pySFeel

Release 1.4.2

Jan 18, 2023

Contents:

CHAPTER 1

the sFeelParse function

class pySFeel.**SFeelParser**

sFeelParse(*text*)

Parse S-FEEL text

This routine parses the passed text, which must be valid S-FEEL

Parameters **param1** (*str*) – The S-FEEL text to be parsed

Returns

(status, value)

'status' is alist of any parsing errors.

'value' is the Python native value of the parsed S-FEEL text.

For an assignment statement the 'value' will be the Python native value assigned to the named variable.

For all other expressions the 'value' will be the Python native value of the S-FEEL expression.

Return type tuple

CHAPTER 2

Data Types

pySFeel converts S-FEEL data into the nearest equivalent Python native data type.

S-Feel data type	Python native data type
number	float
string	str
boolean	bool
days and time duration	datetime.timedelta
year and months duration	int
time	datetime.time
date	datetime.date
date and time	datetime.datetime
List	list
Context	dict
Range	tuple(end0, low0, high1, end1)
	where end0 is '[' or '(' and end1 is ')' or ']'

Literal strings (@"PT5H") are implemented as both literal strings (@"PT5H") and as bare strings (PT5H).
@"PT5H" > @"PT4H" can be written as PT5H > PT4H and would return True

NOTE: For safety, enclose 'codes' in double quotes.

The ICD-10 code P04D, if not enclosed in double qoutes, will be interpreted as a day and time duration of 4 days.

The AN-SNAP code if 499A will throw a syntax error if not enclosed in double quotes, as '499' will be interpreted as a number that should be followed by an operator.

CHAPTER 3

List and Context Filters

pySFeel supports List and Context filters with one deviation from the standard - the dot operator requires the List to be enclosed in brackets.

Hence, fred.y is **not** the 'y' filter on the List of Contexts named 'fred' (as fred.y is a valid name).
However (fred).y is the 'y' filter on the List of Contexts named fred.

CHAPTER 4

Assignment and Variable names

There's one extension - an assignment operator (<-) which will store a Python internal value against a named variable. Named variables are valid in S-FEEL expressions in pySFeel.

```
fred <- 7 bill <- 9 fred = bill
```

This will return False

```
fred <- [{x:1,y:2},{x:2,y:3}] (fred).y
```

This will return [2,3]

CHAPTER 5

Dot operators

All the dot operators are supported, however '.time offset', '.start included' and '.end included' can be ambiguous (because of the embedded space). Hence the variants '.time_offset', '.start_included', '.end_included' are supported.

    thisDateTime.time offset

will fail because thisDateTime.time looks like a valid FEEL variable name. However the following two alternatives will work

    thisDateTime.time_offset (thisDateTime).time offset

CHAPTER 6

Usage

```
import pySFeel
parser = pySFeel.SFeelParser()
sfeelText = '7.3 in [2.0 .. 9.1]'
(status, retVal) = parser.sFeelParse(sfeelText)
if 'errors' in status:
    print('With errors:', status['errors'])
```

- retVal will be True
- The dictonary ‘status’ will have the key ‘errors’ if you have errors in your sfeelText.
- status[‘errors’] is a list of strings. It may help in diagnosing your S-FEEL syntax errors.

CHAPTER 7

Built-in Functions

pySFeel has support all the standard FEEL built-in functions with some differences because pySFeel is a Python implementation.

Name(paramters)	Parameter Domain
date(from)	date string
date(from)	date and time
date(year,month,day)	year,month,day are numbers
date and time(date,time)	date is a date or date time; time is a time
date and time(from)	date time string
time(from)	time string
time(from)	time, date and time
number(from,grouping, separator,decimal separator)	string,string, string
string(from)	non null
duration(from)	duration string
years and months duration(from, to)	both are date or both are date and time
not(negand)	boolean
substring(string,start, position,length?)	string,number
string length(string)	string
upper case(string)	string
lower case(string)	string
substring before (string,match)	string,string
substring after (string,match)	string,string
replace(input,pattern, replacement,flags?)	string
contains(string,match)	string
starts with(string,match)	string
ends with(string,match)	string
matches(input,pattern, flags?)	string
split(string,delimiter)	string
list contains(list,element)	list,any element of the semantic domain
count(list)	list

Name(paramters)	Parameter Domain
min(list) min(C1,. . . ,Cn),N>0 max(list) max(C1,. . . ,Cn),N>0	non-empty list of comparable items or argument list of one or more c
sum(list) sum(N1,. . . ,Nn),N>0	list of 0 or more numbers or argument list of one or more numbers
mean(list) mean(N1,. . . ,Nn),N>0	non-empty list of numbers or argument list of one or more numbers
all(list) all(B1,. . . ,Bn),N>0	list of Boolean items of argument list of one or more Boolean items
any(list) any(B1,. . . ,Bn),N>0	list of Boolean items of argument list of one or more Boolean items
sublist(list,start position, length?)	list,number, number
append(list,item. . .)	list,any element
concatenate(list. . .)	list
insert before(list,position, newItem)	list,number,any element
remove(list,position)	list,number
reverse(list)	list
index of(list,match)	list,any element
union(list,. . .)	list
distinct values(list)	list
flattern(list)	list
product(list) product(N1,. . . ,Nn)	list is a list of numbers. N1..Nn are numbers
median(list) median(N1,. . . ,Nn)	list is a list of numbers. N1..Nn are numbers
stddev(list) stddev(N1,. . . ,Nn)	list is a list of numbers. N1..Nn are numbers
mode(list) mode(N1,. . . ,Nn)	list is a list of numbers. N1..Nn are numbers
deciman(n,scale)	number,number
floor(n)	number
ceiling(n)	number
abs(n)	number
modulo(dividend,divisor)	number,number
sqrt(n)	number
log(n)	number
exp(n)	number
odd(n)	number
even(n)	number
is(expr, expr)	expr, expr
before(range, range)	range, range
after(range, range)	range, range
meets(range, range)	range, range
met by(range, range)	range, range
overlaps(range, range)	range, range
overlaps before(range, range)	range, range
overlaps after(range, range)	range, range
finishes(range, range)	range, range
finished by(range, range)	range, range
includes(range, range)	range, range
during(range, range)	range, range
starts(range, range)	range, range
started by(range, range	range, range
coincides(range, range)	range, range
day of year(date)	date, date and time
day of week(date)	date, date and time
month of year(date)	date, date and time
week of year(date)	date, date and time
sort(list, function(x,y) expr)	list

Name(paramters)	Parameter Domain
now()	
today()	

Note: The support for the sort() function is very, very limited. Only the anonymous form is supported (function is defined within the sort call). Also, 'expr' is limited to 'name0 < name1' or 'name0 > name1' (ascending or decending). However, list can be a list of Contexts, in which case name0 and name1 must be 'name0.attr' and 'name1.attr' and 'attr' must be the same attribute for both 'name0' and 'name1'.

Note: The support for 'some/every in ... satifies expression' is also limited in that 'expression' must be 'name relop expr' or 'odd(name)' or 'even(name)'. Again, the 'name.attr' form is suported where a list of Contexts is being tested.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

P

S