

Using a Fundamental Vector Class

Copyright (c) 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]: 1 from skvectors import create_class_Fundamental_Vector
```

```
In [2]: 1 # Create a 3-dimensional fundamental vector class
2
3 # The first argument is a string with the name of the class
4 # to be created.
5
6 # The number of elements in the iterable given as the second
7 # argument determines the number of dimensions for the class.
8
9 FVC = create_class_Fundamental_Vector('FVC', 'abc')
10
11 # Explicit alternative:
12 # FVC = \
13 #     create_class_Fundamental_Vector(
14 #         name = 'FVC',
15 #         component_names = [ 'a', 'b', 'c' ],
16 #         brackets = [ '<', '>' ],
17 #         sep = ', '
18 #     )
```

```
In [3]: 1 # Number of dimensions for vectors in the class
2       2 FVC.dimensions()
```

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

```
In [4]: 1 # Brackets for vectors in the class
        2 # (Used when printing a vector and when applying str to a vector)
        3 FVC.brackets
```

```
Out[4]: ['<', '>']
```

```
In [5]: 1 # Separator between components for vectors in the class
        2 # (Used when printing a vector and when applying str or repr to a vector)
        3 FVC.sep
```

```
Out[5]: ', '
```

```
In [6]: 1 # List of component names for vectors in the class
        2 FVC.component_names()
```

```
Out[6]: ['a', 'b', 'c']
```

```
In [7]: 1 # Initialize a vector
        2 FVC(1, -2, +3)
```

```
Out[7]: FVC(a=1, b=-2, c=3)
```

```
In [8]: 1 # Initialize a vector
        2 FVC(a=1, b=-2, c=+3)
```

```
Out[8]: FVC(a=1, b=-2, c=3)
```

```
In [9]: 1 # Initialize a vector
        2 l = [ 1, -2, 3 ]
        3 FVC(*l)
```

```
Out[9]: FVC(a=1, b=-2, c=3)
```

```
In [10]: 1 # Initialize vector
         2 d = { 'a': 1, 'b': -2, 'c': 3 }
         3 FVC(**d)
```

```
Out[10]: FVC(a=1, b=-2, c=3)
```

```
In [11]: 1 # Initialize a vector
        2 FVC.fill(8)
```

Out[11]: FVC(a=8, b=8, c=8)

```
In [12]: 1 # Number of dimensions of vector
        2 u = FVC(0, 0, 0)
        3 u.dimensions()
```

Out[12]: 3

```
In [13]: 1 # Number of dimensions of vector
        2 u = FVC(0, 0, 0)
        3 len(u)
```

Out[13]: 3

```
In [14]: 1 # List of component names for vector
        2 u = FVC(0, 0, 0)
        3 u.cnames
```

Out[14]: ['a', 'b', 'c']

```
In [15]: 1 # Check if something is a vector
        2 u = FVC(-3, 4, 5)
        3 FVC.is_vector(u)
```

Out[15]: True

```
In [16]: 1 # Check if something is a vector
        2 d = { 'a': -3, 'b': 4, 'c': 5 }
        3 FVC.is_vector(d)
```

Out[16]: False

```
In [17]: 1 # Print a vector
        2 u = FVC(2, 4, 6)
        3 print(u)
```

<2, 4, 6>

```
In [18]: 1 # Applying str to a vector
        2 u = FVC(2, 4, 6)
        3 str(u)
```

Out[18]: '<2, 4, 6>'

```
In [19]: 1 # Applying str to a vector inside a string
        2 u = FVC(-3.3, 4.6, -5.5)
        3 'str applied to a vector: {!s}'.format(u)
```

Out[19]: 'str applied to a vector: <-3.3, 4.6, -5.5>'

```
In [20]: 1 # Applying repr to a vector
        2 u = FVC(2, 4, 6)
        3 repr(u)
```

Out[20]: 'FVC(a=2, b=4, c=6)'

```
In [21]: 1 # NB: This does only work if the sep parameter in the class
        2 # creation above contains a comma, or a comma and space(s)
        3
        4 # Applying repr to a vector
        5 u = FVC(2, 4, 6)
        6 eval(repr(u))
```

Out[21]: FVC(a=2, b=4, c=6)

```
In [22]: 1 # Applying repr to a vector inside a string
        2 u = FVC(-3.3, 4.6, -5.5)
        3 '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]: 1 # Applying format to a vector
        2 u = FVC(2.222222, 4.444444, 6.666666)
        3 format(u, '.3e')
```

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

```
In [24]: 1 # Applying format to vectors inside a string
        2 u = FVC(2.222222, 4.444444, 6.6666666)
        3 v = FVC(-3.3, 4.6, -5.5)
        4 '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]: 1 # Check if vector contains a value
        2 u = FVC(2, 3, 4)
        3 3 in u
```

Out[25]: True

```
In [26]: 1 # Check if a vector does not contain a value
        2 u = FVC(2, 3, 4)
        3 3.0 not in u
```

Out[26]: False

```
In [27]: 1 # The component values of a vector
        2 u = FVC(-6, 8, 3)
        3 u.a, u.b, u.c
```

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

```
In [28]: 1 # Change the component values of a vector
        2 u = FVC(0, 0, 0)
        3 u.a, u.b, u.c = 6, 7, 8
        4 u
```

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

```
In [29]: 1 # Change a component value of a vector
        2 u = FVC(0, 0, 0)
        3 u.a += 100
        4 u
```

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

```
In [30]: 1 # Change a component value of a vector
         2 u = FVC(3, -4, 20)
         3 u.c //= 8
         4 u
```

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

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

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

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

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

```
In [33]: 1 # Indexing of a vector
         2 u = FVC(7, -8, 9)
         3 u[0:3], u[:], u[::]
```

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

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

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

```
In [35]: 1 # Change the component values of a vector
         2 u = FVC(0, 0, 0)
         3 u[0:3] = 7, -8, 9
         4 u
```

Out[35]: FVC(a=7, b=-8, c=9)

```
In [36]: 1 # Change the component values of a vector
        2 u = FVC(0, 0, 0)
        3 v = FVC(7, -8, 9)
        4 u[:] = v
        5 u
```

Out[36]: FVC(a=7, b=-8, c=9)

```
In [37]: 1 # Change the component values of a vector
        2 u = FVC(0, 0, 0)
        3 u[:] = (cv for cv in [ 7, -8, 9 ])
        4 u
```

Out[37]: FVC(a=7, b=-8, c=9)

```
In [38]: 1 # List of the component values of a vector
        2 u = FVC(7, -8, 9)
        3 u.cvalues, u.component_values(), u[:]
```

Out[38]: ([7, -8, 9], [7, -8, 9], [7, -8, 9])

```
In [39]: 1 # List of the component values
        2 u = FVC(7, -8, 9)
        3 list(u), [ *u ], [ getattr(u, cn) for cn in u.cnames ]
```

Out[39]: ([7, -8, 9], [7, -8, 9], [7, -8, 9])

```
In [40]: 1 # Iterate over the components
        2 u = FVC(7, -8, 9)
        3 x, y, z = u
        4 x, y, z
```

Out[40]: (7, -8, 9)

```
In [41]: 1 # Iterate over the components
        2 u = FVC(7, -8, 9)
        3 g = (cv for cv in u)
        4 print(*g)
```

7 -8 9

```
In [42]: 1 # Iterate over the components
         2 u = FVC(7, -8, 9)
         3 components = iter(u)
         4 next(components), next(components), next(components)
```

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

```
In [43]: 1 # Check if a vector is equal to another
         2 u = FVC(2.0, 4.0, 6.0)
         3 v = FVC(2, 4, 6)
         4 u == v
```

Out[43]: True

```
In [44]: 1 # Check if a vector is not equal to another
         2 u = FVC(2, 4, 6)
         3 v = FVC(2.0, 4.0, 6.0)
         4 u != v
```

Out[44]: False

```
In [45]: 1 # Create a dictionary from the components of a vector and their names
         2 u = FVC(2, 4, 6)
         3 u.as_dict()
```

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

```
In [46]: 1 # Make shallow copy of vector
         2 u = FVC(2, 4, 6)
         3 v = FVC(*u)
         4 v
```

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

```
In [47]: 1 # Make shallow copy of vector
         2 u = FVC(2, 4, 6)
         3 v = u.copy()
         4 v
```

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


```
In [48]: 1 # Create a vector by applying a lambda function to each of its components
        2 u = FVC(-3.3, 4.6, -5.5)
        3 u(lambda s: 10 + s * 1000)
```

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

```
In [49]: 1 # Create a vector by applying abs to each of its components
        2 u = FVC(-3.3, 4.6, -5.5)
        3 u(abs)
```

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

```
In [50]: 1 # Create a vector by applying abs to each of its components
        2 u = FVC(-3, 4, -5)
        3 FVC(*map(abs, u))
```

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

```
In [51]: 1 # Create a vector by applying the int class to each of its components
        2 u = FVC(-3.3, 4.6, -5.5)
        3 u(int)
```

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

```
In [52]: 1 # Change the components of a vector by applying the int class to each component
        2 u = FVC(-3.3, 4.6, -5.5)
        3 u[:] = map(int, u)
        4 u
```

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

```
In [53]: 1 # Create a vector method that takes 1 vector as argument
2
3
4 def square(s):
5
6     return s**2
7
8
9 FVC.create_vector_method_arg1('square', square)
10 u = FVC(2, 3, -4)
11 u.vector_square()
```

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

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

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

```
In [55]: 1 # Create a vector method that takes 2 vectors as arguments
2
3
4 def add(s, t):
5
6     return s + t
7
8
9 FVC.create_vector_method_arg2('add', add)
10
11 u = FVC(2, 3, -4)
12 v = FVC(1, -2, 3)
13 s = 1000
14 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]:

```
1  # Create a vector method that takes 3 vectors as arguments
2
3
4  def select(r, s, t):
5
6      if r < 0:
7          result = s
8      else:
9          result = t
10
11     return result
12
13
14  FVC.create_vector_method_arg3('select', select)
15
16  u = FVC(-2, 0, 3)
17  v = FVC(1, 3, 5)
18  w = FVC(2, 4, 6)
19  s = 0
20  t = 100
21  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))

In []:

1