



ArcGIS Python Field Calculations

- .cal files you can save or load and share
- Field calculator is GUI to
`arcpy.CalculateField_management()`
- Python or VB scripts
- Built-in numeric,datetime, string functions

Common String Functions

txtField1	txtField2
sample Text	Sample text
more sample Text	More sample text
Even more sample Text	Even more sample text

Show Codeblock
txtField2 =
!txtField1!.capitalize()

txtField1	txtField2
sample Text	sample text
more sample Text	more sample text
Even more sample Text	even more sample text

Show Codeblock
shortField =
!txtField1!.lower()

txtField1	txtField2
sample Text	SAMPLE TEXT
more sample Text	MORE SAMPLE TEXT
Even more sample Text	EVEN MORE SAMPLE TEXT

Show Codeblock
shortField =
!txtField1!.upper()

Common String Functions

txtField1	txtField2
sample Text	sample text
more sample Text	more sample text
Even more sample Text	Even more sample text

Show Codeblock

```
txtField2 =  
!txtField1!.replace("Text", "text")
```

txtField1	txtField2
sample Text	Text
more sample Text	Text
Even more sample Text	Text

Pre-Logic Script Code:

```
def getLastWord(strVar):  
#function uses .split to create space-delimited list  
lstStrings = strVar.split(" ")  
lastString = lstStrings[ (len(lstStrings)) - 1]  
return lastString
```

txtField2 =

```
getLastWord( !txtField1!)
```

Common Numeric Functions

shortFld1	shortFld2
-5	5
4	4
9	9
15	15

Show Codeblock

shortFld2 =

```
abs(!shortFld1)
```

dblFld1	shortFld2
0.120337	0
1.925391	1
9.747292	9

Show Codeblock

shortFld2 =

```
int(!dblFld1)
```

dblFld1	shortFld2
0.120337	0
1.925391	1
9.747292	9

Show Codeblock

shortFld2 =

```
math.floor(!dblFld1)
```

dblFld1	shortFld2
0.120337	1
1.925391	2
9.747292	10

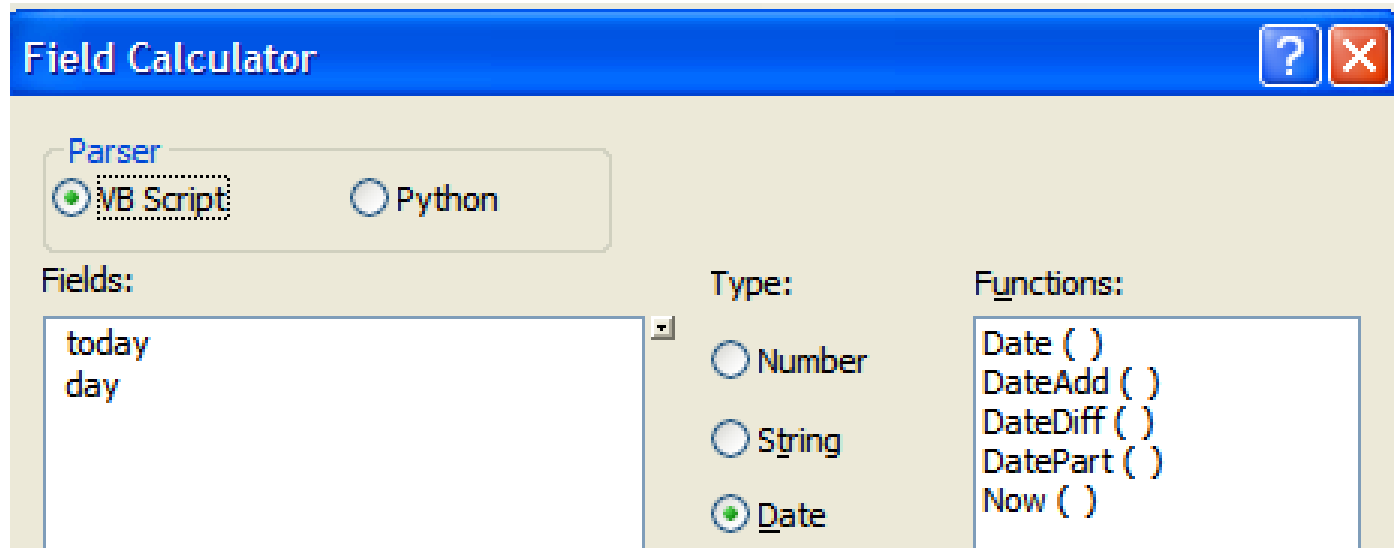
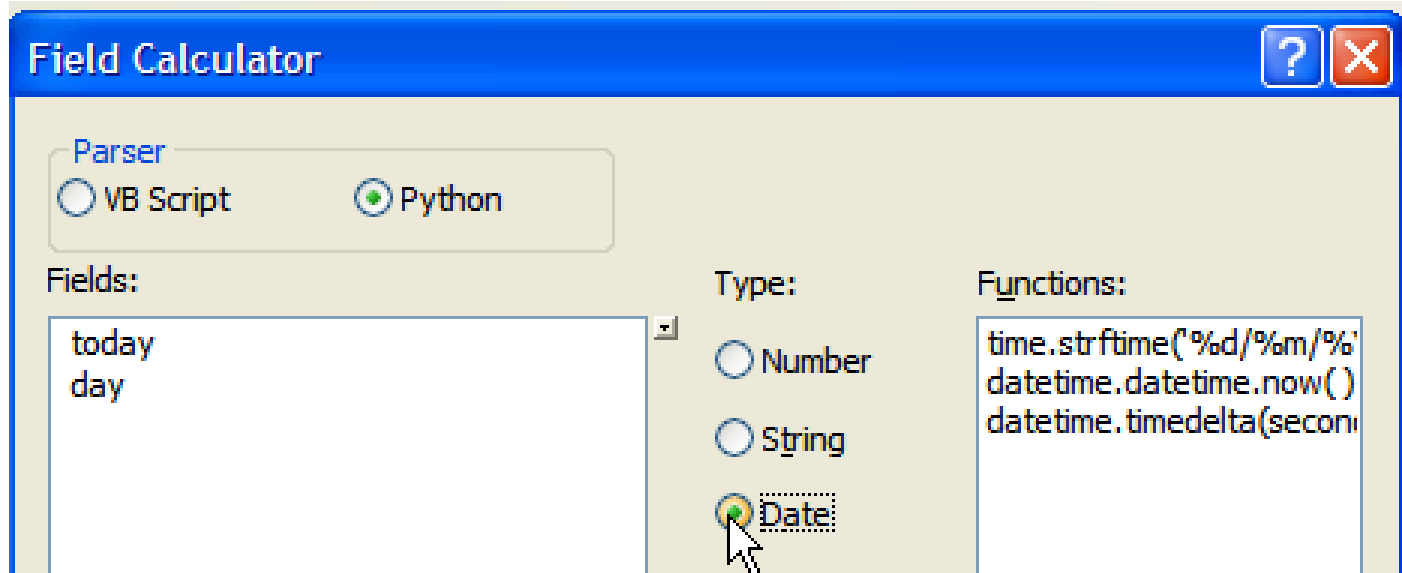
Show Codeblock

shortFld2 =

```
math.ceil(!dblFld1)
```

Functions:

math.acos()
 math.acosh()
 math.asin()
 math.asinh()
 math.atan()
 math.atan2()
 math.atanh()
 math.ceil()
 math.copysign()
 math.cos()
 math.cosh()
 math.degrees()



Common VB Date Functions

nowDate	shortField	shortField =
2/5/2011	36	DatePart ("y", [nowDate])
2/5/2011	36	
2/5/2011	36	

nowDate	shortField	shortField =
2/5/2011	2	DatePart ("m", [nowDate])
2/5/2011	2	
2/5/2011	2	

nowDate	shortField	shortField =
2/5/2011	2011	DatePart ("yyyy", [nowDate])
2/5/2011	2011	
2/5/2011	2011	
2/5/2011	2011	
2/5/2011	2011	
2/5/2011	2011	
2/5/2011	2011	
2/5/2011	2011	
2/5/2011	2011	
2/5/2011	2011	

nowDate	shortField	shortField =
2/5/2011	5	DatePart ("d", [nowDate])
2/5/2011	5	
2/5/2011	5	



nowDate
2/5/2011
2/5/2011
2/5/2011

```
nowDate =
DateAdd ("m",3, [nowDate] )
```

nowDate
5/5/2011
5/5/2011
5/5/2011

nowDate
5/5/2011
5/5/2011
5/5/2011

```
nowDate =
DateAdd ("d",-30, [nowDate] )
```

nowDate
4/5/2011
4/5/2011
4/5/2011

nowDate
4/5/2011
4/5/2011
4/5/2011

```
nowDate =
DateAdd ("yyyy",-10, [nowDate] )
```

nowDate
4/5/2001
4/5/2001
4/5/2001

```
shortField =
DateDiff ("d", [oldDate], [nowDate] )
```

oldDate	nowDate	shortField
1/30/2011	2/5/2011	6
2/1/2011	2/5/2011	4
2/2/2011	2/5/2011	3

Field Calculator [?] [X]

Parser: VB Script Python

Fields: OBJECTID, Shape, TYPE, Shape_Length, Root_Length

Type: Number String Date

Functions: Abs (), Atn (), Cos (), Exp (), Fix (), Int (), Log (), Sin (), Sqr (), Tan ()

Show Codeblock

Pre-Logic Script Code:

```
dim sqr_Root
sqr_Root = Sqr ([Shape_Length])
```

Root_Length =

sqr_Root

Buttons: Clear, Load..., Save..., Help, OK, Cancel

Field Calculator [?] [X]

Parser: VB Script Python

Fields: OBJECTID, Shape, TYPE, Shape_Length, Root_Length

Type: Number String Date

Functions: math.log10(), math.log1p(), math.modf(), math.pi(), math.pow(), math.radians(), math.sin(), math.sinh(), math.sqrt(), math.tan(), math.tanh(), math.trunc()

Show Codeblock

Pre-Logic Script Code:

```
sqr_Root = math.sqrt(!Shape_Length!)
```

Root_Length =

sqr_Root

Field Calculator [X]

There was a failure during processing, check the Geoprocessing Results window for details.

Buttons: OK

Table

points

	OBJECTID	BEARS_ID	testID
	30	31	40
	31	32	41
	32	33	42

Show Codeblock

Pre-Logic Script Code:

```
def myFunction(x):
    return x + 10
```

testID =

```
myFunction(!OBJECTID!)
```

Pre-Logic Script Code:

Pre-Logic Script Code:

```
def myFunction(x):
    return x + !BEARS_ID!
```



Failed to execute. Parameters are not valid.



ERROR [000989](#): Python syntax error: Parsing error <type 'exceptions.SyntaxError'>: inval



Failed to execute (CalculateField).

testID =

```
myFunction(!OBJECTID!)
```

<http://nrm.salrm.uaf.edu/~dverbyla/nrm638>



points

	OBJECTID	BEARS_ID	testID
	30	31	61
	31	32	63
	32	33	65

Pre-Logic Script Code:

```
def myFunction(x1,x2):  
    return x1 + x2
```

testID =

```
myFunction( !OBJECTID!, !BEARS_ID! )
```



dblField	dblFld2
0.120337	0.077951
1.925391	0.105874
9.747292	0.019057
30.806258	0.468266
75.21059	0.354507
155.956679	0.881536
288.929001	0.046239
492.90012	0.448008
789.530686	0.952365
1203.369434	0.525708

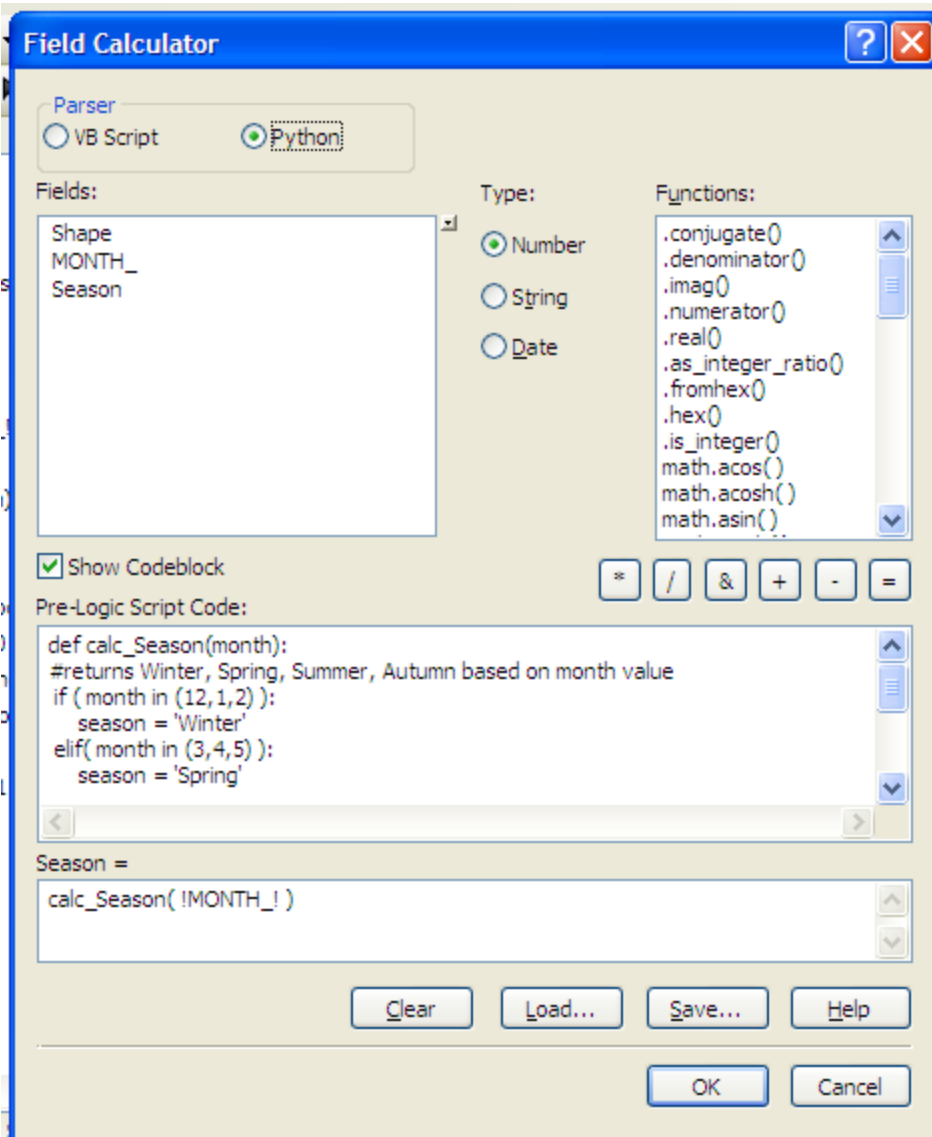
Pre-Logic Script Code:

```
def makeRandom(dblVar):  
    'returns random number from 0.0 to 1.0 '  
    import random  
    return random.random()
```



dblFld2 =

```
makeRandom(!dblFld2!)
```



caribou_locations			
	Shape *	MONTH_	Season
	Point	2	Winter
	Point	2	Winter
	Point	3	Spring
	Point	3	Spring
	Point	3	Spring

class	Name
6	Black Spruce
7	Scrub
8	Sedge Meadow
5	White Spruce
6	Black Spruce
7	Scrub
8	Sedge Meadow
7	Scrub
5	White Spruce
6	Black Spruce
6	Black Spruce
7	Scrub
7	Scrub
6	Black Spruce
6	Black Spruce
5	White Spruce
7	Scrub
8	Sedge Meadow
5	White Spruce
8	Sedge Meadow
6	Black Spruce
7	Scrub
7	Scrub

Field Calculator

Parser: VB Script Python

Fields: OID class Name

Type: Number String Date

Show Codeblock

Pre-Logic Script Code:

```
def vegName(vclass):
    'return vegclass string based on class integer value'
    if (vclass == 1):
        vegName = 'Balsam Polar'
    elif (vclass == 2):
        vegName = 'Quaking Aspen'
```

Name =

vegName(!class!)

Field Calculator

Parser: VB Script Python

Fields: OID class Name

Type: Number String Date

Show Codeblock

Pre-Logic Script Code:

```
def vegName(vclass):
    vegDict = {1:'Balsam Polar',2:'Quaking Aspen',3:'Alaskan Birch'\
,4:'Larch',5:'White Spruce',6:'Black Spruce'\
,7:'Scrub',8:'Sedge Meadow'}
    return vegDict.get(vclass,'Unclassified')
```

Name =

vegName(!class!)

Cumulative calculations

- Global variable, starting with value of zero

```
lastValue = 0
def cumulate(newValue):
    "function to accumulate newValue "
    global lastValue #will not lose when return from function
    if ( lastValue > 0):
        lastValue = lastValue + newValue
    else:
        lastValue = newValue
    return lastValue
```

shortField	shortFld2
0	0
1	1
9	10
30	40
75	115
155	270
288	558
492	1050
789	1839
1203	3042

Pre-Logic Script Code:

```
lastValue = 0
def cumulate(newValue):
    "function to accumulate newValue"
    global lastValue
    if (lastValue > 0):
        lastValue = lastValue + newValue
```

shortFld2 =

```
cumulate(!shortField!)
```