Chapter 3 - Math Functions and the NumPy Library

Objectives:

- Use mathematical operators to perform calculations
- Apply math functions to float and integer variables
- Import the NumPy library
- Use mathematical functions and constants defined in NumPy

3.0 Simple Mathematical Operations

Here are examples of how to add, subtract, multiply and divide in Python using the `+, −, ∗, /` operators

```
x = 5        # assign a value to x
y = 2        # assign a value to y

z1 = x + y    # adds x and y
z2 = x − y    # subtracts y from x
z3 = x ∗ y    # multiplies x times y
z4 = x / y    # divides x by y
```

If you want to see the results of these operations you can run this code cell and open the Variable Inspector. The spaces around the operators and equal sign are optional and are included for readability, but aren't necessary.

To raise a variable to a power, Python uses the double asterisk like this:

```
z5 = 2 ∗∗ 4    # raises 2 to the 4th power to give 16
```

Here are a couple less common operators (`%` and `//`) that are built in to native Python:

```
x = 17; y = 3  # assign values to x and y (Notice we use ";" to define two variables on the same line.)
a = x // y     # integer division (truncates any decimal values to return an integer)
b = x % y      # remainder of division of x by y (also called the modulo operator)

# let's print these values
print(f'integer division example:  {x} // {y} = {a}')
print(f'modulo arithmetic example:  {x} % {y} = {b}')
```

✅ Skill Check 1

Calculate the value of 2^{10}. Display your result in a print statement. This value is the number of bytes in a kilobyte of computer memory.

3.1 Order of operations: use parentheses sparingly.

Much like when you use a calculator, you can use parentheses to tell Python to group and execute some operations before applying others. Generally, Python performs multiplication and division first and then applies addition and subtraction. It is a good idea to only use parentheses when you need to so you don't needlessly clutter up a calculation. Too many parentheses can make expressions harder rather than easier to read.

```
x = 5; y = 2                 # assign values to x an y
a = 2; b = 4; c = 6          # assign values to a,b,c

z1 = a*x + b*y               # multiplication is computed before addition by default
z2 = a*(x+y)                 # parentheses force addition to be done before multiplication
z3 = a*b/c                   # multiply, then divide
z4 = a/(b*c)                 # parentheses are needed if a is divided by product b*c
z5 = a*x + b*x**2 + c*x**3   # exponentiation example
z6 = a*b*2**(a-b)/c          # division by c is performed after exponentiation
z7 = (a*b/c) * 2**(a-b)      # same result as above but clearer and less subject to misunderstanding
z8 = a*(b*(1/c*(2**(a-b))))  # same result as above, but too many parentheses make this hard to read
```

- If you wish to see the results of these calculations, open the Variable Explorer or you can create a print statement to print the values.

✅ Skill Check 2

The following code cell defines variables `x` and `y`. Calculate the quantity $z = \frac{2x^3}{y^2} + \frac{4x+y^2}{(x-2y)^2}$. Print your value for `z`.

```
x = 2.5
y = 6
z =            # insert your expression for z here
```

3.2 Commonly used built-in functions include `round()` and `abs()`

Built-in functions are functions that come with native Python. For trig and other math functions we'll have to load a specialized library called numpy (see below).

To take the absolute value use the `abs()` function:

```
print(abs(-2))
```

The `abs()` function returns the modulus of complex numbers:

```
c = 1+1j
print("the modulus of",c,"is",abs(c))
```

3.2.1 Some functions (including `round()`) have default values for some arguments

We can round a rumber to the nearest whole number by simply passing the number to the `round()` function like this:

```
print(round(3.14159))
```

However, the `round()` function has an optional argument that specifies the number of digits to keep after the decimal point. For example, if we wanted to keep 2 decimal places we could write:

```
print(round(3.14159,2))
```

Native Python does not have sophisticated mathematical functions like trig functions, Bessel functions, etc. For those, we need to load in a specialized library as we'll see in the next section.

3.3 NumPy Library

Libraries are collections of one or more modules that contain functions and other objects that provide additional functionality to the basic Python package, much like a new instrument adds functionality to a research lab. But just as too much equipment can sometimes complicate and slow down a lab, too many libraries can slow down your code. So, only import the libraries (or parts of libraries) that you need.

Library vs Module. A library is technically a collection of modules, but the terms are often used interchangeably, especially since many libraries only consist of a single module, so don't worry if you mix them. Numpy, for example, has many modules, each corresponding to groups of mathematical functions (such as a the linear algebra module `linalg`).

One of the most widely-used libraries for scientific computation is NumPy (pronounced "num-pie" and not something that rhymes with "grumpy"). NumPy provides a wide range of numerical functions that are not included in native Python. For example, Python doesn't natively include the sine and cosine functions. To use this library, we add an import statement to the beginning of our program, and then reference the functions in this library.

🔆 Example: Area of Circle

Here's a sample program that imports the `numpy` library and uses it to access the value of π to calculate the area of a circle:

```
import numpy as np   # import the numpy library and abbreviate it as np

r = 10               # define the radius of a circle
A = np.pi * r**2     # use the numpy library value for pi to calculate the area of a circle
print("area of circle = ",A)
```

Here are a few things to notice about this code:

- We import the NumPy library at the beginning of the code before it is used
- When we import NumPy we give it a shortened nick-name "np". This abbreviation saves typing and keeps our code cleaner. While you can use any nick-name you want, please, please always use "np" for NumPy. It will make your code easier to follow for other programmers (and yourself).
- To access a constant or function in the library, we use the following format: module_name.thing_name. In this case "np" is the name of the module (i.e library) and "pi" is the name of the object within the library.

3.3.1 List of commonly-used NumPy functions and constants

NumPy has trig functions, square root, log and many others. It also has a function to convert radians to degree and vise versa. The following is a small fraction of the total number of math functions available in NumPy.

```
x = 0.5   # define a value for x
y = 3     # define a value for y

# constants
np.pi         # pi
np.e          # e
np.inf        # infinity
np.nan        # not a number

# logarithmic and exponential functions
np.sqrt(x)    # square root(x)
np.exp(x)     # e^x
np.log(x)     # ln(x)
np.log10(x)   # log base 10(x)
np.log2(x)    # log base 2(x)
```

```
# trigonometric functions
np.sin(x)     # sin(x)
np.cos(x)     # cos(x)
np.tan(x)     # tan(x)

# degree-radian conversions
np.deg2rad(x)     # converts degrees to radians
np.rad2deg(x)     # converts radians to degrees

# inverse trigonometric functions
np.arcsin(x)  # asin(x)
np.arccos(x)  # acos(x)
np.arctan(x)  # atan(x)

# hyperbolic functions
np.sinh(x)    # hyperbolic sin
np.cosh(x)    # hyperbolic cos
np.tanh(x)    # hyperbolic tan
```

✅ Skill Check 3

Calculate $\sin\theta$, $\cos\theta$ and $\tan\theta$ for $\theta = 20^{\circ}$. Don't forget to convert your angle into radians before applying the trig function.

3.3.2 List of fundamental physical constants

We'll copy our previous list of physical constants from Chapter 1 for use in the examples below.

```
c = 299792458             # definition of the speed of light in m/s
h = 6.626e-34             # Planck's constant (J s)
hbar = 1.0546e-34         # "h bar" = h / (2*pi)  (J s)
k = 1.3806e-23            # Boltzmann's constant (J/K)
G = 6.6743e-11            # Gravitational constant (m^3/kg/s^2)
e = 1.602177e-19          # fundamental charge (C)
me = 9.10938e-31          # mass of electron (kg)
epsilon0 = 8.854188e-12   # vacuum permittivity (F/m)
u = 1.66054e-27           # atomic mass unit (kg)
```

✅ Skill Check 4

Use the fundamental physical constants defined above and the numpy value of π to calculate the inverse fine structure constant used in atomic physics: $\alpha^{-1} = 4\pi\epsilon_0\hbar c/e^2$. Display the calculated value using a print statement.

☀ Example: Free fall

Imagine dropping a rock off a tall cliff. The velocity of the rock (ignoring air resistance) when it hits the ground is given by $v = \sqrt{2gh}$. Define variables to store the acceleration of gravity (9.8 m/s^2) and the height of the cliff (30 meters). Calcuate the velocity of the rock when it hits th ground and display your answer in a print statement.

Solution

We import numpy to use the `sqrt()` function, define the variables, calculate the velocity and display the results showing 2 decimal places. Note, we don't actually have to import NumPy since we already imported it. We include it here to show a complete, self-contained example.

```
import numpy as np

g = 9.8   # acceleration of gravity (m/s^2)
h = 30    # height of cliff (m)

v = np.sqrt(2*g*h)    # free-fall velocity

print(f"velocity = {v:.2f} m/s")
```

■ done

✅ Skill Check 5

The mean velocity of a particle with mass m in a gas with temperature T is given by $v_{rms} = \sqrt{3kT/m}$. Calculate and display v_{rms} in m/s for hydrogen molecules ($m = 2.02u$) and oxygen molecules ($m = 16.00u$) at room temperature (300 K).

Key Points

- Base Python includes a few mathematical operators including + - * / % //
- Use parentheses in complex mathematical expressions sparingly to avoid confusion
- Some Python functions have optional arguments that have default values. It is often helpful to explore the these arguments to increase productivity.
- Libraries are commonly used in Python to extend funcionality
- Numpy is a common library used for mathematical computation that has trig, log, square root, and many more advanced functions

This tutorial is a modified adaptation of "Python for Physicists" © Software Carpentry