5 better ways to write Python

In an astronomical distance calculator

The goal of the program:

$$D_{\mathrm{M}} = \begin{cases} D_{\mathrm{H}} \frac{1}{\sqrt{\Omega_k}} \sinh\left[\sqrt{\Omega_k}\, D_{\mathrm{C}}/D_{\mathrm{H}}\right] & \text{for } \Omega_k > 0 \\ D_{\mathrm{C}} & \text{for } \Omega_k = 0 \\ D_{\mathrm{H}} \frac{1}{\sqrt{|\Omega_k|}} \sin\left[\sqrt{|\Omega_k|}\, D_{\mathrm{C}}/D_{\mathrm{H}}\right] & \text{for } \Omega_k < 0 \end{cases}$$

$$D_{\mathrm{C}} = D_{\mathrm{H}} \int_0^z \frac{dz'}{E(z')} \qquad D_{\mathrm{H}} \equiv \frac{c}{H_0}$$

$$E(z) \equiv \sqrt{\Omega_{\mathrm{M}}\,(1+z)^3 + \Omega_k\,(1+z)^2 + \Omega_\Lambda}$$

Hogg (2000), Distance Measures in Cosmology

5: Write Helper Functions instead of complex expressions

```
def E_z(z,omega_m,omega_lambda,omega_k):
    return 1/np.sqrt((1+z)**3 * omega_m + (1+z)**2 * omega_k + omega_lambda)
```

```
def num_int(function,low,upper,steps):
```

```
def D_M(D_C,omega_k,D_H):
    if omega_k > 0:
        return D_H/np.sqrt(omega_k) * np.sinh(D_C/D_H * np.sqrt(omega_k))
    elif omega_k == 0:
        return D_C
    else:
        omega_k = np.abs(omega_k)
        return D_H/np.sqrt(omega_k) * np.sin(D_C/D_H * np.sqrt(omega_k))
```

20: Prefer raising exceptions to returning None

```
def E_z(z,omega_m,omega_lambda,omega_k):
    term = np.sqrt((1+z)**3 * omega_m + (1+z)**2 * omega_k + omega_lambda)
    if term == 0:
        return None
    else:
        return 1/term
```

```
def E_z(z,omega_m,omega_lambda,omega_k):
    term = np.sqrt((1+z)**3 * omega_m + (1+z)**2 * omega_k + omega_lambda)
    if term == 0:
        raise ValueError("E(z) cannot be zero. Try again with z and omega_lambda not equal to zero.")
    else:
        return 1/term
```

23: Provide Optional Behaviour with Keyword Arguments

```
def distcalc(z,omega_m,omega_lambda,omega_k):
    ...
    return distance
```

```
def distcalc(z,omega_m=0.28,omega_lambda=0.72,omega_k=0):
    ...
    return distance
```

84: Write docstrings for Every Function, Class, and Module

```
def distcalc(z,omega_m=0.28,omega_lambda=0.72,omega_k=0,H0 = 75e3):
    '''Calculates the transverse comoving distance based on the formula defined in D_M.
    As an intermediary step, calculates comoving distance, D_C, by numerically integrating over the function E(z)
    and multiplying by the Hubble distance, D_H.
    All formulas implemented in this function are taken from Hogg (2000), Distance Measures in Cosmology.
    '''
```

```
print(num_int.__doc__)
print(D_M.__doc__)
print(distcalc.__doc__)
```

```
Numerically integrates a function from 0 to an upper bound using Simpson's Rule.

Defines the formula used to calculate transverse comoving distance, based on the value of Omega_k.

Calculates the transverse comoving distance based on the formula defined in D_M.
    As an intermediary step, calculates comoving distance, D_C, by numerically integrating over the function E(z)
    and multiplying by the Hubble distance, D_H.
    All formulas implemented in this function are taken from Hogg (2000), Distance Measures in Cosmology.
```

4: Prefer Interpolated f-strings

```python
print(f"The distance to an object of redshift {zinput:.2f} is {zdist:.2f} Megaparsecs.")

print("The distance to an object of redshift %f is %f Megaparsecs." % (zinput,zdist))
```