**Power Function**

**Implementation notes**

Following an object-oriented structure, power\_ module was developed to implement required functionalities for a traditional exponential function in the calculator. The class contains 2 attributes (base, exponent), mutators, constructor, toString method as well as a helper function calculating the result of the base powered to the exponent. The constructor will initialize:

* Base (attribute) which represents the “x” for the transcendental function xy
* Exponent (attribute) which represents the “y” for the transcendental function xy

The constructor will initialize the attributes “base” and “exponent”. The mutators can be used to modify both attributes’ values. Lastly, the toString function can be used to print out the attributes’ values and the helper can be used to return the result of the exponential operation.

**Pseudocodes**

**Algorithm** setBase(base)

**Input** base is an integer or a string

**if** “/” **in** base do:

try

numerator 🡨 base[0 : base.index(‘/’)]

denumerator 🡨 base[base.index(‘/’) + 1]

base 🡨 float(numerator) / float(denumerator)

except ZeroDivison

print(“Can’t divide by 0”)

**else**

base 🡨 float(base)

**Algorithm** setExponent(exponent)

**Input** exponent is an integer or a string

**if** “/” **in** exponent do:

try

numerator 🡨 exponent[0 : exponent.index(‘/’)]

denumerator 🡨 exponent[exponent.index(‘/’) + 1]

exponent 🡨 float(numerator) / float(denumerator)

except ZeroDivison

print(“Can’t divide by 0”)

**else**

exponent 🡨 float(exponent)

total 🡨 total **+**