In [125…
```
from qmul import *
```

In [126…
```
x = sympy.Symbol('x')
```

In [127…
```
eq = (7*x**1.5)
eq
```

Out[127… $7x^{1.5}$

In [128…
```
coeffExp = eq.as_coeff_exponent(x)
coeffExpFrac = [Fraction(str(eqNum)).as_integer_ratio() for eqNum in coeffExp]
coeffExpFrac
```

Out[128… [(7, 1), (3, 2)]

Add 1 due to integration

In [129…
```
# Add by a value equal to the denominator of the 2nd/Divisor Fraction (e.g. if denom

factors = np.transpose(coeffExpFrac)
denominator = factors[1, 1]
print(denominator)

# factors[:, 1][0] is the Numerator of the 2nd Fraction
factors[:, 1][0] += denominator
print(repr(factors))
```

```
2
array([[7, 5],
       [1, 2]])
```

Fraction division is a multiplication by its reciprocal of the 2nd/Divisor Fraction

In [130…
```
factors[:, 1] = factors[:, 1][::-1]
print(repr(factors))
```

```
array([[7, 2],
       [1, 5]])
```

In [131…
```
result = []

for A, B in factors:
    # Set the Inputs by using setInputs()
    setInputs(A, B)
```

```
    # Execute the Circuit by using execCirc()
    resultMul = execCirc()
    result.append(resultMul)
    print(f'Expression: {A}*{B} = {resultMul}')
```

```
Expression: 7*2 = 14
Expression: 1*5 = 5
```

In [132…

```
# New Coefficient for the integrated expression
integCoef = Fraction(result[0], result[1])
integCoef
```

Out[132… `Fraction(14, 5)`

In [133…

```
# New Exponent for the integrated expression
integExp = coeffExpFrac[1][0]+1
integExp
```

Out[133… `4`

In [134…

```
integCoef*x**integExp
```

Out[134…

$$\frac{14x^4}{5}$$