- Let's start by going over what we have so far:
- **piecewise.py**
- Piecewise.py defines a class `PiecewisePolynomial` for handling piecewise polynomials, allowing evaluation, differentiation, integration, and multiplication of piecewise functions. The `PiecewisePolynomial` class encapsulates functionalities to manipulate piecewise polynomials efficiently, enabling various mathematical operations and computations on functions defined by segments or intervals.
- Functionalities and Methods (piecewise.py)
- Class `PiecewisePolynomial`:
    - Initialization (`__init__`):
    - Parameters:
        - `c`: Coefficients of the piecewise polynomial.
        - `x`: Breakpoints defining the intervals where the polynomial changes.
    - Functionality:
        - Checks the shape and dimensions of input coefficients and breakpoints.
        - Initializes the instance variables `self.breakpoints` (storing breakpoints) and `self.coeffs` (storing coefficients) for the piecewise polynomial.
- Function for Evaluation (`__call__`):
    - Parameters:
        - `xp`: Point or array of points where the piecewise polynomial is to be evaluated.
        - `break_down`: Optional flag to determine whether the evaluation considers a breakdown.
    - Functionality:
        - Evaluates the piecewise polynomial at given point(s) `xp`.
        - Utilizes `_evaluate_at_point` method to evaluate the polynomial.
- Helper Methods:
    - `_evaluate_at_point`: Evaluates the polynomial at a single point.
    - `_evaluate_polynomial`: Computes the value of the polynomial at a given point.
    - `_get_coefs`: Obtains coefficients based on the evaluated point.
- Differentiation and Integration:
    - `derivative`: Computes the derivative of the piecewise polynomial.
    - `antiderivative`: Computes the antiderivative (integral) of the piecewise polynomial.
    - `integrate`: Calculates the definite integral of the piecewise polynomial between given bounds.
- Multiplication of Polynomials:
    - `mult`: Performs multiplication between two piecewise polynomials.
- Usage:
    - This class provides a versatile framework to handle piecewise polynomials, offering methods for evaluation, differentiation, integration, and multiplication.
    - It allows users to work with complex functions defined by intervals with different polynomial expressions.

- ## earth_model.py
- earth_model.py defines a class `Prem` that models the Earth's interior structure based on the Preliminary Reference Earth Model (PREM). This code mainly constructs an Earth model using the PREM parameters and provides a comprehensive interface to extract various physical properties at different depths within the Earth.
- Methods and Functionalities (earth_model.py)
- Constants and Parameters Initialization:
    - The code initializes various parameters related to Earth's structure (density, seismic velocities, quality factors, etc.) derived from the PREM model.
    - These parameters are organized and stored in arrays like `_density_params`, `_vp_params`, `_vs_params`, `_q_kappa_params`, and `_q_mu_params`.
- Class `Prem`:
    - Initialization (`__init__`):
        - The `Prem` class is initialized with default parameters such as breakpoints (`_bps`) and various parameters for density, seismic velocities, and quality factors.
        - It instantiates instances of `PiecewisePolynomial` (from the `piecewise` module) for density, P-wave velocity (`vp`), S-wave velocity (`vs`), and quality factors for κ (`q_kappa`) and μ (`q_mu`).
    - Methods for Earth Properties:
        - `density`, `vp`, `vs`, `qkappa`, `qshear`: These methods evaluate density, P-wave velocity, S-wave velocity, κ, and μ quality factors at a given radius (`r`).
        - `bulk_modulus` and `shear_modulus`: Compute the bulk modulus and shear modulus at a given radius (`r`).
        - `mass` and `moment_or_inertia`: Calculate mass and moment of inertia inside a given radius (`r`) and inner radius (`r_inner`).
        - `gravity` and `pressure`: Estimate gravity and pressure at a given radius (`r`).
    - Tabulation Methods:
        - `tabulate_model_inwards` and `tabulate_model_outwards`: Generate a record array representing the Earth model with evaluations at different depths considering discontinuities.
- Usage:
    - The `Prem` class provides methods to evaluate various Earth properties at different depths/radii based on the PREM model.
    - It allows the computation of seismic velocities, density, quality factors, moduli, mass, gravity, and pressure at various depths within the Earth.
- Now let’s look at the new stuff!:
- ## Part1_velocity_modulus.ipynb
    - The code implements parts of PREM.
    - 1. Seismic Velocities (Vp and Vs) Calculation:
        - P-wave Velocity (Vp): The P-wave velocity in PREM is parameterized as a function of radius (depth) within different layers of the Earth. The formula is represented as a piecewise function:

$$\$V_P(r) = \begin{cases} p_{0,0} + p_{0,1}r + p_{0,2}r^2 + p_{0,3}r^3 & r \leq 1221.5 \text{ km} \\ p_{1,0} + p_{1,1}r + p_{1,2}r^2 + p_{1,3}r^3 & 1221.5 \leq r \leq 3480.0 \text{ km} \\ \vdots & \vdots \\ p_{12,0} + p_{12,1}r + p_{12,2}r^2 + p_{12,3}r^3 & 6368.0 \leq r \leq 6371.0 \text{ km} \end{cases}$$

- S-wave Velocity (Vs): The S-wave velocity in PREM is calculated similarly, also as a piecewise function based on radius within different Earth layers:

$$V_S(r) = \begin{cases} s_{0,0} + s_{0,1}r + s_{0,2}r^2 + s_{0,3}r^3 & r \leq 1221.5\,\text{km} \\ s_{1,0} + s_{1,1}r + s_{1,2}r^2 + s_{1,3}r^3 & 1221.5 \leq r \leq 3480.0\,\text{km} \\ \vdots & \vdots \\ s_{12,0} + s_{12,1}r + s_{12,2}r^2 + s_{12,3}r^3 & 6368.0 \leq r \leq 6371.0\,\text{km} \end{cases}$$

- Here, coefficients p and s represent the parameters for P-wave and S-wave velocity equations, respectively. The equations are evaluated based on the radius (/depth) to obtain the seismic velocities at various depths within the Earth.

- 2. Bulk and Shear Moduli Calculation:
- Bulk Modulus (kappa) and Shear Modulus (mu):
  - These moduli are fundamental properties of materials that describe their response to stress and strain.
- The formulas to compute these moduli from seismic velocities and density are:

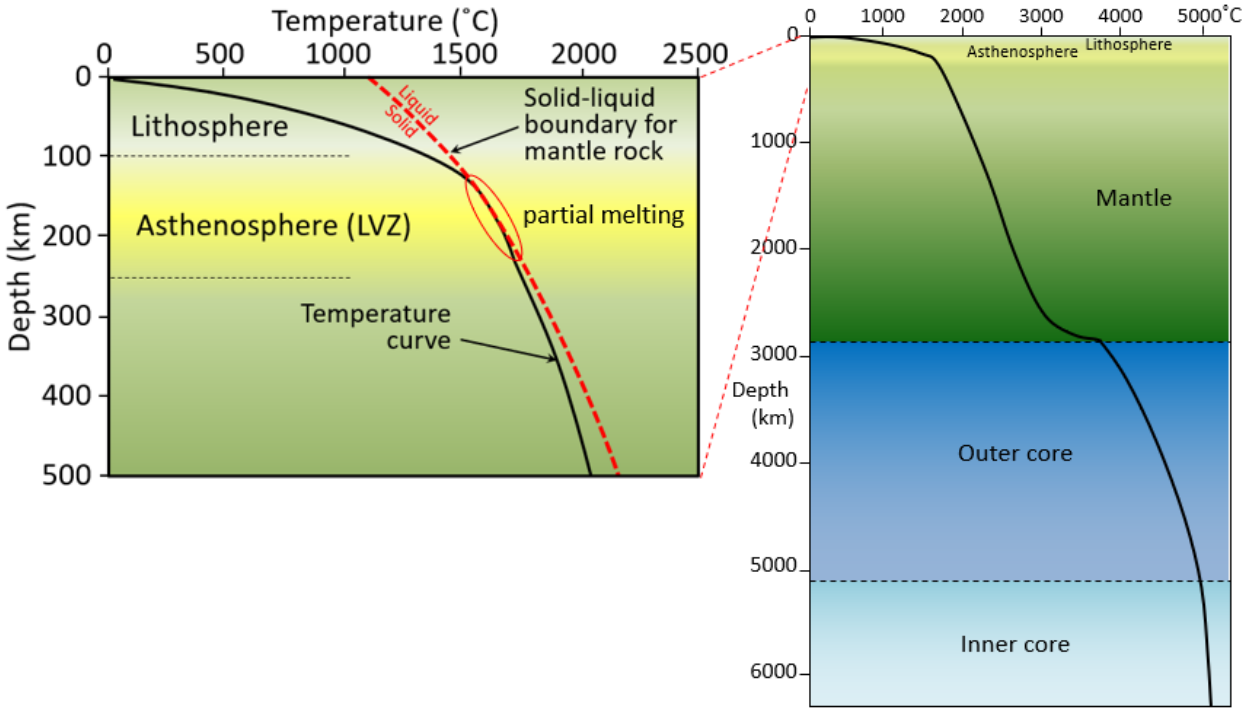
$$\kappa = \rho \cdot V_P^2$$

$$\mu = \rho \cdot V_S^2$$

-
- Where:
  - kappa is the bulk modulus.
  - mu is the shear modulus.
  - V_P is the P-wave velocity.
  - V_S is the S-wave velocity.
  - rho is the density of the Earth's material at a specific depth.
- These moduli provide information about the Earth's material properties, like its ability to transmit seismic waves and respond to deformation.
- The calculations in the code involve evaluating the appropriate polynomial equations for velocities (Vp and Vs) based on depth and then using these velocities to derive the bulk and shear moduli at different depths within the Earth.

Things are getting thermo……

- Now we start free-styling a little bit, all of the initial boxes are ticked. We have density, thermal conductivity, and an estimation of thermal gradient
- I imported approximate temperatures for my depths in the earth thanks to Steven Earle:



Layer	Depth (kilometers)	Temperature increase (Celsius)	Temperature increase rate (Degrees per kilometre)
Lithosphere	0 to 100	0 to 1400	14
Asthenosphere	100 to 250	1400 to 1700	2
Mantle	250 to 1000	1700 to 2100	0.53

	1000 to 2000	2100 to 2600	0.5
	2000 to 2890	2600 to 3800	1.35
	2890 to 4000	3800 to 4600	0.72
Outer Core			
	4000 to 5100	4600 to 5000	0.36
Inner core	5100 to 6370	5000 to 5100	0.079

- This gives us temperature and thermal gradient as a function of depth!
- https://geo.libretexts.org/Bookshelves/Geology/Physical_Geology_(Earle)/09%3A_Earths_Interior/9.02%3A_The_Temperature_of_Earths_Interior#:~:text=The%20temperature%20is%20around%201000,depending%20on%20the%20tectonic%20setting.
- Then I estimated the thermal conductivity in several layers of the earth based on the average thermal conductivity values of commonly found minerals in that layer:

Layer	Depth (km)	Common Minerals	Estimated Thermal Conductivity (W/m/K)
Crust	0-50	Feldspar, Quartz, Clay minerals	2.05
Upper Mantle	50-410	Olivine, Pyroxene, Garnet	3.8
Transition	410-660	Spinel, Perovskite, Magnesiowüstite	6.1
Lower Mantle	660-2,890	Bridgmanite, Ferropericlase, Calcium-Perovskite	8.1
Outer Core	2,890-5,150	Iron, Nickel	10 - 15
Inner Core	5,150-6,371	Iron, Nickel	20 - 80

- Now we can finally calculate a nice crude model of heat flow inside of the earth! We have all the ingredients!
- I have merged them into one referenceable table:

Layer	Depth (kilometers)	Temperature increase (Celsius)	Temperature increase rate (Degrees per kilometer)	Common Minerals	Estimated Thermal Conductivity (W/m/K)
Lithosphere	0 to 100	0 to 1400	14	Feldspar, Quartz, Clay minerals	2.05
Asthenosphere	100 to 250	1400 to 1700	2	Olivine, Pyroxene, Garnet	3.8
Mantle	250 to 1000	1700 to 2100	0.53	Olivine, Pyroxene, Garnet	3.8
	1000 to 2000	2100 to 2600	0.5	Spinel, Perovskite, Magnesiowüstite	6.1
	2000 to 2890	2600 to 3800	1.35	Bridgmanite, Ferropericlase, Calcium-Perovskite	8.1
Outer Core	2890 to 4000	3800 to 4600	0.72	Iron, Nickel	10 - 15
	4000 to 5100	4600 to 5000	0.36	Iron, Nickel	10 - 15
Inner core	5100 to 6370	5000 to 5100	0.079	Iron, Nickel	20 - 80

- From this table, we can easily calculate an average model of heat flow!
- $q(t)= k*(dT/dz)$
- Here is a table of my manual calculations based on these, but you can see the python calculations in Part_2_temp_heat.ipynb
-

Layer	Depth (kilometers)	Temperature increase (Celsius)	Temperature increase rate (Degrees per	Common Minerals	Estimated Thermal Conductivity (W/m/K)	Temperature increase °/km*1km/1000m	q(t)=K*(dT/dz) W/m^2	q(t) mW/m^2

			kilometre)					
Lithosphere	0 to 100	0 to 1400	14	Feldspar, Quartz, Clay minerals	2.05	0.014	0.0287	28.7
Asthenosphere	100 to 250	1400 to 1700	2	Olivine, Pyroxene, Garnet	3.8	0.002	0.0076	7.6
Mantle	250 to 1000	1700 to 2100	0.53	Olivine, Pyroxene, Garnet	3.8	0.00053	0.002014	2.014
	1000 to 2000	2100 to 2600	0.5	Spinel, Perovskite, Magnesiowüstite	6.1	0.0005	0.00305	3.05
	2000 to 2890	2600 to 3800	1.35	Bridgmanite, Ferropericlase, Calcium-Perovskite	8.1	0.00135	0.010935	10.935
Outer Core	2890 to 4000	3800 to 4600	0.72	Iron, Nickel	12.5	0.00072	0.009	9
	4000 to 5100	4600 to 5000	0.36	Iron, Nickel	12.5	0.00036	0.0045	4.5
Inner core	5100 to 6370	5000 to 5100	0.079	Iron, Nickel	50	0.000079	0.00395	3.95

- Next I'm trying to model the heat in the earth. Wish me luck! Then we can hopefully do it to Mars too:)