Using a Vector Class

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

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
# Number of dimensions for vectors in the class
VC.dimensions()
```

Out[3]:

```
3
```

In [4]:

```python
# List of component names for vectors in the class
VC.component_names()
```

Out[4]:

```
['a', 'b', 'c']
```

In [5]:

```
# Null value for vector components in the class
VC.component_null()
```

Out[5]:

```
0
```

In [6]:

```
# Unit value for vector components in the class
VC.component_unit()
```

Out[6]:

```
1
```

In [7]:

```
# Basis vectors in class
VC.basis_a(), VC.basis_b(), VC.basis_c()
```

Out[7]:

```
(VC(a=1, b=0, c=0), VC(a=0, b=1, c=0), VC(a=0, b=0, c=1))
```

In [8]:

```
# Vector with all the components set to the cnull value
VC.zero()
```

Out[8]:

```
VC(a=0, b=0, c=0)
```

In [9]:

```
# Vector with all the components set to the cunit value
VC.one()
```

Out[9]:

```
VC(a=1, b=1, c=1)
```

In [10]:

```
# Initialize vector
VC(1, -2, +3)
```

Out[10]:

```
VC(a=1, b=-2, c=3)
```

In [11]:

```
# Initialize vector
VC(a=1, b=-2, c=+3)
```

Out[11]:

```
VC(a=1, b=-2, c=3)
```

In [12]:

```
# NB: This does not work
# VC(1, -2, c=3)
```

In [13]:

```
# Initialize vector
l = [ 1, -2, 3 ]
VC(*l)
```

Out[13]:

```
VC(a=1, b=-2, c=3)
```

In [14]:

```
# Initialize vector
d = { 'a': 1, 'b': -2, 'c': 3 }
VC(**d)
```

Out[14]:

```
VC(a=1, b=-2, c=3)
```

In [15]:

```
# Initialize vector
VC.repeat_cvalue(8)
```

Out[15]:

```
VC(a=8, b=8, c=8)
```

In [16]:

```
# Number of dimensions of vector
v = VC.zero()
v.dimensions()
```

Out[16]:

```
3
```

In [17]:

```
# Number of dimensions of vector
v = VC.zero()
len(v)
```

Out[17]:

```
3
```

In [18]:

```
# Print vector
print(VC(2, 4, 6))
```

```
<2, 4, 6>
```

In [19]:

```
# Apply str() to vector
v = VC(2, 4, 6)
str(v)
```

Out[19]:

```
'<2, 4, 6>'
```

In [20]:

```python
# Apply str() to vector inside a string
v = VC(-3.3, 4.6, -5.5)
'str() of vector in string: {!s}'.format(v)
# 'str() of vector in string: {v0!s}'.format(v0=v)
# 'str() of vector in string: {v!s}'.format_map(vars())
```

Out[20]:

```
'str() of vector in string: <-3.3, 4.6, -5.5>'
```

In [21]:

```python
# Apply repr() to vector
v = VC(2, 4, 6)
repr(v)
```

Out[21]:

```
'VC(a=2, b=4, c=6)'
```

In [22]:

```python
# Apply repr() to vector
v = VC(2, 4, 6)
eval(repr(v))
```

Out[22]:

```
VC(a=2, b=4, c=6)
```

In [23]:

```python
# Apply repr() to vector inside a string
v = VC(-3.3, 4.6, -5.5)
'repr() of vector in string: {!r}'.format(v)
# 'repr() of vector in string: {v0!r}'.format(v0=v)
# 'repr() of vector in string: {v!r}'.format_map(vars())
```

Out[23]:

```
'repr() of vector in string: VC(a=-3.3, b=4.6, c=-5.5)'
```

In [24]:

```python
# Format vector
v = VC(2.222222, 4.444444, 6.6666666)
format(v, '.3e')
```

Out[24]:

```
'<2.222e+00, 4.444e+00, 6.667e+00>'
```

In [25]:

```python
# Format vectors inside string
u = VC(2.222222, 4.444444, 6.6666666)
w = VC(-3.3, 4.6, -5.5)
'Two vectors: {0:.4e} and {1:.2e}'.format(u, w)
# 'Two vectors: {v1:.4e} and {v2:.2e}'.format(v1=u, v2=w)
# 'Two vectors: {u:.4e} and {w:.2e}'.format_map(vars())
```

Out[25]:

```
'Two vectors: <2.2222e+00, 4.4444e+00, 6.6667e+00> and <-3.30e+00, 4.60e+00, -5.50e+00>'
```

In [26]:

```python
# Check if vector contains a value
v = VC(2, 3, 4)
3 in v
```

Out[26]:

```
True
```

In [27]:

```python
# Check if vector does not contain a value
v = VC(2, 3, 4)
3 not in v
```

Out[27]:

```
False
```

In [28]:

```python
# The component values
v = VC(-6, 8, 3)
v.a, v.b, v.c
```

Out[28]:

```
(-6, 8, 3)
```

In [29]:

```python
# Changing the component values
v = VC.zero()
v.a, v.b, v.c = 6, 7, 8
v
```

Out[29]:

```
VC(a=6, b=7, c=8)
```

In [30]:

```python
# # The component values / Indexing of vector
v = VC(-6, 8, 3)
v[0], v[1], v[2]
```

Out[30]:

```
(-6, 8, 3)
```

In [31]:

```python
# Indexing of vector
v = VC(-6, 8, 3)
v[0:3], v[:], v[::]
```

Out[31]:

```
([-6, 8, 3], [-6, 8, 3], [-6, 8, 3])
```

In [32]:

```python
v[:] = (cv for cv in [ -6, 8, 3 ])
v
```

Out[32]:

```
VC(a=-6, b=8, c=3)
```

In [33]:

```
# Change the component values
v = VC.zero()
v[0], v[1], v[2] = 6, 7, 8
v
```

Out[33]:

```
VC(a=6, b=7, c=8)
```

In [34]:

```
# Change the component values
v = VC.zero()
v[0:3] = 6, 7, 8
v
```

Out[34]:

```
VC(a=6, b=7, c=8)
```

In [35]:

```
# Change the component values
u = VC.zero()
w = VC(6, 7, 8)
u[:] = w
u
```

Out[35]:

```
VC(a=6, b=7, c=8)
```

In [36]:

```
# List of the component values
v = VC(2, 4, 6)
v.cvalues, v.component_values(), v[:]
```

Out[36]:

```
([2, 4, 6], [2, 4, 6], [2, 4, 6])
```

In [37]:

```
# List of the component values
v = VC(2, 4, 6)
list(v), [ *v ], [ getattr(v, cn) for cn in v.cnames ]
```

Out[37]:

```
([2, 4, 6], [2, 4, 6], [2, 4, 6])
```

In [38]:

```
# Iterate over the components
x, y, z = VC(2, 4, 6)
x, y, z
```

Out[38]:

```
(2, 4, 6)
```

In [39]:

```
# Iterate over the components
v = VC(2, 4, 6)
g = (cv for cv in v)
print(*g)
```

```
2 4 6
```

In [40]:

```
# Iterate over the components
v = VC(2, 4, 6)
components = iter(v)
next(components), next(components), next(components)
```

Out[40]:

```
(2, 4, 6)
```

In [41]:

```
# Check if vectors are equal
v = VC(2, 4, 6)
v == VC(2.0, 4.0, 6.0)
```

Out[41]:

```
True
```

In [42]:

```
# Check if vectors are not equal
v = VC(2, 4, 6)
v != VC(2.0, 4.0, 6.0)
```

Out[42]:

```
False
```

In [43]:

```
# Apply abs to the a-component
v = VC(-2, 3, -4)
v.c_abs_a()
```

Out[43]:

```
VC(a=2, b=3, c=-4)
```

In [44]:

```
# Apply unary minus to the c-component
v = VC(2, 3, 4)
v.c_neg_c()
```

Out[44]:

```
VC(a=2, b=3, c=-4)
```

In [45]:

```python
# Apply unary minus to all components except the c-component
v = VC(2, 3, 4)
v.c_neg_bar_c()
```

Out[45]:

```
VC(a=-2, b=-3, c=4)
```

In [46]:

```python
# Apply unary plus to the b-component and the c-component
v = VC(2, 3, 4)
v.c_pos_b_c()
```

Out[46]:

```
VC(a=2, b=3, c=4)
```

In [47]:

```python
# Add 100 to the c-component
v = VC(2, 3, 4)
v.c_add_c(100)
```

Out[47]:

```
VC(a=2, b=3, c=104)
```

In [48]:

```python
# Add 100 in-place to the c-component
v = VC(2, 3, 4)
v.c_iadd_c(100)
v
```

Out[48]:

```
VC(a=2, b=3, c=104)
```

In [49]:

```python
# Subtract 3 from the b-component
v = VC(2, 3, 4)
v.c_sub_b(3)
```

Out[49]:

```
VC(a=2, b=0, c=4)
```

In [50]:

```python
# Subtract 3 in-place from the b-component
v = VC(2, 3, 4)
v.c_isub_b(3)
v
```

Out[50]:

```
VC(a=2, b=0, c=4)
```

In [51]:

```
# Multiply all components except none by 8
v = VC(2, 3, 4)
v.c_mul_bar(8)
```

Out[51]:

```
VC(a=16, b=24, c=32)
```

In [52]:

```
# Multiply in-place all components except none by 8
v = VC(2, 3, 4)
v.c_imul_bar(8)
v
```

Out[52]:

```
VC(a=16, b=24, c=32)
```

In [53]:

```
# Raise the a-component to the power of 10
v = VC(2, 3, 4)
v.c_pow_a(10)
```

Out[53]:

```
VC(a=1024, b=3, c=4)
```

In [54]:

```
# Raise in-place the a-component to the power of 10
v = VC(2, 3, 4)
v.c_ipow_a(10)
v
```

Out[54]:

```
VC(a=1024, b=3, c=4)
```

In [55]:

```
# True divide none of the components by 0
v = VC(2, 3, 4)
v.c_truediv(0)
```

Out[55]:

```
VC(a=2, b=3, c=4)
```

In [56]:

```
# True divide in-place all of the components by 10
v = VC(2, 3, 4)
v.c_itruediv_bar(10)
v
```

Out[56]:

```
VC(a=0.2, b=0.3, c=0.4)
```

In [57]:

```
# Floor divide of all of the components by 2
v = VC(2, 3, 4)
v.c_floordiv_a_b_c(2)
```

Out[57]:

```
VC(a=1, b=1, c=2)
```

In [58]:

```
# Floor divide in-place all of the components by 2
v = VC(2, 3, 4)
v.c_ifloordiv_a_b_c(2)
v
```

Out[58]:

```
VC(a=1, b=1, c=2)
```

In [59]:

```
# Mod of all of the components by 2
v = VC(2, 3, 4)
v.c_mod_a_b_c(2)
```

Out[59]:

```
VC(a=0, b=1, c=0)
```

In [60]:

```
# Mod in-place of all of the components by 2
v = VC(2, 3, 4)
v.c_imod_a_b_c(2)
v
```

Out[60]:

```
VC(a=0, b=1, c=0)
```

In [61]:

```
# Multiply the c-component by 100
v = VC(2, 4, 6)
mul_c = v.c_mul_c
mul_c(100)
```

Out[61]:

```
VC(a=2, b=4, c=600)
```

In [62]:

```
# Multiply in-place the c-component by 100
v = VC(2, 4, 6)
imul_c = v.c_imul_c
imul_c(100)
v
```

Out[62]:

```
VC(a=2, b=4, c=600)
```

In [63]:

```
# Apply unary minus to the a-component and the c-component
v = VC(2, 4, 6)
neg_a_c = getattr(v, 'c_neg_a_c')
neg_a_c()
```

Out[63]:

```
VC(a=-2, b=4, c=-6)
```

In [64]:

```
# Apply several operations to the components
v = VC(2, 3, 4)
f = v.c_mul_c
f(10).c_add_bar(88).c_mul_a_b(88).c_sub_bar_b_c(100000).c_neg_c()
```

Out[64]:

```
VC(a=-92080, b=8008, c=-128)
```

In [65]:

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

Out[65]:

```
6
```

In [66]:

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

Out[66]:

```
-60
```

In [67]:

```
# Vector as dictionary
v = VC(2, 4, 6)
v.as_dict()
```

Out[67]:

```
{'a': 2, 'b': 4, 'c': 6}
```

In [68]:

```
# Make shallow copy of vector
u = VC(2, 4, 6)
w = VC(*u)
w
```

Out[68]:

```
VC(a=2, b=4, c=6)
```

In [69]:

```
# Make shallow copy of vector
u = VC(2, 4, 6)
w = u.copy()
w
```

Out[69]:

```
VC(a=2, b=4, c=6)
```

In [70]:

```
# Apply abs function to each component
v = VC(-3.3, 4.6, -5.5)
v(abs)
```

Out[70]:

```
VC(a=3.3, b=4.6, c=5.5)
```

In [71]:

```
# Apply int class to each component
v = VC(-3.3, 4.6, -5.5)
v(int)
```

Out[71]:

```
VC(a=-3, b=4, c=-5)
```

In [72]:

```
# Apply lambda function to each component
v = VC(-3.3, 4.6, -5.5)
v(lambda s: 10 + s * 1000)
```

Out[72]:

```
VC(a=-3290.0, b=4610.0, c=-5490.0)
```

In [73]:

```
# Round components to 3 decimals
v = VC(2.22222, 4.444444, 6.6666666)
round(v, ndigits=3)
```

Out[73]:

```
VC(a=2.222, b=4.444, c=6.667)
```

In [74]:

```
# Round components to integer value
v = VC(2.22222, 4.444444, 6.6666666)
round(v)
```

Out[74]:

```
VC(a=2.0, b=4.0, c=7.0)
```

In [75]:

```
# Round component values
v = VC(a=-55555555.5, b=-33333333.3, c=55555555.5)
round(v, -4)
```

Out[75]:

```
VC(a=-55560000.0, b=-33330000.0, c=55560000.0)
```

In [76]:

```python
# Check if something is a vector
v = VC(-3, 4, 5)
VC.is_vector(v)
```

Out[76]:

```
True
```

In [77]:

```python
# Check if something is a vector
d = { 'x': -3, 'y': 4, 'z': 5 }
VC.is_vector(d)
```

Out[77]:

```
False
```

In [78]:

```python
# Check if vector is zero vector
v = VC.zero()
v.is_zero_vector()
```

Out[78]:

```
True
```

In [79]:

```python
# Check if vector is zero vector
v = VC(0, 1e-14, 0)
v.is_zero_vector()
```

Out[79]:

```
False
```

In [80]:

```python
# Check if vector is not zero vector
bool(VC(0, 0, 0))
```

Out[80]:

```
False
```

In [81]:

```python
# Check if vector is not zero vector
bool(VC(0, 1e-14, 0))
```

Out[81]:

```
True
```

In [82]:

```python
# Apply unary minus to vector
v = VC(-3, 4, 5)
-v
```

Out[82]:

```
VC(a=3, b=-4, c=-5)
```

In [83]:

```
# Apply unary plus to vector
v = VC(-3, 4, 5)
+v
```

Out[83]:

```
VC(a=-3, b=4, c=5)
```

In [84]:

```
# Addition of vectors
v = VC(-3, 4, 5)
v + VC(1, 1, -1)
```

Out[84]:

```
VC(a=-2, b=5, c=4)
```

In [85]:

```
# In-place addition of vectors
v = VC(-3, 4, 5)
v += VC(1, 1, -1)
v
```

Out[85]:

```
VC(a=-2, b=5, c=4)
```

In [86]:

```
# Subtraction of vectors
v = VC(-3, 4, 5)
v - VC(1, 1, -1)
```

Out[86]:

```
VC(a=-4, b=3, c=6)
```

In [87]:

```
# In-place subtraction of vectors
v = VC(-3, 4, 5)
v -= VC(1, 1, -1)
v
```

Out[87]:

```
VC(a=-4, b=3, c=6)
```

In [88]:

```
# Multiplication of vectors
v = VC(-1, 2, 3)
v * VC(2, 0, -2)
```

Out[88]:

```
VC(a=-2, b=0, c=-6)
```

In [89]:
```python
# In-place multiplication of vectors
v = VC(-1, 2, 3)
v *= VC(2, 0, -2)
v
```

Out[89]:
```
VC(a=-2, b=0, c=-6)
```

In [90]:
```python
# Multiplication of vector and scalar
v = VC(-1, 2, 3)
2 * v, v * 2
```

Out[90]:
```
(VC(a=-2, b=4, c=6), VC(a=-2, b=4, c=6))
```

In [91]:
```python
# In-place multiplication of vector and scalar
v = VC(-1, 2, 3)
v *= 2
v
```

Out[91]:
```
VC(a=-2, b=4, c=6)
```

In [92]:
```python
# True division of vectors
v = VC(-3, 4, 6)
v / VC(2, -2, 2)
```

Out[92]:
```
VC(a=-1.5, b=-2.0, c=3.0)
```

In [93]:
```python
# In-place true division of vectors
v = VC(-3, 4, 6)
v /= VC(2, -2, 2)
v
```

Out[93]:
```
VC(a=-1.5, b=-2.0, c=3.0)
```

In [94]:
```python
# True division of vector and scalar ***
v = VC(-3, 4, 6)
v / 6
```

Out[94]:
```
VC(a=-0.5, b=0.6666666666666666, c=1.0)
```

In [95]:

```python
# In-place true division of vector and scalar
v = VC(-3, 4, 6)
v /= 2
v
```

Out[95]:

```
VC(a=-1.5, b=2.0, c=3.0)
```

In [96]:

```python
# Vector to the power of vector
v = VC(-3, 4, 6)
v**VC(2, -2, 2)
```

Out[96]:

```
VC(a=9, b=0.0625, c=36)
```

In [97]:

```python
# In-place vector to the power of vector
v = VC(-3, 4, 6)
v **= VC(2, -2, 2)
v
```

Out[97]:

```
VC(a=9, b=0.0625, c=36)
```

In [98]:

```python
# Vector to the power of scalar ***
v = VC(-3, 5, 6)
v**2
```

Out[98]:

```
VC(a=9, b=25, c=36)
```

In [99]:

```python
# In-place vector to the power of scalar
v = VC(-3, 5, 6)
v **= 2
v
```

Out[99]:

```
VC(a=9, b=25, c=36)
```

In [100]:

```python
# Floor division of vectors
v = VC(-3, 5, 6)
v // VC(2, -2, 2)
```

Out[100]:

```
VC(a=-2, b=-3, c=3)
```

In [101]:

```
# In-place floor division of vectors
v = VC(-3, 5, 6)
v //= VC(2, -2, 2)
v
```

Out[101]:

```
VC(a=-2, b=-3, c=3)
```

In [102]:

```
# Floor division of vector and scalar ***
v = VC(-3, 5, 6)
v // 2
```

Out[102]:

```
VC(a=-2, b=2, c=3)
```

In [103]:

```
# In-place floor division of vector and scalar
v = VC(-3, 5, 6)
v //= 2
v
```

Out[103]:

```
VC(a=-2, b=2, c=3)
```

In [104]:

```
# Vector modulus vector **
u = VC(-3, 5, 6)
w = VC(2, -2, 2)
u % w
```

Out[104]:

```
VC(a=1, b=-1, c=0)
```

In [105]:

```
# In-place vector modulus vector
v = VC(-3, 5, 6)
w = VC(2, -2, 2)
v %= w
v
```

Out[105]:

```
VC(a=1, b=-1, c=0)
```

In [106]:

```
# Modulus of vector and scalar ***
v = VC(-3, 5, 6)
v % 2
```

Out[106]:

```
VC(a=1, b=1, c=0)
```

```
In [107]:
# In-place modulus of vector and scalar
v = VC(-3, 5, 6)
v %= 2
v
```

```
Out[107]: VC(a=1, b=1, c=0)
```

```
In [108]:
# Sum of vectors
VC.sum_of_vectors([ ])
```

```
Out[108]: VC(a=0, b=0, c=0)
```

```
In [109]:
# Sum of vectors
vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
VC.sum_of_vectors(vectors)
```

```
Out[109]: VC(a=1, b=0, c=10)
```

```
In [110]:
# 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[110]: VC(a=1, b=0, c=10)
```

```
In [111]:
# Sum of vectors and scalars
VC.sum_of_vectors([ VC(-1, 2, 3), 100, VC(-2, -2, 2), 8000 ])
```

```
Out[111]: VC(a=8097, b=8100, c=8105)
```

```
In [112]:
# Product of vectors
VC.prod_of_vectors([ ])
```

```
Out[112]: VC(a=1, b=1, c=1)
```

```
In [113]:
# Product of vectors
vectors = [ VC(-1, 2, 3), VC(-2, -2, 2), VC(4, 0, 5) ]
VC.prod_of_vectors(vectors)
```

```
Out[113]: VC(a=8, b=0, c=30)
```

In [114]:

```
# 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[114]:

```
VC(a=8, b=0, c=30)
```

In [115]:

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

Out[115]:

```
VC(a=-10.0, b=20.0, c=-30.0)
```

In [ ]:

```

```