Matrix Programming

Indexing

Can slice arrays like lists or strings

```python
>>> block
array([[ 10,  20,  30,  40],
       [110, 120, 130, 140],
       [210, 220, 230, 240]])

>>> block[0:3, 0:2]
array([[ 10,  20],
       [110, 120],
       [210, 220]])
```
Matrices

Indexing

Can slice arrays like lists or strings

```python
>>> block
array([[ 10,  20,  30,  40],
        [110, 120, 130, 140],
        [210, 220, 230, 240]])
```

```python
>>> block[0:3, 0:2]
array([[ 10,  20],
        [110, 120],
        [210, 220]])
```

Are slices aliases or copies?

Can assign to slices

```python
>>> block[1, 1:3] = 0
```

```python
>>> block
array([[ 10,  20,  30,  40],
        [110,   0,   0, 140],
        [210, 220, 230, 240]])
```
Slice on both sides to shift data

```python
>>> vector = array([10, 20, 30, 40])
>>> vector[0:3] = vector[1:4]
>>> vector
array([20, 30, 40, 40])
```

Not overwritten

Is this easier to understand than a loop?
Can use lists or arrays as subscripts

```python
>>> vector
array([0, 10, 20, 30])
>>> subscript = [3, 1, 2]
>>> vector[subscript]
array([30, 10, 20])
```

Also works in multiple dimensions

Though operation may not be obvious

```python
>>> square = numpy.array([[5, 6], [7, 8]])
>>> square[[1]]
array([[7, 8]])
```
Also works in multiple dimensions
Though operation may not be obvious

```python
>>> square = numpy.array([[5, 6], [7, 8]])
>>> square[ [1] ]
array([[7, 8]])
```

Did we mention NumPy's excellent documentation?

Also works in multiple dimensions
Though operation may not be obvious

```python
>>> square = numpy.array([[5, 6], [7, 8]])
>>> square[ [1] ]
array([[7, 8]])
```

LOOPS
Comparisons

```python
>>> vector
array([0, 10, 20, 30])

>>> vector < 25
array([ True, True, False, False ],
dtype=bool)
```

Data type is Boolean

Use a Boolean subscript as a *mask*

```python
>>> vector
array([0, 10, 20, 30])

>>> vector[ vector < 25 ]
array([0, 10, 20])
```
Use a Boolean subscript as a *mask*

```python
>>> vector
array([0, 10, 20, 30])
>>> vector[ vector < 25 ]
array([0, 10, 20])
```

<table>
<thead>
<tr>
<th>vector</th>
<th>vector &lt; 25</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>T</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

**Copy or alias?**

Use masking for assignment

```python
>>> a = array([0, 1, 2, 3])
>>> mask = array([True, False, True, False])
>>> a[mask] = array([100, 101, 102, 103])
>>> a
array([100, 1, 101, 3])
```

<table>
<thead>
<tr>
<th>result</th>
<th>a</th>
<th>mask</th>
<th>fill</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>T</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>F</td>
<td>101</td>
</tr>
<tr>
<td>101</td>
<td>2</td>
<td>T</td>
<td>102</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>F</td>
<td>103</td>
</tr>
</tbody>
</table>

**Taken in order**
The `putmask` function works slightly differently

```python
>>> a = array([0, 1, 2, 3])
>>> putmask(a, mask,
       array([100, 101, 102, 103]))
>>> a
array([100, 1, 102, 3])
```

Python does not allow objects to re-define the meaning of `and`, `or`, and `not`

```python
>>> vector = array([0, 10, 20, 30])
>>> vector <= 20
array([True, True, True, False], dtype=bool)
>>> (vector <= 20) and (vector >= 20)
ValueError: The truth value of an array with more than one element is ambiguous.
```
Use `logical_and` / `logical_or` functions

```python
>>> logical_and(vector <= 20, vector >= 20)
array([False, False,  True, False], dtype=bool)
```

Or `|` for `or`, `&` for `and`

```python
>>> (vector <= 20) & (vector >= 20)
array([False, False,  True, False], dtype=bool)
```

The operators `|` and `&` deserve another look:

```python
>>> a = n.array([ 1,  2 ])
>>> b = n.array([ 1, -1 ])
>>> a | b
array([ 1, -1 ])
>>> a & b
array([ 1,  2 ])
```

`logical_and/logical_or` treat nonzero as `True`
Use `where` instead of `if/else`

```python
>>> vector = array([10, 20, 30, 40])
>>> where(vector < 25, vector, 0)
array([10, 20, 0, 0])
>>> where(vector > 25, vector/10, vector)
array([10, 20, 3, 4])
```

What do `choose` and `select` do?
Review:
- Arrays can be sliced
- Or subscripted with vectors of indices
- Or masked with conditionals

LOOPS