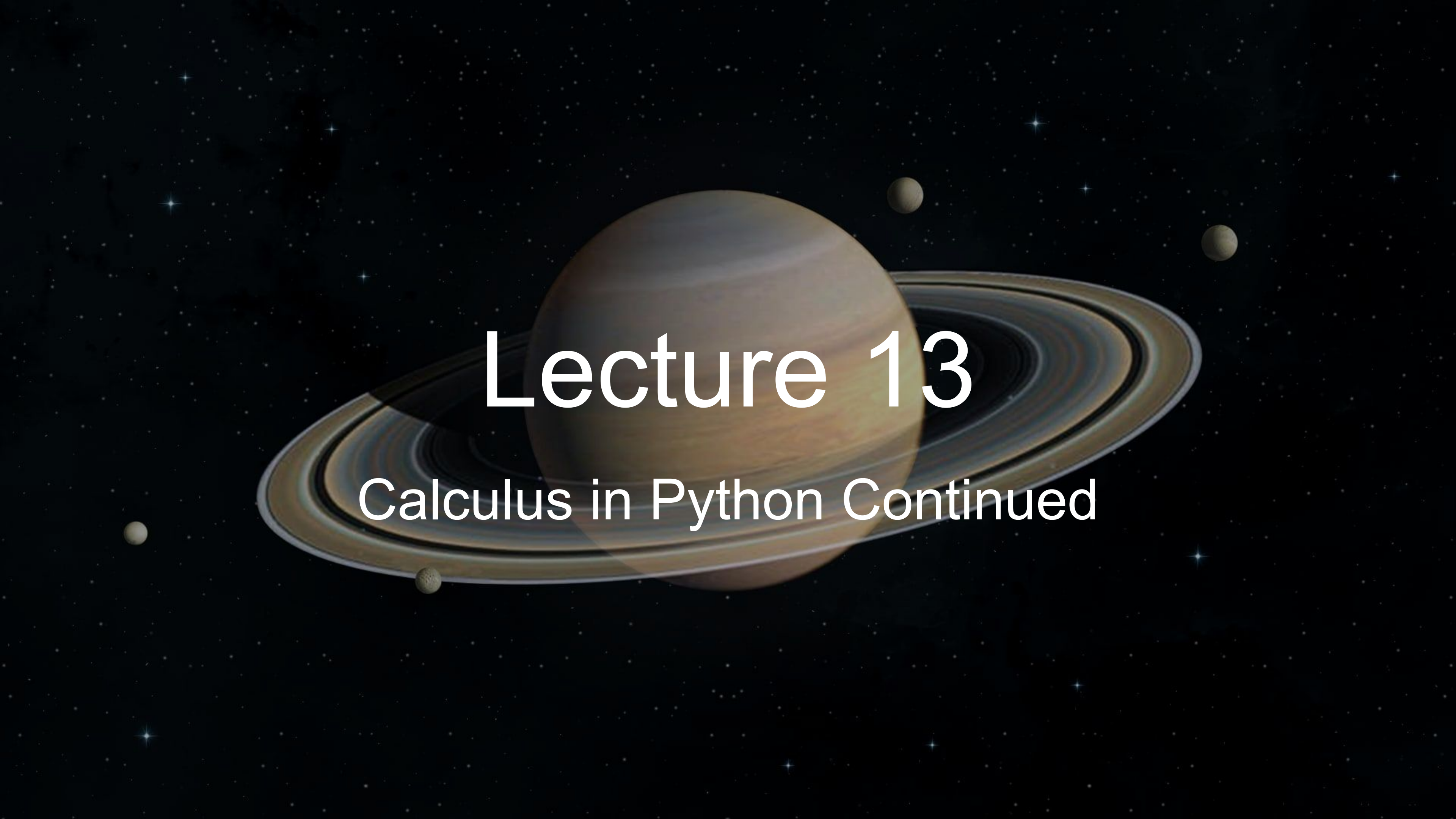
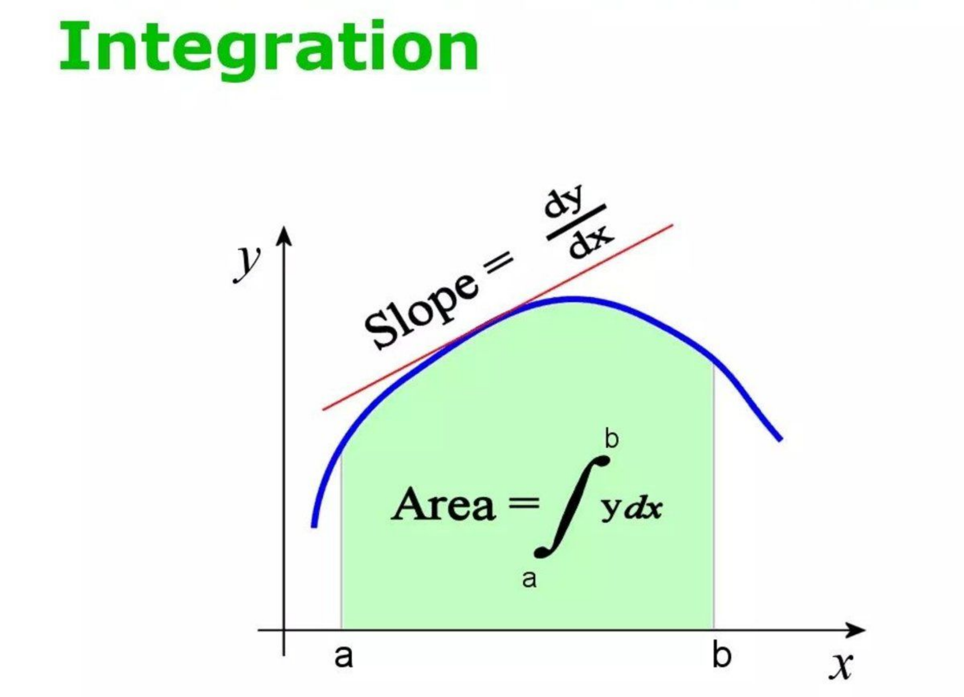


Lecture 13

Calculus in Python Continued

Integration

- Analytically defined as the area under the curve of a function
- A sort of inverse to derivatives. If you take the integral of a derivative you get the original function back
- Unlike derivatives, we always need to be careful of our limits



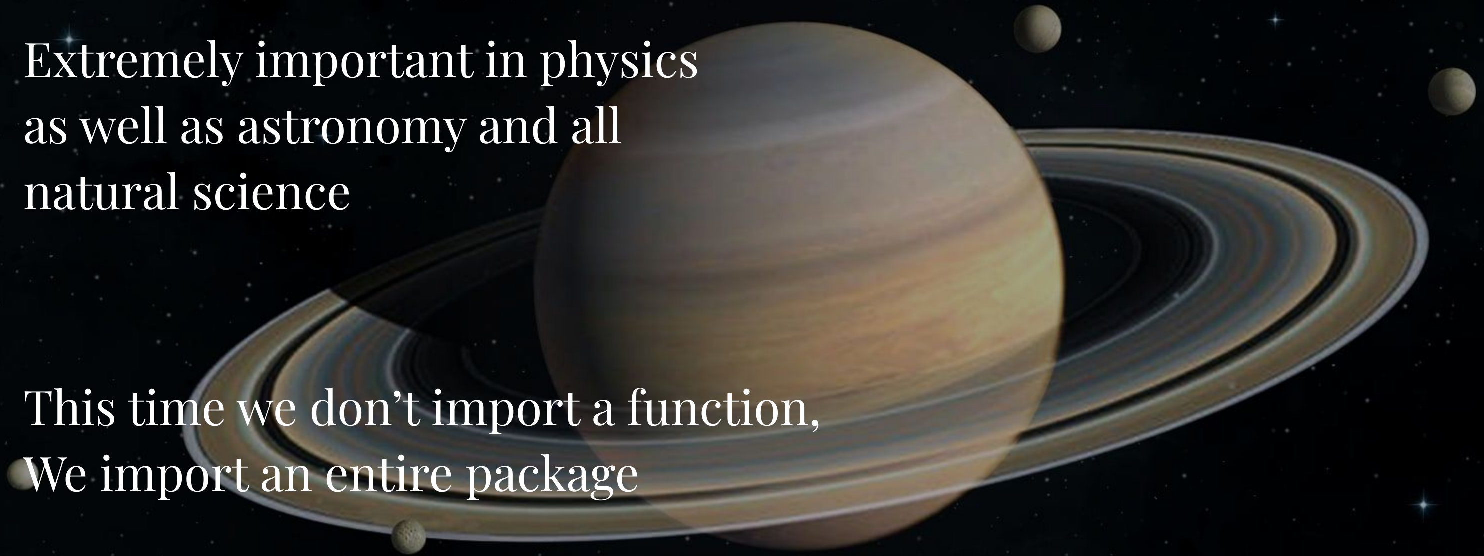
Numerical Integration

- Extremely important in physics as well as astronomy and all natural science

- This time we don’t import a function,
- We import an entire package

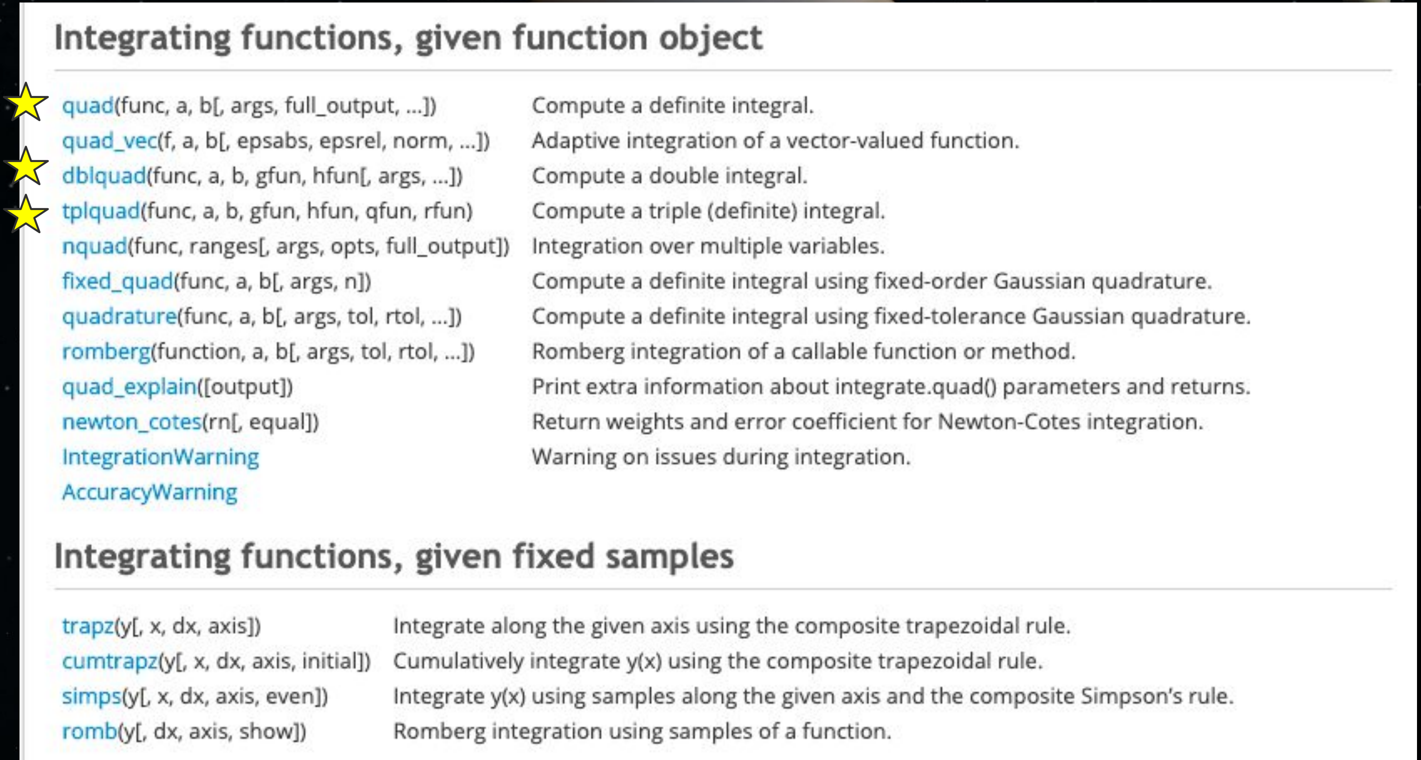
```
import scipy.integrate as integrate

from scipy.integrate import ...
```



Options

`Scipy.integrate` has tons of different options



Integrating functions, given function object

★ quad(func, a, b[, args, full_output, ...])	Compute a definite integral.
quad_vec(f, a, b[, epsabs, epsrel, norm, ...])	Adaptive integration of a vector-valued function.
★ dblquad(func, a, b, gfun, hfun[, args, ...])	Compute a double integral.
★ tplquad(func, a, b, gfun, hfun, qfun, rfun)	Compute a triple (definite) integral.
nquad(func, ranges[, args, opts, full_output])	Integration over multiple variables.
fixed_quad(func, a, b[, args, n])	Compute a definite integral using fixed-order Gaussian quadrature.
quadrature(func, a, b[, args, tol, rtol, ...])	Compute a definite integral using fixed-tolerance Gaussian quadrature.
romberg(function, a, b[, args, tol, rtol, ...])	Romberg integration of a callable function or method.
quad_explain([output])	Print extra information about integrate.quad() parameters and returns.
newton_cotes(rn[, equal])	Return weights and error coefficient for Newton-Cotes integration.
IntegrationWarning	Warning on issues during integration.
AccuracyWarning	

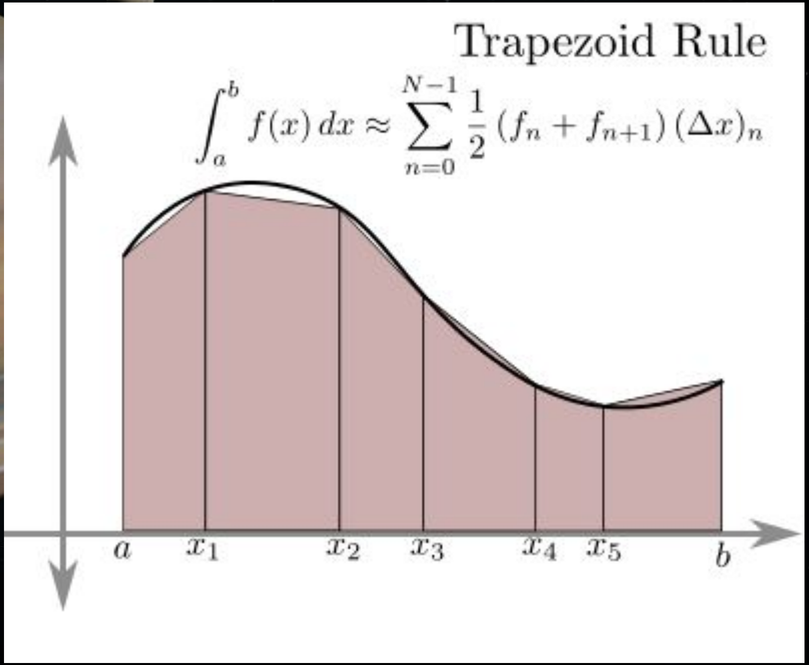
Integrating functions, given fixed samples

trapz(y[, x, dx, axis])	Integrate along the given axis using the composite trapezoidal rule.
cumtrapz(y[, x, dx, axis, initial])	Cumulatively integrate y(x) using the composite trapezoidal rule.
simps(y[, x, dx, axis, even])	Integrate y(x) using samples along the given axis and the composite Simpson's rule.
romb(y[, dx, axis, show])	Romberg integration using samples of a function.

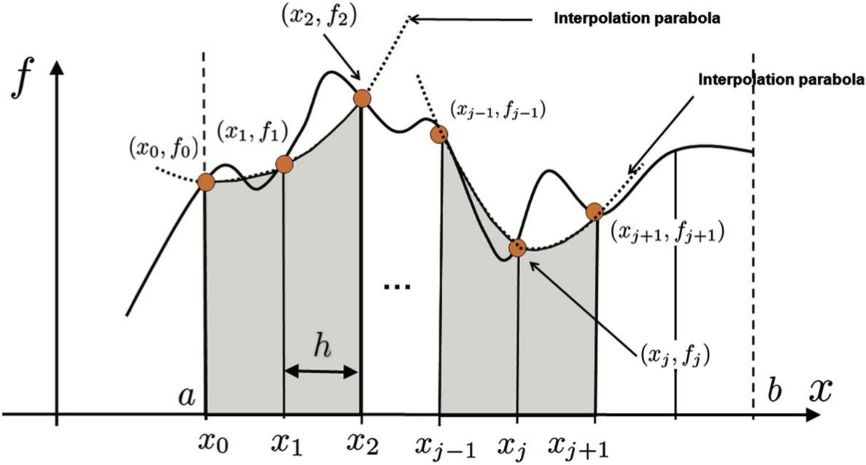
Types of Integration: Trapezoid Rule

- An integration approximation which uses trapezoids to get a close calculation of the area under the curve.

- The approximation is formed by drawing trapezoids between points on the curve and summing the area of those trapezoids


Trapezoid Rule
$\int_a^b f(x)\,dx \approx \sum_{n=0}^{N-1} \frac{1}{2}(f_n + f_{n+1})(\Delta x)_n$
a
x_1
x_2
x_3
x_4
x_5
b

Types of Integration: Simpson's Rule

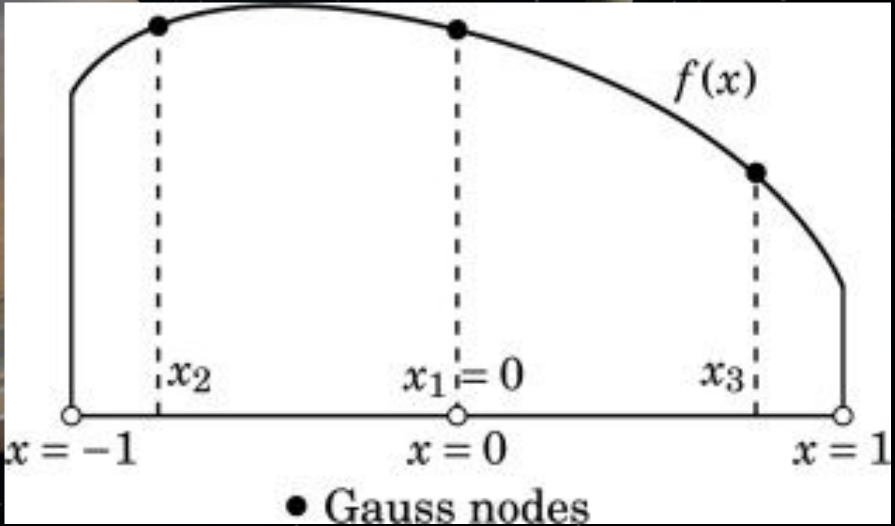


- Another approximation method of computing integrals.
- This time, instead of drawing trapezoids between points, we draw parabolas
- The integral of those parabola are then approximated and then summed

Types of Integration: Gaussian Quadrature

- Another approximation method which uses a weighted sum of function values within the bounds of the integral

- We won't write out this method by hand. Lucky for us Scipy has a built in function for us to use for this.

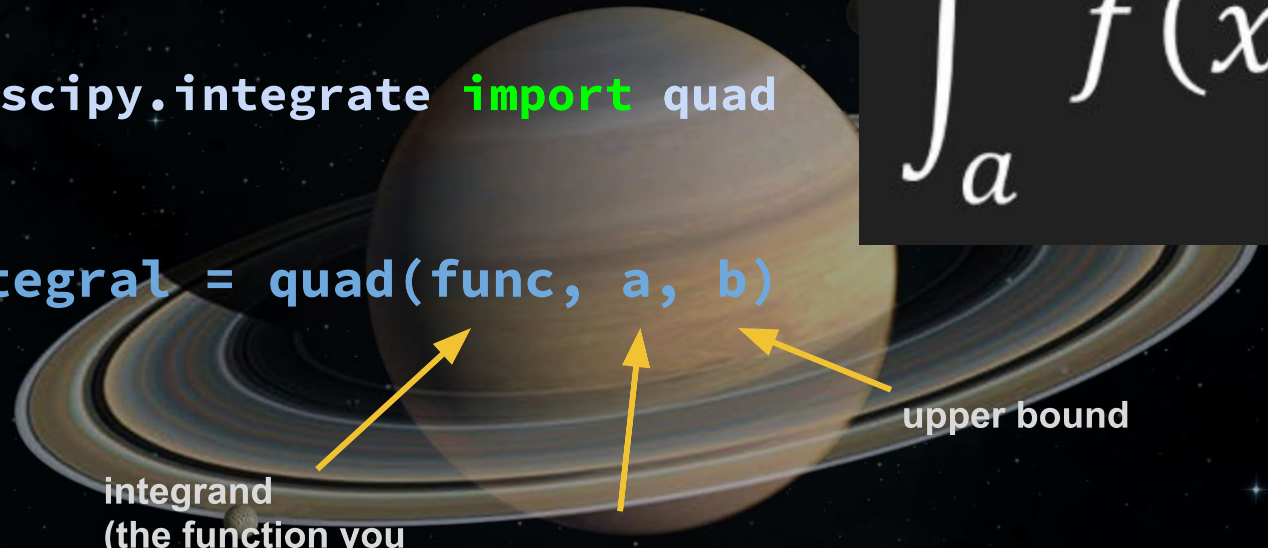


Single-variable integral....

$$\int_a^b f(x)dx$$

```
from scipy.integrate import quad
```

```
integral = quad(func, a, b)
```



integrand
(the function you
want to integrate)

lower bound

upper bound

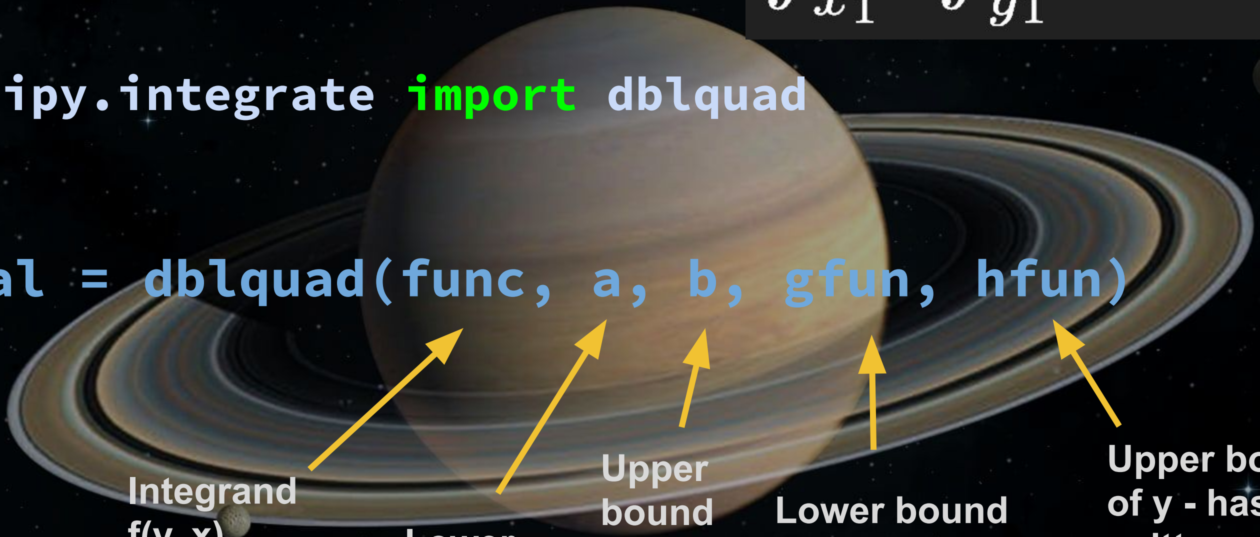
Result can be extracted by calling `integral[0]`
Error can be extracted by calling `integral[1]`

Double integral....

$$\int_{x_1}^{x_2} \int_{y_1}^{y_2} f(x, y) dy dx$$

```
from scipy.integrate import dblquad
```

```
integral = dblquad(func, a, b, gfun, hfun)
```



Integrand f(y, x)

Lower bound of x

Upper bound of x

Lower bound of y - has to be written as a function y(x)

Upper bound of y - has to be written as a function y(x)

Check documentation
Order matters here

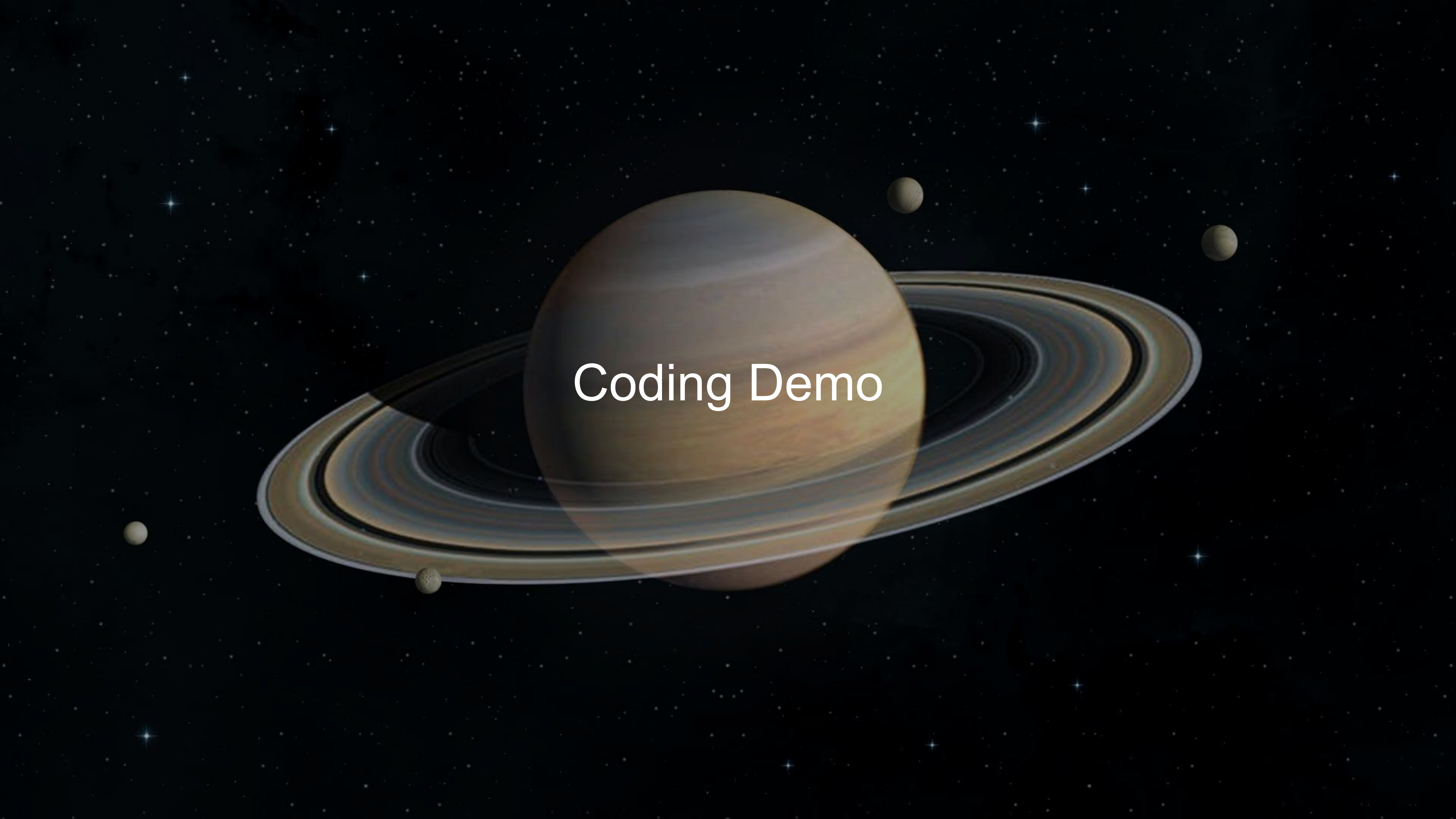
Double Integral example

$$\int_3^6 \int_1^{x^2} xy \; dy dx$$

```
def func(y, x):
    return x*y
```

```
I = dblquad(func, 3, 6, lambda x:1, lambda x: x**2)
```

Similarly, you can also do a triple integral using `tplquad`



Coding Demo