Week 1 - Python basics and data types!

Welcome to PyEarth and the growing field of Data Science in Earth Science! This notebook represents your starting point to explore how computational tools and big data are transforming our understanding of the Earth and its systems. In this class we'll discuss several topics (listed below) and introduce several of the main data types used in Python programming.

The tools of data science enable us to learn about the Earth through the approaches of:

- Exploration
  - Identifying patterns in data through visualization
- Inference
  - Using data to obtain reliable insights about the Earth through the application of statistics
- Prediction
  - Use analysis of data we can observe to make informed predictions of things we cannot observe using machine learning

---

Why Python? 🐍

Python is one of the most popular programming languages for scientific computing and data analysis. Here's why:

- **Ease of Use**: Its simple syntax makes it beginner-friendly, yet powerful enough for advanced applications.
- **Versatility**: Python excels in handling diverse tasks—from data cleaning and visualization to machine learning and geospatial analysis.
- **Extensive Libraries**: Libraries like `numpy`, `pandas`, `matplotlib`, `scikit-learn`, and `xarray` are specifically designed to handle scientific and geospatial data efficiently.

Open Access and Collaboration 🌐

The Earth Sciences rely on collaborative, open-access research to address global challenges such as climate change, natural disasters, and resource management. Python aligns perfectly with this philosophy because:

- It is **free and open-source**, accessible to anyone with a computer.

- The Python community actively contributes to open-access tools, ensuring continuous innovation and knowledge sharing.
- It integrates seamlessly with open datasets, fostering transparency and reproducibility in research.

The Era of Big Data 📊

Earth Sciences are experiencing a data revolution:

- Satellite missions provide **terabytes of data** daily, capturing everything from atmospheric composition to surface deformation.
- Climate models, seismic networks, and remote sensing technologies generate massive datasets requiring advanced computational techniques to process and analyze.

Every year, Data Science provides new insights in the Earth Sciences, whether that's new methods for detecting critical mineral deposits (https://undark.org/2024/11/13/the-search-for-critical-minerals-is-going-high-tech/), understanding climate change patterns (https://science.nasa.gov/earth/explore/the-ocean-and-climate-change/), forecasting natural disasters (https://www.frontiersin.org/journals/environmental-science/articles/10.3389/fenvs.2023.1194918/full), or analyzing satellite imagery to monitor deforestation and urban growth (https://www.iceye.com/blog/deforestation-and-forest-degradation-monitoring-with-sar-satellite-imagery). The integration of advanced data analysis techniques with large-scale Earth Science datasets is revolutionizing the field, enabling more precise predictions and better-informed decisions.

Using Python, you'll learn to harness these datasets, uncover patterns, and draw meaningful insights that contribute to solving critical environmental challenges.

Understanding data types in Python 🐍

In Python, understanding data types is fundamental to writing effective and efficient code. Data types define the kind of data that a variable can hold, influencing how you can use and manipulate the data. Let’s explore the key data types you’ll encounter.

Core Data Types

1. Numeric Types

- **Integers (`int`)**: Whole numbers, e.g., `42`, `-7`.

- **Floats (`float`)**: Decimal numbers, e.g., `3.14`, `-0.001`.
- **Complex Numbers (`complex`)**: Numbers with a real and imaginary part, e.g., `3 + 4j`.

2. Text Type

- **Strings (`str`)**: Text data, enclosed in single or double quotes, e.g., `'Earth'`, `"Science"`.

3. Sequence Types

- **Lists (`list`)**: Ordered, mutable collections, e.g., `[1, 2, 3, "Earth"]`.
- **Tuples (`tuple`)**: Ordered, immutable collections, e.g., `(1, 2, 3)`.
- **Ranges (`range`)**: Represents a sequence of numbers, often used in loops, e.g., `range(5)` (0 to 4).

4. Mapping Type

- **Dictionaries (`dict`)**: Unordered, mutable collections of key-value pairs, e.g., `{'key': 'value', 'year': 2024}`.

5. Set Types

- **Sets (`set`)**: Unordered collections of unique elements, e.g., `{1, 2, 3}`.
- **Frozen Sets (`frozenset`)**: Immutable sets, e.g., `frozenset([1, 2, 3])`.

6. Boolean Type

- **Booleans (`bool`)**: Logical values, either `True` or `False`.

7. None Type

- **NoneType (`None`)**: Represents the absence of a value, e.g., `None`.

Why Are Data Types Important? 🧐

1. **Efficiency**: Different types allow Python to optimize memory usage and operations.
2. **Validation**: Ensures that data is used correctly in your code, reducing bugs.
3. **Flexibility**: Enables working with diverse data formats, from numbers and text to complex structures like lists and dictionaries.

Real-World Applications 🌏

In Earth Sciences, you’ll encounter various data types:

- Numbers for temperature, precipitation, or seismic measurements.
- Strings for metadata like locations or names of data files.
- Lists and dictionaries to organize and process large datasets.

Understanding and correctly applying these data types is crucial for writing code that processes and analyzes Earth Science data effectively. Let’s explore them in more detail in the cells ahead!

Exercise 1

In the following cells we'll create some new (Earth-related) variables. Think about what type of data is best for each variable.

- Name of our planet
- The average surface temperature in Celsius
- Whether Earth has a Moon (True/False)
- The planet's radius in kilometers
- The Moon's radius in kilometers
- A list of the planets in the solar system.

1a - Create the first two variables and use the code provided to print these variables. Bonus question, what data type did you use for Surface Temperature (use type(SurfaceT))?

(3 points)

In [2]:
```python
Name = "Earth"
print(f"Our planet is called {Name}")

SurfaceT = 15 # degrees C
print(f"Our planet's average surface temperature is {SurfaceT:.1f} $^o$C")
```

```
Our planet is called Earth
Our planet's average surface temperature is 15.0 $^o$C
```

1b - If statements & basic Arithmetric Operations

An if statement allows your program to make decisions and execute specific blocks of code based on certain conditions. It's a fundamental tool for controlling the flow of a program.

Using the cell below, create two more variables. One should indicate whether (or not) the Earth has a moon (use a boolean), the second should contain the Earth's radius. **Create an if statement so that if the Earth has a Moon the code calculates the relative volume of the Moon and Earth (you'll have to make a new variable for the radius of**

the Moon first) and prints the result (i.e., "the volume of the Moon is XX% the volume of the Earth"). If the Earth doesn't have a moon the code should print a message stating the "the Earth does not have a Moon". Remember the volume of a sphere is $(4/3) * \pi * r^3$. Assume π = 3.14.

(5 points)

```
In [14]: Moon = True
r_Earth = 6378 # km
r_Moon = 1738 # km
if Moon is True:
    v_Earth = (6378**3)*(4/3)*3.14
    v_Moon = (1738**3)*(4/3)*3.14
    v_ratio = (v_Moon/v_Earth)
    print(f"The volume of the Moon is {v_ratio*100}% the volume of the Earth
else:
    print("The Earth does not have a Moon")
```

```
The volume of the Moon is 2.023465348709695% the volume of the Earth
```

1c - Lists, loops, dictionaries, and functions

In the cell below create a list of the planets in the solar system. Use a for loop to print each planet's name and it's position from the Sun. Example output: "Mercury is the 1st planet from the Sun".

(5 points)

```
In [15]: Planets = ["Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus

# Write your for loop

for idx, planet in enumerate(Planets):
    index=idx+1
    if(index==1):
        print(f"{planet} is the {index}st planet from the sun.")
    elif(index==2):
        print(f"{planet} is the {index}nd planet from the sun.")
    elif(index==3):
        print(f"{planet} is the {index}rd planet from the sun.")
    else:
        print(f"{planet} is the {index}th planet from the sun.")
```

```
Mercury is the 1st planet from the sun.
Venus is the 2nd planet from the sun.
Earth is the 3rd planet from the sun.
Mars is the 4th planet from the sun.
Jupiter is the 5th planet from the sun.
Saturn is the 6th planet from the sun.
Uranus is the 7th planet from the sun.
Neptune is the 8th planet from the sun.
```

A dictionary in Python is a collection of key-value pairs. It's used to store data values where each key maps to a corresponding value. This can be incredibly useful for numerous operations as the value you're interested in is directly tied to a 'key'. In the cell below create a dictionary containing the same list of planets as the 'keys', with their radii as the 'values':

(2 points)

In [16]:
```
Planet_radii = {'Mercury': 2440,
                'Venus': 6052,
                'Earth': 6378,
                'Mars': 3390,
                'Jupiter': 69911,
                'Saturn': 54364,
                'Uranus': 15759,
                'Neptune': 24622}
```

Write a function that performs the same calculation as you did above for the Moon (i.e, the volume of a planet relative to the volume of the Earth expressed as a percentage). Then use a for loop to print the volume of each planet relative to the volume of the Earth:

(10 points)

In [17]:
```
def volume_percent(r_Earth, r_planet):
    ## write your function in here
    v_Earth = (r_Earth**3)*(4/3)*3.14
    v_planet = (r_planet**3)*(4/3)*3.14

    volume_perc = (v_planet/v_Earth)*100

    return volume_perc

# Create a for loop that calls your function and prints the volume of each p

for planet in Planets:
    if (planet != "Earth"):
        print(f"Planet {planet} has a volume that is {volume_percent(Planet_
```

```
Planet Mercury has a volume that is 5.599070479385001% the volume of Earth's
Planet Venus has a volume that is 85.43645369765676% the volume of Earth's
Planet Mars has a volume that is 15.015698858902002% the volume of Earth's
Planet Jupiter has a volume that is 131699.15792260223% the volume of Eart
h's
Planet Saturn has a volume that is 61927.08914609533% the volume of Earth's
Planet Uranus has a volume that is 1508.4552186476078% the volume of Earth's
Planet Neptune has a volume that is 5753.292260038912% the volume of Earth's
```

Exercise 2 - Sea-level rise and climate change

Climate change is real, it's happening, and we're not doing enough to stop it!

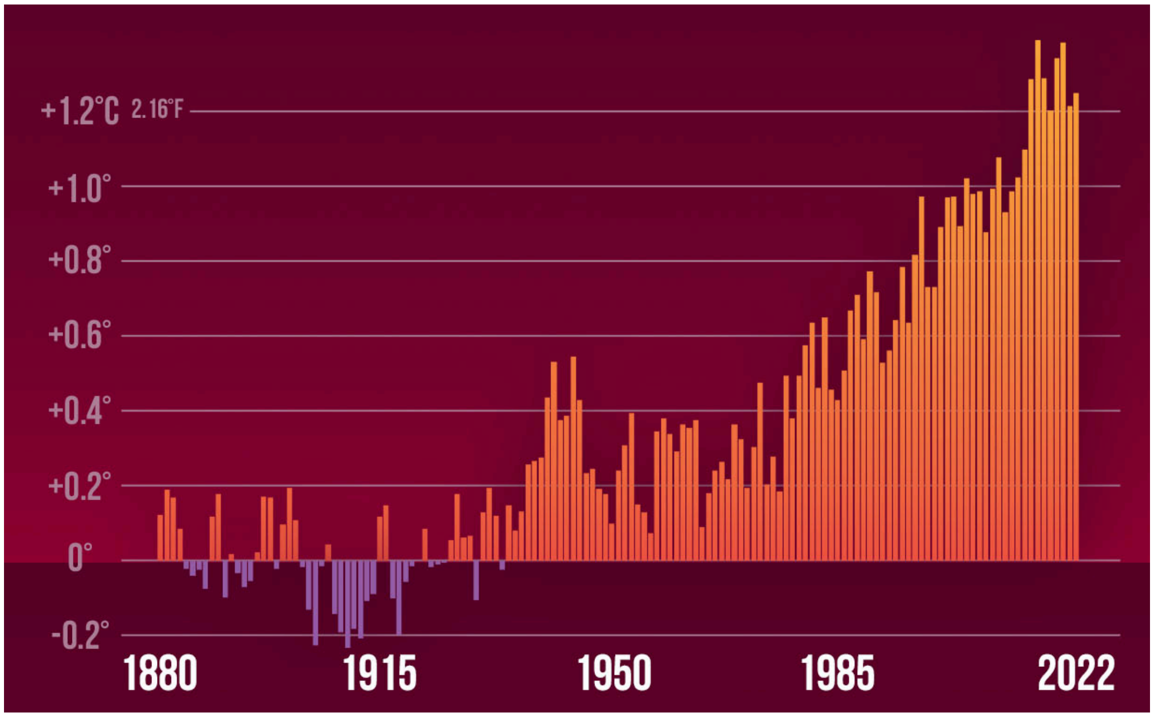

Image from https://climateknowledgeportal.worldbank.org/overview

The climate has, of course, changed in the geological past, but not at the rate we're seeing at the modern day! For the last ~34 Million Years there has always been some ice on the continent of Antarctica on the south pole. Melting of ice sheets on Antarctica (and Greenland) is causing sea-levels to rise, putting many island nations at risk of permanently loosing their land, homes, and way of life (https://only.one/read/sinking-islands-rising-costs). Sea levels are predicted to rise by ~1m by 2100, but what sea level rise might we expect if ALL the ice on Antarctica suddenly melted (this would require global average temperatures to rise > 10^{o}C)?

In the cell below, use Python to calculate the amount of sea level rise expected in this scenario (complete melting of Antarctica). At the end of the code/calculation print a message stating whether or not McCone Hall would be submerged in this scenario!

Rules

- Assume pi is 3.14
- You are NOT allowed to google anything!!! I want to see your problem solving skills not the ability to google an answer, I'm interested in whether the answer is 1m, 10m, 50m, or 100m.
- Assume a radius for Antarctica of 2000 km
- Assume an ice sheet thickness on Antarctica of 2 km
- You shouldn't need any other information!

(25 points)

```
In [18]: antarctica_radius = 2000*1000 #m
         antarctica_surface_area = (antarctica_radius**2)*3.14
         ice_thickness = 2*1000 #m
         ice_volume = antarctica_surface_area*ice_thickness
         ice_density = 0.9
         antarctica_water_volume = ice_volume*ice_density #m

         earth_surface_area = 4*3.14*((r_Earth*1000)**2) #m
         ocean_surface_area = earth_surface_area*0.7

         estimated_sea_level_rise = antarctica_water_volume/ocean_surface_area

         mccone_hall_elevation = 96 #m

         print(f"Based on some very rough estimations, Earth's sea level would rise b
```

```
Based on some very rough estimations, Earth's sea level would rise by 63.212
85931611525 meters if Antarctica melted instantly. Sitting at an elevation o
f 96 meters, McCone Hall would not get submerged.
```

```
In [ ]:
```