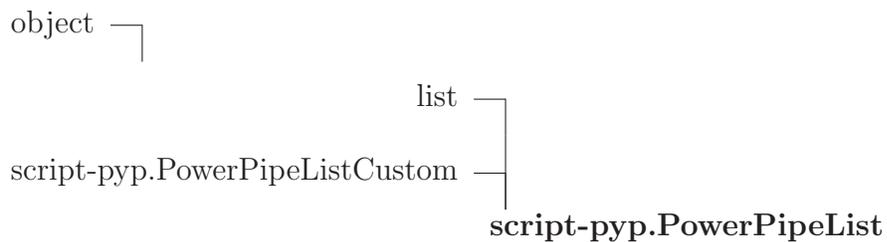
1.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

1.7.3 Class Variables

Name	Description
OFF	Value: <code>'\x1b[0m'</code>
RED	Value: <code>'\x1b[31m'</code>
GREEN	Value: <code>'\x1b[32m'</code>
YELLOW	Value: <code>'\x1b[33m'</code>
MAGENTA	Value: <code>'\x1b[35m'</code>
CYAN	Value: <code>'\x1b[36m'</code>
WHITE	Value: <code>'\x1b[37m'</code>
BLUE	Value: <code>'\x1b[34m'</code>
BOLD	Value: <code>'\x1b[1m'</code>
COLORS	Value: <code>['\x1b[0m', '\x1b[31m', '\x1b[32m', '\x1b[33m', '\x1b[35m...</code>

1.8 Class PowerPipeList



defines pp object, allows manipulation of entire input using python list methods

1.8.1 Methods

`__init__`(*self*, **args*)

x.`__init__`(...) initializes *x*; see *x*.`__class__`.`__doc__` for signature

Return Value

new list

Overrides: `object.__init__` `exitit`(inherited documentation)

`divide`(*self*, *n_split*)

splits list into subarrays with *n_split* members

Parameters

n_split: number of members produced by split

(*type*=int @return : new array split up by *n_split* @rtype : list<str>)

`delimit`(*self*, *delimiter*)

splits up array based on delimited instead of newlines

Parameters

delimiter: delimiter used for split

(*type*=str)

Return Value

new string split by *delimiter* and joined by ' '

(*type*=list<str>)

`oneliner`(*self*, *delimiter*=' ')

combines list to one line with optional delimiter

Parameters

delimiter: delimiter used for joining to one line

(*type*=str)

Return Value

one line output joined by *delimiter*

(*type*=list<str>)

uniqer(*self*)

returns only unique elements from list

Return Value

unique items

*(type=list<str>)***get_strings**(*self*, *iterables*)

returns a list of strings from nested lists

Parameters**iterables**: nested lists containing strs or PypStrs*(type=list)***Return Value**

unnested list of strings

*(type=list<str>)***unlister**(*self*)

splits a list into one element per line

Parameters**self**: nested list*(type=list<str>)***Return Value**

unnested list

*(type=list<str>)***after**(*self*, *target*, *after_n=1*)

consolidates after_n lines after matching target text to 1 line

Parameters**target**: target string to find*(type=str)***after_n**: number of lines to consolidate*(type=int)***Return Value**

list of after_n members

(type=list<str>)

before(*self*, *target*, *before_n=1*)

consolidates *before_n* lines before matching target text to 1 line

Parameters

target: target string to find
(*type=str*)

before_n: number of lines to consolidate
(*type=int*)

Return Value

list of *before_n* members
(*type=list<str>*)

matrix(*self*, *target*, *matrix_n=1*)

consolidates *matrix_n* lines surrounding matching target text to 1 line

Parameters

target: target string to find
(*type=str*)

matrix_n: number of lines to consolidate
(*type=int*)

Return Value

list of *matrix_n* members
(*type=list<str>*)

Inherited from list

`__add__()`, `__contains__()`, `__delitem__()`, `__delslice__()`, `__eq__()`, `__ge__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__gt__()`, `__iadd__()`, `__imul__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__mul__()`, `__ne__()`, `__new__()`, `__repr__()`, `__reversed__()`, `__rmul__()`, `__setitem__()`, `__setslice__()`, `__sizeof__()`, `append()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`, `reverse()`, `sort()`

Inherited from object

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`

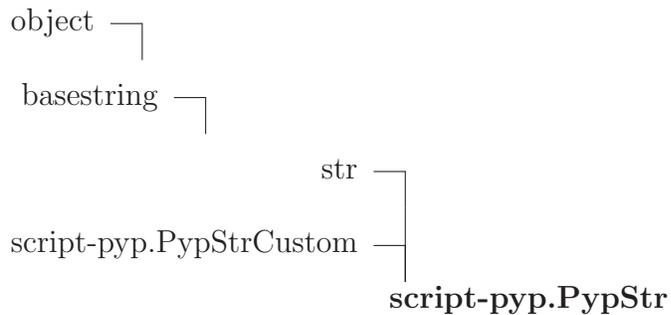
1.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

1.8.3 Class Variables

Name	Description
<i>Inherited from list</i>	
<code>__hash__</code>	

1.9 Class PypStr



defines p string object, allows manipulation of input line by line using python string methods

1.9.1 Methods

<code>__init__(self, *args)</code> <code>x.__init__(...)</code> initializes x; see <code>x.__class__.__doc__</code> for signature Overrides: <code>object.__init__</code> extit(inherited documentation)

<code>trim(self)</code> returns everything but the last directory/file Parameters <code>self</code> : directory path (<i>type=</i> <code>str</code>) Return Value directory path missing without last directory/file (<i>type=</i> <code>PypStr</code>)

kill(*self*, *to_kill*)

replaces to_kill with " in string

Parameters

to_kill: string to remove
(*type=*str)

Return Value

string without to_kill
(*type=PypStr*)

Inherited from str

__add__(), __contains__(), __eq__(), __format__(), __ge__(), __getattr__(), __getitem__(), __getnewargs__(), __getslice__(), __gt__(), __hash__(), __le__(), __len__(), __lt__(), __mod__(), __mul__(), __ne__(), __new__(), __repr__(), __rmod__(), __rmul__(), __sizeof__(), __str__(), capitalize(), center(), count(), decode(), encode(), endswith(), expandtabs(), find(), format(), index(), isalnum(), isalpha(), isdigit(), islower(), isspace(), istitle(), isupper(), join(), ljust(), lower(), lstrip(), partition(), replace(), rfind(), rindex(), rjust(), rpartition(), rsplit(), rstrip(), split(), splitlines(), startswith(), strip(), swapcase(), title(), translate(), upper(), zfill()

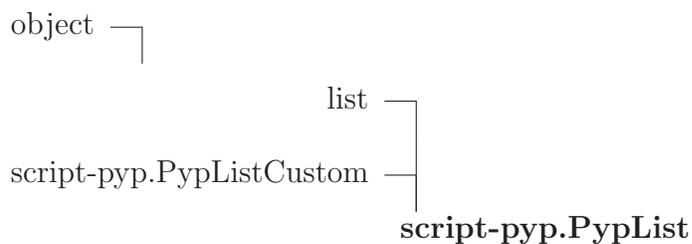
Inherited from object

__delattr__(), __reduce__(), __reduce_ex__(), __setattr__(), __subclasshook__()

1.9.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

1.10 Class PypList



defines p list object, allows manipulation of input line by line using python list methods

1.10.1 Methods

```
__init__(self, *args)
```

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Return Value
new list

Overrides: object.__init__ extit(inherited documentation)

Inherited from list

```
__add__(), __contains__(), __delitem__(), __delslice__(), __eq__(), __ge__(), __getattr__(),
__getitem__(), __getslice__(), __gt__(), __iadd__(), __imul__(), __iter__(), __le__(), __len__(),
__lt__(), __mul__(), __ne__(), __new__(), __repr__(), __reversed__(), __rmul__(), __setitem__(),
__setslice__(), __sizeof__(), append(), count(), extend(), index(), insert(), pop(), re-
move(), reverse(), sort()
```

Inherited from object

```
__delattr__(), __format__(), __reduce__(), __reduce_ex__(), __setattr__(), __str__(), __subclasshook__()
```

1.10.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

1.10.3 Class Variables

Name	Description
<i>Inherited from list</i>	
__hash__	

1.11 Class Pyp

```
object ┌
      │
      └─ script-pyp.Pyp
```

pyp engine. manipulates input stream using python methods

1.11.1 Methods

`__init__(self)`

`x.__init__(...)` initializes `x`; see `x.__class__.__doc__` for signature

Overrides: `object.__init__` extit(inherited documentation)

`get_custom_macro_paths(self)`

returns customized paths to macro files if they are setup

`cmds_split(self, cmds, macros)`

splits total command array based on pipes taking into account quotes, parantheses and escapes. returns array of commands that will be processed procedurally. Substitutes macros without executable commands.

Parameters

`cmds`: user supplied command set
(*type=list<str>*)

`macros`: user defined macros
(*type=dict<str:dict>*)

Return Value

list of commands to be evaluated
(*type=list<str>*)

`load_macros(self, macro_path)`

loads macro file; returns macros dict

Parameters

`macro_path`: file path to macro file
(*type=str*)

Return Value

dictionary of user defined macros
(*type=dict<str:dict>*)

write_macros(*self, macros, macro_path, cmds*)

writes macro file

Parameters

macros: dictionary of user defined macros
(*type=dict<str:dict>*)

macro_path: file path to macro file
(*type=str*)

cmds: commands to be saved as a macro
(*type=list<str>*)

delete_macros(*self, macros, macro_path*)

deletes macro from file

Parameters

macros: dictionary of user defined macros
(*type=dict<str:dict>*)

macro_path: file path to macro file
(*type=str*)

list_macros(*self, macros*)

prints out formatted macros, takes dictionary macros as input

Parameters

macros: dictionary of user defined macros
(*type=dict<str:dict>*)

load_file(*self*)

loads file for pyp processing

Return Value

file data
(*type=list<str>*)

shell(*self*, *command*)

executes a shell commands, returns output in array sh

Parameters

command: shell command to be evaluated
(*type=*str)

Return Value

output of shell command
(*type=*list<str>)

shelld(*self*, *command*, **args*)

executes a shell commands, returns output in dictionary based on args

Parameters

command: shell command to be evaluated
(*type=*str)

args: optional delimiter. default is ":".
(*type=*list)

Return Value

hashed output of shell command based on delimiter
(*type=*dict<str:str>)

keep(*self*, **args*)

keeps lines based on string matches

Parameters

args: strings to search for
(*type=*list<str>)

Return Value

True if any of the strings are found else False
(*type=*bool)

lose(*self*, **args*)

removes lines based on string matches

Parameters

args: strings to search for
(type=list<str>)

Return Value

True if any of the strings are not found else False
(type=bool)

array_tracer(*self*, *input*, *power_pipe=False*)

generates colored, numbered output for lists and dictionaries and other types

Parameters

input: one line of input from evaluated pyp command
(type=any)

power_pipe: Output from powerpipe (pp) evaluation
(type=bool)

Return Value

colored output based on input contents
(type=str)

cmd_split(*self*, *cmds*)

takes a command (as previously split up by pipes and input as array cmds), and returns individual terms (cmd_array) that will be evaluated individually. Also returns a string_format string that will be used to stitch together the output with the proper spacing based on the presence of "+" and ","

Parameters

cmds: individual commands separated by pipes
(type=list<str>)

Return Value

individual commands with corresponding string format
(type=list<str>)

all_meta_split(*self*, *input_str*)

splits a string on any metacharacter

Parameters

input_str: input string
(*type=**str*)

Return Value

list with no metacharacters
(*type=**list*<*str*>)

string_splitter(*self*)

splits self.p based on common metacharacters. returns a dictionary of this information.

Return Value

input split up by common metacharacters
(*type=**dict*<*str*:*list*<*str*>>)

join_and_format(*self*, *join_type*)

joins self.p arrays with a specified metacharacter

Parameters

join_type: metacharacter to join array
(*type=**str*)

Return Value

string joined by metacharacter
(*type=**str*)

array_joiner(*self*)

generates a dict of self.p arrays joined with various common metacharacters

Return Value

input joined by common metacharacters
(*type=**dict*<*str*:*str*>)

translate_preset_variables(*self, file_input, second_stream_input*)

translates variables to protected namespace dictionary for feeding into eval command.

Parameters

file_input: data from file
(*type=list*)

second_stream_input: input from second stream
(*type=list<str>*)

Return Value

values of preset variable for direct use by users
(*type=dict<str:str>*)

initialize_n(*self*)

initializes history dict for a particular n, where n is the line of the input being processed

safe_eval(*self, cmd, variables*)

evaluates a str command (cmd) safely. takes a dictionary of protected namespace variables as input. returns output of python call, which can be any type of python data type (typically str or array though).

Parameters

cmd: command to be evaluated using python
(*type=str*)

variables: preset variables used for evaluation
(*type=dictionary*)

Return Value

output from python evaluation
(*type=list<str>*)

update_history(*self*, *total_output*, *power_pipe*)

updates history dictionary with output from python evaluation

Parameters

total_output: output from python evaluation
(*type=list<str>*)

power_pipe: presence of powerpipe (pp) in eval
(*type=bool*)

flatten_list(*self*, *iterables*)

returns a list of strings from nested lists

Parameters

iterables: nested list to flatten
(*type=list<str>*)

power_pipe_eval(*self*, *cmd*, *inputs*, *power_pipe_type*)

evaluates pp statement. returns sanitized result.

Parameters

cmd: power pipe command
(*type=str*)

inputs: inputs from std-in or previous python eval
(*type=list<str>*)

power_pipe_type: kind of powerpipe (future use)
(*type=list*)

Return Value

'p' and output of python evaluation
(*type=list<str>*)

detect_power_pipe(*self, command, power_pipe_type*)

detects presense of powerpipe

Parameters

command: command to be evaluated
(*type=*str)
power_pipe_type: kind of powerpipe (future use)
(*type=*str)

Return Value

True if powerpipe else False
(*type=*bool)

format_input(*self, cmd, input_set*)

checks for powerpipe presence, evaluates powerpipe pp and returns formatted output if detected

Parameters

cmd: user command
(*type=*str)
input_set: input from std-in or previous python evaluation
(*type=*list<str>)

Return Value

command, input set, presence of powerpipe
(*type=*list<str>)

unlist_p(*self, p*)

changes one item arrays to strings for input cleanup

Parameters

p: input from std-in or previous pyp evaluation
(*type=*list)

Return Value

will return a string if input is a list and has one member
(*type=*list<str>,str)

process(*self*, *inputs*, *file_input*, *cmds*, *second_stream_input*)

takes primary data from input stream (can be string, array or dictionary), applies user commands to it, captures this output. Also, generates several variables such as line counter and various string split and join variables. Outputs string, array, or dictionary in a format delimited by the string formatting stamp

Parameters

inputs: inputs from std-in or previous pyp eval
(*type=*str,list)

file_input: inputs from file
(*type=*list)

cmds: python commands to be evaluated
(*type=*list<str>)

second_stream_input: second stream input
(*type=*list<str>)

output(*self*, *total_cmds*)

generates final output.

Parameters

total_cmds: all commands executed
(*type=*list<str>)

initilize_input(*self*)

decides what type of input to use (all arrays. can be from rerun file, yaml, or st-in. also does some basic processing, for using different delimiters or looking for jts numbers

Return Value

starting input for pyp processing
(*type=*list)

main(*self*)

generates input stream based on file, std-in options, rerun, starts process loop, generates output

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

1.11.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

1.11.3 Instance Variables

Name	Description
history	master record of all manipulations (<i>type=</i> dict<int:dict>)
n	current input line number (<i>type=</i> int)
p	current input line being manipulated (<i>type=</i> str or list)
pwd	current directory (<i>type=</i> str)

Index

- script-pyp (*script*), 2–19
 - script-pyp.Colors (*class*), 2–3
 - script-pyp.PowerPipeList (*class*), 3–7
 - script-pyp.PowerPipeList.after (*method*), 5
 - script-pyp.PowerPipeList.before (*method*), 5
 - script-pyp.PowerPipeList.delimit (*method*), 4
 - script-pyp.PowerPipeList.divide (*method*), 4
 - script-pyp.PowerPipeList.get_strings (*method*), 5
 - script-pyp.PowerPipeList.matrix (*method*), 6
 - script-pyp.PowerPipeList.oneliner (*method*), 4
 - script-pyp.PowerPipeList.uniquer (*method*), 4
 - script-pyp.PowerPipeList.unlister (*method*), 5
 - script-pyp.PowerPipeListCustom (*class*), 2
 - script-pyp.Pyp (*class*), 9–19
 - script-pyp.Pyp.all_meta_split (*method*), 13
 - script-pyp.Pyp.array_joiner (*method*), 14
 - script-pyp.Pyp.array_tracer (*method*), 13
 - script-pyp.Pyp.cmd_split (*method*), 13
 - script-pyp.Pyp.cmds_split (*method*), 10
 - script-pyp.Pyp.delete_macros (*method*), 11
 - script-pyp.Pyp.detect_power_pipe (*method*), 16
 - script-pyp.Pyp.flatten_list (*method*), 16
 - script-pyp.Pyp.format_input (*method*), 17
 - script-pyp.Pyp.get_custom_macro_paths (*method*), 10
 - script-pyp.Pyp.initialize_n (*method*), 15
 - script-pyp.Pyp.initilize_input (*method*), 18
 - script-pyp.Pyp.join_and_format (*method*), 14
 - script-pyp.Pyp.keep (*method*), 12
 - script-pyp.Pyp.list_macros (*method*), 11
 - script-pyp.Pyp.load_file (*method*), 11
 - script-pyp.Pyp.load_macros (*method*), 10
 - script-pyp.Pyp.lose (*method*), 12
 - script-pyp.Pyp.main (*method*), 18
 - script-pyp.Pyp.output (*method*), 18
 - script-pyp.Pyp.power_pipe_eval (*method*), 16
 - script-pyp.Pyp.process (*method*), 17
 - script-pyp.Pyp.safe_eval (*method*), 15
 - script-pyp.Pyp.shell (*method*), 11
 - script-pyp.Pyp.shelld (*method*), 12
 - script-pyp.Pyp.string_splitter (*method*), 14
 - script-pyp.Pyp.translate_preset_variables (*method*), 14
 - script-pyp.Pyp.unlist_p (*method*), 17
 - script-pyp.Pyp.update_history (*method*), 15
 - script-pyp.Pyp.write_macros (*method*), 10
 - script-pyp.PypCustom (*class*), 2
 - script-pyp.PypFunctionCustom (*class*), 2
 - script-pyp.PypList (*class*), 8–9
 - script-pyp.PypListCustom (*class*), 2
 - script-pyp.PypStr (*class*), 7–8
 - script-pyp.PypStr.kill (*method*), 7
 - script-pyp.PypStr.trim (*method*), 7
 - script-pyp.PypStrCustom (*class*), 2