Using a Fundamental Vector Class

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

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

# The first argument is a string with the name of the class
# to be created.

# The number of elements in the iterable given as the second
# argument determines the number of dimensions for the class.

FVC = create_class_Fundamental_Vector('FVC', 'abc')

# Explicit alternative:
# FVC = \
#     create_class_Fundamental_Vector(
#         name = 'FVC',
#         component_names = [ 'a', 'b', 'c' ],
#         brackets = [ '<', '>' ],
#         sep = ', '
#     )
```

In [3]:
```
# Number of dimensions for vectors in the class
FVC.dimensions()
```

Out[3]:
```
3
```

In [4]:
```
# Brackets for vectors in the class
# (Used when printing a vector and when applying str to a vector)
FVC.brackets
```
Out[4]:
```
['<', '>']
```
In [5]:
```
# Separator between components for vectors in the class
# (Used when printing a vector and when applying str or repr to a vector)
FVC.sep
```
Out[5]:
```
', '
```
In [6]:
```
# List of component names for vectors in the class
FVC.component_names()
```
Out[6]:
```
['a', 'b', 'c']
```
In [7]:
```
# Initialize a vector
FVC(1, -2, +3)
```
Out[7]:
```
FVC(a=1, b=-2, c=3)
```
In [8]:
```
# Initialize a vector
FVC(a=1, b=-2, c=+3)
```
Out[8]:
```
FVC(a=1, b=-2, c=3)
```
In [9]:
```
# Initialize a vector
l = [ 1, -2, 3 ]
FVC(*l)
```
Out[9]:
```
FVC(a=1, b=-2, c=3)
```
In [10]:
```
# Initialize vector
d = { 'a': 1, 'b': -2, 'c': 3 }
FVC(**d)
```
Out[10]:
```
FVC(a=1, b=-2, c=3)
```

In [11]:
```
# Initialize a vector
FVC.fill(8)
```
Out[11]:
```
FVC(a=8, b=8, c=8)
```

In [12]:
```
# Number of dimensions of vector
u = FVC(0, 0, 0)
u.dimensions()
```
Out[12]:
```
3
```

In [13]:
```
# Number of dimensions of vector
u = FVC(0, 0, 0)
len(u)
```
Out[13]:
```
3
```

In [14]:
```
# List of component names for vector
u = FVC(0, 0, 0)
u.cnames
```
Out[14]:
```
['a', 'b', 'c']
```

In [15]:
```
# Check if something is a vector
u = FVC(-3, 4, 5)
FVC.is_vector(u)
```
Out[15]:
```
True
```

In [16]:
```
# Check if something is a vector
d = { 'a': -3, 'b': 4, 'c': 5 }
FVC.is_vector(d)
```
Out[16]:
```
False
```

In [17]:
```python
# Print a vector
u = FVC(2, 4, 6)
print(u)
```
```
<2, 4, 6>
```

In [18]:
```python
# Applying str to a vector
u = FVC(2, 4, 6)
str(u)
```
Out[18]:
```
'<2, 4, 6>'
```

In [19]:
```python
# Applying str to a vector inside a string
u = FVC(-3.3, 4.6, -5.5)
'str applied to a vector: {!s}'.format(u)
```
Out[19]:
```
'str applied to a vector: <-3.3, 4.6, -5.5>'
```

In [20]:
```python
# Applying repr to a vector
u = FVC(2, 4, 6)
repr(u)
```
Out[20]:
```
'FVC(a=2, b=4, c=6)'
```

In [21]:
```python
# NB: This does only work if the sep parameter in the class
# creation above contains a comma, or a comma and space(s)

# Applying repr to a vector
u = FVC(2, 4, 6)
eval(repr(u))
```
Out[21]:
```
FVC(a=2, b=4, c=6)
```

In [22]:
```python
# Applying repr to a vector inside a string
u = FVC(-3.3, 4.6, -5.5)
'repr applied to a vector: {!r}'.format(u)
```
Out[22]:
```
'repr applied to a vector: FVC(a=-3.3, b=4.6, c=-5.5)'
```

```
In [23]: # Applying format to a vector
         u = FVC(2.222222, 4.444444, 6.6666666)
         format(u, '.3e')

Out[23]: '<2.222e+00, 4.444e+00, 6.667e+00>'
```

```
In [24]: # Applying format to vectors inside a string
         u = FVC(2.222222, 4.444444, 6.6666666)
         v = FVC(-3.3, 4.6, -5.5)
         'format applied to two vectors: {:.4e} and {:.2e}'.format(u, v)

Out[24]: 'format applied to two vectors: <2.2222e+00, 4.4444e+00, 6.6667e+00> and <-3.30e+00, 4.60e+00, -5.50e+00>'
```

```
In [25]: # Check if vector contains a value
         u = FVC(2, 3, 4)
         3 in u

Out[25]: True
```

```
In [26]: # Check if a vector does not contain a value
         u = FVC(2, 3, 4)
         3.0 not in u

Out[26]: False
```

```
In [27]: # The component values of a vector
         u = FVC(-6, 8, 3)
         u.a, u.b, u.c

Out[27]: (-6, 8, 3)
```

```
In [28]: # Change the component values of a vector
         u = FVC(0, 0, 0)
         u.a, u.b, u.c = 6, 7, 8
         u

Out[28]: FVC(a=6, b=7, c=8)
```

```
In [29]: # Change a component value of a vector
         u = FVC(0, 0, 0)
         u.a += 100
         u

Out[29]: FVC(a=100, b=0, c=0)
```

```
In [30]: # Change a component value of a vector
         u = FVC(3, -4, 20)
         u.c //= 8
         u

Out[30]: FVC(a=3, b=-4, c=2)
```

```
In [31]: # The component values / Indexing of vector
         u = FVC(7, -8, 9)
         u[0], u[1], u[2]

Out[31]: (7, -8, 9)
```

```
In [32]: # The component values / Indexing of vector
         u = FVC(7, -8, 9)
         u[-3], u[-2], u[-1]

Out[32]: (7, -8, 9)
```

```
In [33]: # Indexing of a vector
         u = FVC(7, -8, 9)
         u[0:3], u[:], u[::]

Out[33]: ([7, -8, 9], [7, -8, 9], [7, -8, 9])
```

```
In [34]: # Change the component values of a vector
         u = FVC(0, 0, 0)
         u[0], u[1], u[2] = 7, -8, 9
         u

Out[34]: FVC(a=7, b=-8, c=9)
```

In [35]:
```
# Change the component values of a vector
u = FVC(0, 0, 0)
u[0:3] = 7, -8, 9
u
```
Out[35]:
```
FVC(a=7, b=-8, c=9)
```

In [36]:
```
# Change the component values of a vector
u = FVC(0, 0, 0)
v = FVC(7, -8, 9)
u[:] = v
u
```
Out[36]:
```
FVC(a=7, b=-8, c=9)
```

In [37]:
```
# Change the component values of a vector
u = FVC(0, 0, 0)
u[:] = (cv for cv in [ 7, -8, 9 ])
u
```
Out[37]:
```
FVC(a=7, b=-8, c=9)
```

In [38]:
```
# List of the component values of a vector
u = FVC(7, -8, 9)
u.cvalues, u.component_values(), u[:]
```
Out[38]:
```
([7, -8, 9], [7, -8, 9], [7, -8, 9])
```

In [39]:
```
# List of the component values
u = FVC(7, -8, 9)
list(u), [ *u ], [ getattr(u, cn) for cn in u.cnames ]
```
Out[39]:
```
([7, -8, 9], [7, -8, 9], [7, -8, 9])
```

In [40]:
```
# Iterate over the components
u = FVC(7, -8, 9)
x, y, z = u
x, y, z
```
Out[40]:
```
(7, -8, 9)
```

In [41]:
```python
# Iterate over the components
u = FVC(7, -8, 9)
g = (cv for cv in u)
print(*g)
```

```
7 -8 9
```

In [42]:
```python
# Iterate over the components
u = FVC(7, -8, 9)
components = iter(u)
next(components), next(components), next(components)
```

Out[42]:
```
(7, -8, 9)
```

In [43]:
```python
# Check if a vector is equal to another
u = FVC(2.0, 4.0, 6.0)
v = FVC(2, 4, 6)
u == v
```

Out[43]:
```
True
```

In [44]:
```python
# Check if a vector is not equal to another
u = FVC(2, 4, 6)
v = FVC(2.0, 4.0, 6.0)
u != v
```

Out[44]:
```
False
```

In [45]:
```python
# Create a dictionary from the components of a vector and their names
u = FVC(2, 4, 6)
u.as_dict()
```

Out[45]:
```
{'a': 2, 'b': 4, 'c': 6}
```

In [46]:
```python
# Make shallow copy of vector
u = FVC(2, 4, 6)
v = FVC(*u)
v
```

Out[46]:
```
FVC(a=2, b=4, c=6)
```

```
In [47]: # Make shallow copy of vector
         u = FVC(2, 4, 6)
         v = u.copy()
         v

Out[47]: FVC(a=2, b=4, c=6)

In [48]: # Create a vector by applying a lambda function to each of its components
         u = FVC(-3.3, 4.6, -5.5)
         u(lambda s: 10 + s * 1000)

Out[48]: FVC(a=-3290.0, b=4610.0, c=-5490.0)

In [49]: # Create a vector by applying abs to each of its components
         u = FVC(-3.3, 4.6, -5.5)
         u(abs)

Out[49]: FVC(a=3.3, b=4.6, c=5.5)

In [50]: # Create a vector by applying abs to each of its components
         u = FVC(-3, 4, -5)
         FVC(*map(abs, u))

Out[50]: FVC(a=3, b=4, c=5)

In [51]: # Create a vector by applying the int class to each of its components
         u = FVC(-3.3, 4.6, -5.5)
         u(int)

Out[51]: FVC(a=-3, b=4, c=-5)

In [52]: # Change the components of a vector by applying the int class to each component
         u = FVC(-3.3, 4.6, -5.5)
         u[:] = map(int, u)
         u

Out[52]: FVC(a=-3, b=4, c=-5)
```

In [53]:
```
# Create a vector method that takes 1 vector as argument


def square(s):

    return s**2


FVC.create_vector_method_arg1('square', square)
u = FVC(2, 3, -4)
u.vector_square()
```

Out[53]:
```
FVC(a=4, b=9, c=16)
```

In [54]:
```
# Create, from a built in function, a vector method that takes 1 vector as argument
FVC.create_vector_method_arg1('abs', lambda s: abs(s))
u = FVC(2, 3, -4)
u.vector_abs()
```

Out[54]:
```
FVC(a=2, b=3, c=4)
```

In [55]:
```
# Create a vector method that takes 2 vectors as arguments


def add(s, t):

    return s + t


FVC.create_vector_method_arg2('add', add)

u = FVC(2, 3, -4)
v = FVC(1, -2, 3)
s = 1000
u.vector_add(v), v.vector_add(s)
```

Out[55]:
```
(FVC(a=3, b=1, c=-1), FVC(a=1001, b=998, c=1003))
```

In [56]:
```python
# Create a vector method that takes 3 vectors as arguments


def select(r, s, t):

    if r < 0:
        result = s
    else:
        result = t

    return result


FVC.create_vector_method_arg3('select', select)

u = FVC(-2, 0, 3)
v = FVC(1, 3, 5)
w = FVC(2, 4, 6)
s = 0
t = 100
u.vector_select(v, w), u.vector_select(s, t)
```

Out[56]:
```
(FVC(a=1, b=4, c=6), FVC(a=0, b=100, c=100))
```