Matematica

Release 0.9.3

Caio vieira

Oct 24, 2021

CONTENTS

CHAPTER

ONE

INTRODUCTION

Matematica is a more pythonish simple and powerful python library.

```
import Matematica as mat

x = mat.add([n for n in range(10)])
print(x)
```

output::

```
45
```

1.1 Basic operators

I've re-created all 4 basic operators so you can easily manage them.

```
import Matematica as mat

x = mat.divide([mat.multiply([4, 6, 9]), mat.subtract([2, 3, 4])])
print(x)
```

output:

```
-43.2
```

It is really more convenient this way because you don't get confused within your code.

you can read more about them in Operators

1.2 Quadratic operators

You can do quadratic equations within square roots with recursive exponentiation, and the best part is: IT'S HUMAN READABLE!

```
from Matematica import nRoot as r, qdeq as q, xpnt as x

n = x([q(1, 3, 2)[1], x([r(8, 3), 10])])
print(n)
```

output:

```
4294967296.0
```

you can see that it's easy to mix up things. There are some limitations though. See more in Exponentiation

1.3 Utilities

There are some situations that you can get stuck on like when working with floats.

```
from Matematica import fract, divide

y = divide([78, 7, 9, 5])
x = fract(float(format(y, '.1f')))
print(f"Before: {y}\nAfter: {x}")
```

output:

```
Before: 0.24761904761904763
After: 1/5
```

there are some limitations though(for now). See more in Utilities

1.4 Others

there are some useful but not categorized functions that you can find in Others

CHAPTER

TWO

OPERATORS

Here you will see how easy and handy it is to work with the basic operators.

`add(arg=[0])` Add a n number of numbers

`subtract(arg=[0])` Subtract a n number of numbers

`multiply(arg=[0])` Multiply a n number of numbers

`divide(arg=[0])` Divide a n number of numbers

as you can see, they are really self explanatory.

2.1 Examples

You can do all kind of things that involves lists, like list comprehensions:

```
import Matematica as mat

x = mat.add([n for n in range(10)])
print(x)
```

output:

```
45
```

CHAPTER

THREE

EXPONENTIATION

Here are the exponentiation/quadratic related functions. they are unstable at the moment, but works well in expected situations.

xpnt(arg=[0]) exponentiation operation. it can do it recursively, like:

```
from Matematica import xpnt

x = xpnt([2, 2, 2])
print(x)
```

output:

```
16
```

here 2 is raised to the power of 2 and then the result is raised to the power of 2. If only one value is given, it will raise it to the power of 2, as in:

```
from Matematica import xpnt

x = xpnt([3])
print(x)
```

output:

```
9
```

Note that you can work with lists just like the *basic operators*.

nRoot(arg0=1, arg1=2) gets the 'n' root of a number, as in 'a square root'(witch is default when only the first argument is given).

sample:

```
from Matematica import nRoot

x = nRoot(8, 3)
print(x)
```

output:

```
2.0
```

Note the floating point. nRoot() has a floating point precision of 1, see an example:

```
from Matematica import nRoot

x = nRoot(10)
print(x)
```

output:

```
3.1
```

qdeq(a, b, c) solves a simple quadratic equation and returns a tuple with the results. the first item is the '+' version of the formula, and the second is the '-' version. if the discriminant is negative, it returns `False`

qdeqDisc(a, b, c) calculates the discriminant for the quadratic formula.

basic operators: #

CHAPTER

FOUR

UTILITIES

Here are some tools that make things nicer.

`fract(arg)` turns a decimal into a fraction. Example:

```
from Matematica import fract, divide

y = divide([78, 7, 9, 5])
x = fract(float(format(y, '.1f')))
print(f"Before: {y}\nAfter: {x}")
```

output:

```
Before: 0.24761904761904763
After: 1/5
```

Note that it only works(for now) with 1 floating point precision.

CHAPTER

FIVE

OTHERS

Here are some undefined type of functions. with time there will be a place for every thing.

`fact(arg)` calculates the factorial of a given number. If negative, it returns False.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search