The ``7-base\_geometry`` module

======================

Using ``BaseGeometry``

-------------------

Importing the function from the module:

>>> BG = \_\_import\_\_("7-base\_geometry").BaseGeometry

Checking for module docstring:

>>> m = \_\_import\_\_("7-base\_geometry").\_\_doc\_\_

>>> len(m) > 1

True

Checking for class docstring:

>>> c = \_\_import\_\_("7-base\_geometry").BaseGeometry.\_\_doc\_\_

>>> len(c) > 1

True

Checking for method docstring:

>>> mod1 = \_\_import\_\_("7-base\_geometry").BaseGeometry.area.\_\_doc\_\_

>>> len(mod1) > 1

True

>>> mod2 = \_\_import\_\_("7-base\_geometry").BaseGeometry.integer\_validator.\_\_doc\_\_

>>> len(mod2) > 1

True

Checking for area failure:

>>> b = BG()

>>> b.area()

Traceback (most recent call last):

...

Exception: area() is not implemented

Checking too many arguments for area:

>>> b.area(1)

Traceback (most recent call last):

...

TypeError: area() takes 1 positional argument but 2 were given

Checking integer validator for passing integer:

>>> b.integer\_validator("integer", 1)

Checking for integer == 0:

>>> b.integer\_validator("integer", 0)

Traceback (most recent call last):

...

ValueError: integer must be greater than 0

Checking for integer < 0;

>>> b.integer\_validator("integer", -5)

Traceback (most recent call last):

...

ValueError: integer must be greater than 0

Checking for non-integer types:

>>> b.integer\_validator("bool", True)

Traceback (most recent call last):

...

TypeError: bool must be an integer

>>> b.integer\_validator("float", 1.5)

Traceback (most recent call last):

...

TypeError: float must be an integer

>>> b.integer\_validator("complex", complex(1, 1))

Traceback (most recent call last):

...

TypeError: complex must be an integer

>>> b.integer\_validator("string", "hello")

Traceback (most recent call last):

...

TypeError: string must be an integer

>>> b.integer\_validator("tuple", (1, 2))

Traceback (most recent call last):

...

TypeError: tuple must be an integer

>>> b.integer\_validator("list", [1, 2, 3])

Traceback (most recent call last):

...

TypeError: list must be an integer

>>> b.integer\_validator("dict", {"key": "value"})

Traceback (most recent call last):

...

TypeError: dict must be an integer

>>> b.integer\_validator("set", {"hello", "world"})

Traceback (most recent call last):

...

TypeError: set must be an integer

>>> b.integer\_validator("frozenset", frozenset(["hello", "world"]))

Traceback (most recent call last):

...

TypeError: frozenset must be an integer

>>> b.integer\_validator("bytes", b"bytes")

Traceback (most recent call last):

...

TypeError: bytes must be an integer

>>> b.integer\_validator("bytearrays", bytearray(b"bytes"))

Traceback (most recent call last):

...

TypeError: bytearrays must be an integer

Checking for no arguments to integer\_validator:

>>> b.integer\_validator()

Traceback (most recent call last):

...

TypeError: integer\_validator() missing 2 required positional arguments: 'name' and 'value'

Checking for only 1 argument to integer\_validator:

>>> b.integer\_validator("integer")

Traceback (most recent call last):

...

TypeError: integer\_validator() missing 1 required positional argument: 'value'

Checking for too many arguments:

>>> b.integer\_validator("integer", 1, 2)

Traceback (most recent call last):

...

TypeError: integer\_validator() takes 3 positional arguments but 4 were given