Using a Vector Class

Copyright (c) 2017, 2019 Tor Olav Kristensen, http://subcube.com (http://subcube.com)

https://github.com/t-o-k/scikit-vectors (https://github.com/t-o-k/scikit-vectors)

Use of this source code is governed by a BSD-license that can be found in the LICENSE file.

In [1]:
```python
from skvectors import create_class_Vector
```

In [2]:
```python
# Create a 3-dimensional vector class

VC = create_class_Vector('VC', 'abc')

# Explicit alternative:
# VC = \
#     create_class_Vector(
#         name = 'VC',
#         component_names = [ 'a', 'b', 'c' ],
#         brackets = [ '<', '>' ],
#         sep = ', ',
#         cnull = 0,
#         cunit = 1,
#         functions = None
#     )
```

In [3]:
```python
# Null value for vector components in the class
VC.component_null()
```

Out[3]: 0

In [4]:
```python
# Unit value for vector components in the class
VC.component_unit()
```

Out[4]: 1

```
In [5]:  1  # Basis vectors in class
         2  VC.basis_a(), VC.basis_b(), VC.basis_c()
```

Out[5]: (VC(a=1, b=0, c=0), VC(a=0, b=1, c=0), VC(a=0, b=0, c=1))

```
In [6]:  1  # Create a vector with all the components set to the cnull value
         2  VC.zero()
```

Out[6]: VC(a=0, b=0, c=0)

```
In [7]:  1  # Create a vector with all the components set to the cunit value
         2  VC.one()
```

Out[7]: VC(a=1, b=1, c=1)

```
In [8]:  1  # Null value for vector components
         2  v = VC(7, -8, 9)
         3  v.cnull
```

Out[8]: 0

```
In [9]:  1  # Unit value for vector components
         2  v = VC(7, -8, 9)
         3  v.cunit
```

Out[9]: 1

```
In [10]:  1  # Sum of component values in vector
          2  v = VC(-3, 4, 5)
          3  v.csum
```

Out[10]: 6

```
In [11]:  1  # Product of component values in vector
          2  v = VC(-3, 4, 5)
          3  v.cprod
```

Out[11]: -60
```

```
In [12]:  1  # Check if vector is zero vector
          2  v = VC.zero()
          3  v.is_zero_vector()

Out[12]:  True
```

```
In [13]:  1  # Check if vector is zero vector
          2  v = VC(0, 1e-14, 0)
          3  v.is_zero_vector()

Out[13]:  False
```

```
In [14]:  1  # Check if vector is not zero vector
          2  v = VC(0, 0, 0)
          3  bool(v)

Out[14]:  False
```

```
In [15]:  1  # Check if vector is not zero vector
          2  v = VC(0, 1e-14, 0)
          3  bool(v)

Out[15]:  True
```

```
In [16]:  1  # Check if vector contains a value
          2  u = VC(2, 3, 4)
          3  u.contains(3)

Out[16]:  True
```

```
In [17]:  1  # Check if a vector does not contain a value
          2  u = VC(2, 3, 4)
          3  u.contains_not(3.0)

Out[17]:  False
```

```
In [18]:  1  # Create a vector from the sum of vectors
          2  VC.sum_of_vectors([ ])

Out[18]:  VC(a=0, b=0, c=0)
```

```python
In [19]:  1  # Create a vector from the sum of vectors
          2  vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
          3  VC.sum_of_vectors(vectors)
```

Out[19]: VC(a=1, b=0, c=10)

```python
In [20]:  1  # Create a vector from the sum of vectors
          2  vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
          3  VC.sum_of_vectors(v for v in vectors)
```

Out[20]: VC(a=1, b=0, c=10)

```python
In [21]:  1  # Create a vector from the sum of vectors and scalars
          2  VC.sum_of_vectors([ VC(-1, 2, 3), 100, VC(-2, -2, 2), 8000 ])
```

Out[21]: VC(a=8097, b=8100, c=8105)

```python
In [22]:  1  # Create a vector from the product of vectors
          2  VC.prod_of_vectors([ ])
```

Out[22]: VC(a=1, b=1, c=1)

```python
In [23]:  1  # Create a vector from the product of vectors
          2  vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
          3  VC.prod_of_vectors(vectors)
```

Out[23]: VC(a=8, b=0, c=30)

```python
In [24]:  1  # Create a vector from the product of vectors
          2  vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
          3  VC.prod_of_vectors(v for v in vectors)
```

Out[24]: VC(a=8, b=0, c=30)

```python
In [25]:  1  # Create a vector from the product of vectors and scalars
          2  VC.prod_of_vectors([ VC(-1, 2, 3), -1/2, VC(-2, -2, 2), 10 ])
```

Out[25]: VC(a=-10.0, b=20.0, c=-30.0)
```

```
In [26]:   1  # Create vectors by applying the math methods floor, ceil and trunc to vector components
           2  from math import floor, ceil, trunc
           3  v = VC(-2.8, 3.3, 5.9)
           4  ceil(v), floor(v), trunc(v)
```

Out[26]: (VC(a=-2, b=4, c=6), VC(a=-3, b=3, c=5), VC(a=-2, b=3, c=5))

```
In [ ]:   1
```