**ABSTRACTION THE FIRST PRINCIPLES**

Abstraction is the process of turning complex ideas into simple ones. It is removing characteristics from something so that only the essential ones remain. Abstraction is **the process of hiding the internal details of an application from the outer world**. Abstraction is used to describe things in simple terms.

For example

print(“Hello World!”)

This is code written in Python. We are just using the print function to output the text “Hello world” on the screen.

To do this, you just need to know how to use the print function.

You don't need to understand how it works under the hood.

But it is good sometimes to understand how a certain function works in the background in order to use it more effectively.

Programming with classes is another way of creating abstractions in software. We begin, however, by thinking about objects. An object is a conceptual model for a category of things, real or imagined, that has a specific responsibility within our program. For example, we might think of an object that holds and provides identifying information about a person.

Objects have state and behavior that allow them fulfill their responsibility. The person object may have state like "given name" and "family name". It may also have related behaviors like "show south name" and "show west name". Following is a graphical depiction of our person object.

Object: person

Responsibility:

To hold and display identifying information

Behavior State

-show west name given name

-show south name family name

For example, let's say you want to use the Python statistics module, which is a build-in module in Python. This means that Python already comes with the module in its library.

You don't need to import it with PIP.

Let's say I want to use the mean function:

from statistics import mean

randomList = [-1.0, 2.5, 3.25, 5.75]

print(mean(randomList))

If there is no data, the [Statistics.error](https://docs.python.org/3/library/statistics.html#statistics.StatisticsError) will be raised. This will print out 2,625.But how does that work inside?

def mean(data):

"""

Return the sample arithmetic mean of data.

>>> mean([1, 2, 3, 4, 4])

2.8

>>> from fractions import Fraction as F

>>> mean([F(3, 7), F(1, 21), F(5, 3), F(1, 3)])

Fraction(13, 21)

>>> from decimal import Decimal as D

>>> mean([D("0.5"), D("0.75"), D("0.625"), D("0.375")])

Decimal('0.5625')

If ``data`` is empty, StatisticsError will be raised.

"""

T, total, n = \_sum(data)

if n < 1:

raise StatisticsError('mean requires at least one data point')

return \_convert(total / n, T)

This is the internal code that runs when you use the statistics built in module in Python library.