Using a Vector Class

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

In [2]:
```
# 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]:
```
# Null value for vector components in the class
VC.component_null()
```

Out[3]: 0

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

Out[4]: 1

In [5]:
```
# Basis vectors in class
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]:
```
# Vector with all the components set to the cnull value
VC.zero()
```

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

In [7]:
```
# Vector with all the components set to the cunit value
VC.one()
```

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

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

Out[8]:
```
0
```

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

Out[9]:
```
1
```

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

Out[10]:
```
6
```

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

Out[11]:
```
-60
```

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

Out[12]: True

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

Out[13]: False

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

Out[14]: False

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

Out[15]: True

In [16]:
```
# Sum of vectors
VC.sum_of_vectors([ ])
```

Out[16]: VC(a=0, b=0, c=0)

In [17]:
```
# Sum of vectors
vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
VC.sum_of_vectors(vectors)
```

Out[17]: VC(a=1, b=0, c=10)

In [18]:
```
# Sum of vectors
vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
VC.sum_of_vectors(v for v in vectors)
```

Out[18]: VC(a=1, b=0, c=10)

In [19]:
```
# Sum of vectors and scalars
VC.sum_of_vectors([ VC(-1, 2, 3), 100, VC(-2, -2, 2), 8000 ])
```

Out[19]: VC(a=8097, b=8100, c=8105)

In [20]:
```
# Product of vectors
VC.prod_of_vectors([ ])
```

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

In [21]:
```
# Product of vectors
vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
VC.prod_of_vectors(vectors)
```

Out[21]: VC(a=8, b=0, c=30)

In [22]:
```
# Product of vectors
vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
VC.prod_of_vectors(v for v in vectors)
```

Out[22]: VC(a=8, b=0, c=30)

In [23]:
```
# Product of vectors and scalars
VC.prod_of_vectors([ VC(-1, 2, 3), -1/2, VC(-2, -2, 2), 10 ])
```

Out[23]: VC(a=-10.0, b=20.0, c=-30.0)

In [24]:
```
# Apply math methods floor, ceil and trunc to vector components
from math import floor, ceil, trunc
v = VC(-2.8, 3.3, 5.9)
ceil(v), floor(v), trunc(v)
```

Out[24]: (VC(a=-2, b=4, c=6), VC(a=-3, b=3, c=5), VC(a=-2, b=3, c=5))

In [ ]:
```

```