| | | |
|---|---|---|
| var | in | variable |

**Returns**
string with value of Glish variable

---

**substitute.substitutestring - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **substitute** |

return the value of a string variable

**Synopsis**
substitutestring(var)

**Description**
Return the value of a string variable. The string is enclosed in "". When the string contains a `"`, it is enclosed in single quotes. If the variable is a vector, the strings are enclosed in `[]` and separated by commas.

**Arguments**

| | | |
|---|---|---|
| var | in | variable |

**Returns**
string with value of Glish variable

---

---

### 1.6.17 itemcontainer - Tool

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |

A generic closure toolcontainer
include "itemcontainer.g"

**Constructors**

| | |
|---|---|
| itemcontainer | Construct an itemcontainer |

**Functions**

| | |
|---|---|
| add | Add the item to the container |
| delete | Remove the item from the container |
| done | Destroy this itemcontainer |
| fromrecord | Add the fields of this record |
| get | Recover value of specified item |
| has_item | Does this item exist |
| length | Returns the number of items |
| makeconst | Make this itemcontainer const |
| makeunconst | Make this itemcontainer unconst |
| names | Returns list of item names |
| set | Put the named item into the container |
| torecord | Return the contents of the container as a record |
| type | Return the type of this tool |

**Description**

This generic closure tool is designed to be used as a container of small items. Once created, the container can be made immutable. There is functionality to put items into the container, remove them and recover the contents of the container. `Items` can be of any type, and different types can be mixed in the one container.
Region of interest tools are actually `itemcomntainer` tools.

**Example**

We make a tool containing some smoothing kernel information. After invoking the `makeconst` function, you can no longer add items to the container or delete from it.

```
- include 'itemcontainer.g';
T
- include 'quanta.g';
T
- kernelType := 'gaussian';
- kernelWidth := dq.quantity(20,'arcsec');
- const i1 := itemcontainer();
- i1.set('type', kernelType);
T
```

```
- i1.set('width', kernelWidth);
T
- i1.makeconst();
T
- i1.set('comment', 'fish');
<fail>: This is a const object, you cannot write to it
        File:   itemcontainer.g, Line 228
        Stack:  .()
```

**itemcontainer - Constructor**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Construct an itemcontainer

**Synopsis**
itemcontainer()

**Returns**
itemcontainer tool

**Description**

This is the only `itemcontainer` constructor.

**Example**

```
- include 'itemcontainer.g'
- i1 := itemcontainer()
```

---

**itemcontainer.type - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Return the type of this tool

**Synopsis**
type()

**Returns**
String

**Description**

This tool function returns the string “itemcontainer”. If you have a tool, but don’t know what kind it is, this function usually exists to tell you who the tool is.

**Example**

```
- myic := itemcontainer();
- if (is_defined('myic') && is_record(myic) &&
      has_field(myic, 'type') && myic.type() == 'itemcontainer') {
     print 'yippee';
  } else {
     print 'itemcontainers are not impressive enough for my work';
  }
```

Note the use of the short cut “&&” operator. A statement is only evaluated if the statement to the left is true.

---

**itemcontainer.done - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Destroy this itemcontainer

**Synopsis**
done()

**Returns**

Bool

### Description

This tool function destroys the state of the current `itemcontainer` tool. The tool still exists, but it is no longer an `itemcontainer` !

### Example

```
- const myic := itemcontainer();
- myic.done();
T
- print 'myic=', myic;
myic=F
```

---

### itemcontainer.has_item - Function

| | |
|---:|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Does this item exist

**Synopsis**
has_item()

**Returns**
Bool

### Description

This tool function indicates whether the named item exists in the `itemcontainer` or not.

### Example

```
- const i1:= itemcontainer();
- i1.set('doggies', "retriever spaniel siamese");
T
- i1.has_item('doggies');
T
- i1.has_item('fish');
F
```

---

**itemcontainer.names - Function**

| | |
|---:|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Returns list of item names

**Synopsis**
names()

**Returns**
Vector of strings

**Description**

This tool function returns the list of item names. Note that any items added with the add will have a field name like “*n” where “n” is a number.

**Example**

```
- const fish := itemcontainer();
- fish.set('types', "salmon trout gold");
T
- fish.set('fish1', 'trout');
T
- fish.add('salmon');
3
- print 'names=', fish.names();
names=types fish1 *1
```

---

**itemcontainer.makeconst - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Make this itemcontainer const

**Synopsis**
makeconst()

**Returns**
Boolean

**Description**

After application of this tool function, you can no longer change the internal state of the `itemcontainer` tool.

**Example**

```
- i1 := itemcontainer();
T
- i1.set('par1', 20);
T
- i1.set('par2', 30);
T
- i1.makeconst();
T
- i1.set('par3', 40);
This is a const tool, you cannot overwrite it
<fail>: This is a const object, you cannot overwrite it
        File:   itemcontainer.g, Line 56
        Stack:  .()
```

Note that this is a different error from

```
- const i1 := itemcontainer()
- i1 := 'fish'
<fail>: <fail>: 'const' values cannot be modified.
```

Here Glish is complaining that it can't reassign the tool `i1`. The previous example is illustrating that after application of the `makeconst` function, you can't change the internal state of the `itemcontainer` tool (nothing to do with trying to overwrite it).

---

**itemcontainer.makeunconst - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Make this itemcontainer unconst

**Synopsis**
makeunconst()

**Returns**
Boolean

**Description**

After application of this tool function, you can change the internal state of the `itemcontainer` tool again.

**Example**

```
- i1 := itemcontainer();
T
- i1.set('par1', 20);
- i1.makeconst();
- i1.set('par3', 40);
This is a const tool, you cannot overwrite it
<fail>: This is a const object, you cannot overwrite it
        File:   itemcontainer.g, Line 56
        Stack:  .()
- i1.makeunconst();
- i1.set('par3', 40);
```

---

**itemcontainer.length - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Returns the number of items

**Synopsis**
length(showdeleted)

**Arguments**

| | |
|---|---|
| showdeleted | yes or no<br>Allowed:Bool<br>Default: F |

**Returns**
Integer

**Description**

This tool function returns the number of items in the list of item names. `itemcontainer`. If `showdeleted` is set to T, illegal fields are counted.

**Example**

```
- const fish := itemcontainer();
- fish.set('types', "salmon trout gold");
T
- fish.set('fish1', 'trout');
T
- fish.add('salmon');
3
- print 'number=', fish.length();
number=3
```

---

**itemcontainer.torecord - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Return the contents of the container as a record

**Synopsis**
torecord(showdeleted)

**Arguments**

| | | |
|---|---|---|
| showdeleted | yes or no<br>Allowed:Bool<br>Default: F | |

**Returns**
Record

**Description**

This tool function recovers the contents of the container into a record where each item has its own named (or numbered) field. If `showdeleted` is T, even "illegal" fields are copied.

**Example**

```
- const fish := itemcontainer();
- fish.set('types', "salmon trout gold");
T
- fish.set('fish1', 'trout');
T
- fish.add('salmon');
3
- x := fish.torecord();
- x
[types=salmon trout gold, fish1=trout, *1=salmon]
- is_record(x);
T
```

---

**itemcontainer.fromrecord - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Add the fields of this record

**Synopsis**
fromrecord()

**Returns**
Bool

**Description**

This tool function makes one item per field in the supplied record.

**Example**

```
- const fish := itemcontainer();
- rec := [=];
- rec.types := "salmon trout gold";
- rec.fish1 := 'trout';
- rec.fish2 := 'salmon';
- fish.fromrecord(rec);
T
- fish.names();
types fish1 fish2
- fish.torecord();
[types=salmon trout gold, fish1=trout, fish2=salmon]
```

---

**itemcontainer.set - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Put the named item into the container

**Synopsis**
set(item, value)

**Arguments**

| | |
|---|---|
| item | Item name<br>Allowed:String |
| value | Item<br>Allowed:Any |

**Returns**
Bool

**Description**

This tool function puts the specified item into the container with the given name.

**Example**

```
- include 'quanta.g';
T
- const i1 := itemcontainer();
- i1.set('type', 'boxcar');
T
- i1.set('width', dq.quantity(10,'arcsec'));
T
```

---

**itemcontainer.add - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Add the item to the container

**Synopsis**
add(value)

**Arguments**

| | |
|---|---|
| value | Item<br>Allowed:Any |

**Returns**
Integer

**Description**

This tool function adds the specified item into the container. The item is given a number one higher than the number of existing items in the container. This tool function returns the item number for the new item.

**Example**

```
- include 'quanta.g';
T
- const i1 := itemcontainer();
- idx := i1.add(dq.quantity(10,'arcsec'));
- x := i1.get(idx);
- x;
[value=10, unit=arcsec]
```

---

**itemcontainer.delete - Function**

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Remove the item from the container

**Synopsis**
delete(item)

**Arguments**

| | |
|---|---|
| item | Item<br>Allowed:String or Integer |

**Returns**

### Description

This tool function removes the specified `item` from the container. Item can be an item name or the number of an item. (It does not remove the field from the container to preserve indexing, but sets the item's value to illegal.)

### Example

```
- include 'quanta.g';
T
- const i1 := itemcontainer();
- idx := i1.add(dq.quantity(10,'arcsec'));
- x := i1.get(idx);
- x;
[value=10, unit=arcsec]
- i1.delete(idx);
T
- i1.get(idx);
<fail>: Requested item "1" is not present
        File:   ./itemcontainer.g, Line 152
        Stack:  .()
-
```

---

## itemcontainer.get - Function

| | |
|---|---|
| **Package** | **utility** |
| **Module** | **misc** |
| **Tool** | **itemcontainer** |

Recover value of specified item

### Synopsis

get(item)

### Arguments

| | |
|---|---|
| item | Item name or number<br>Allowed:String or integer |

### Returns

Glish variable

**Description**

This tool function recovers the specified variable into a Glish variable. You can specify the `item` either by its name or by its number.

**Example**

```
- include 'quanta.g';
T
- const i1 := itemcontainer();
- i1.set('type', 'boxcar');
T
- idx := i1.add(dq.quantity(10,'arcsec'));
- type := i1.get('type');
- type;
boxcar
- type := i1.get(1);
- type;
boxcar
- width := i1.get(2);
- width;
[value=10, unit=arcsec]
- x := i1.get(idx);
- x;
[value=10, unit=arcsec]
```

---

### 1.6.18 is_itemcontainer - Function

**Package** utility
**Module** misc

Is this an itemcontainer tool?

**Synopsis**
is_itemcontainer(thing)

**Arguments**

| | | |
|---|---|---|
| thing | The thing to test<br>Allowed: Any | |

**Returns**
Bool

**Description**

This global function determines whether this is an `itemcontainer` tool.

**Example**

```
- include 'regionmanager.g';
T
- r1 := drm.box();
- is_itemcontainer(r1);     # Regions are made from itemcontainers
T
- i1 := itemcontainer();
- is_itemcontainer(i1);
T
- x := [20];
- is_itemcontainer(x);
F
```

---

There are four required fields, Name, Email, Brief Description and Detail Description. If data is missing in any of these fields your bug report will not be sent.

---

### 1.6.19 ask - Function
**Package utility**
**Module misc**

Displays the AIPS++ question submission form

**Synopsis**
ask()