Python Fundamentals

CFI

1 - Python Concepts

CFI

Python Concepts

In this chapter, we will look at **foundational Python concepts** that will allow us to **generate more complex code**.

Variables & Data Types

Operators & Functions

Data Structures

Conditional Statements & For Loops

By the end of this chapter, we will be ready to start using Python to **analyze our data** and **generate insights**.

Variables & Data Types

In this section, we will be looking at **variables**, **data types**, and **how to convert & combine data types**.

Variables

Assigning: **x** = **7**

Printing: **print**()

Reassigning

Data Types

Data Type	Values	Example
Integer	Whole numbers	**1**, **2**, **3**
Float	Decimal numbers	**1.4**, **2.78**, **3.0091**
Boolean	True or False	**True**, **False**
String	Text	**'a'**, **'Hello'**, **'3ab'**

Convert & Combine Data Types

int() - converts input to integer

float() - converts input to float

string() - converts input to string

Variables & Data Types are a foundational piece for writing code in Python.

Data Structures

In this section, we will see how data structures can be used to **group** our **data and variables together**.

Data Structures

Data Structure	Syntax	Mutable?	Duplicates?	Example
Lists	**[ ]**	**Yes**	**Yes**	**List_a** = ['a','b','c'] **List_b** = [1,2,3] **List_c** = [1, 'a', True, 2.56]
Tuples	**( )**	**No**	**Yes**	**tuple_a** = ('a','b','c') **tuple_b** = (1,2,3) **tuple_c** = (1, 'a', True, 2.56)
Dictionaries	**{ }**	**Yes**	**No**	**dict_a** = {'first_name': 'Frank', 'last_name': 'Park', 'age': 20}

Data Structures

In this section, we will see how data structures can be used to **group** our **data and variables together**.

Manipulating Lists

Syntax	Definition
`list.sort()`	sort in ascending order
`list.sort(reverse=True)`	sort in descending order
`list.append()`	add new items to a list
`list[1] = 'New Value'`	replace items in a list
`del list[1]`	delete items in a list

Data structures are an important concept because we use them to set the framework of larger projects and analysis.

Operators & Functions

In this section, we will examine **operators** and **functions** – which we can use to **interact with our data**.

Operators

Math Operators	Comparison Operators	Logical Operators
+, **-**, ***, **/**	**==**	**and**, **or**
** - exponent	**!=**	**!=** - not
% - modulo	**<**, **>**	**==** - is

Operators & Functions

In this section, we will examine **operators** and **functions** – which we can use to **interact with our data**.

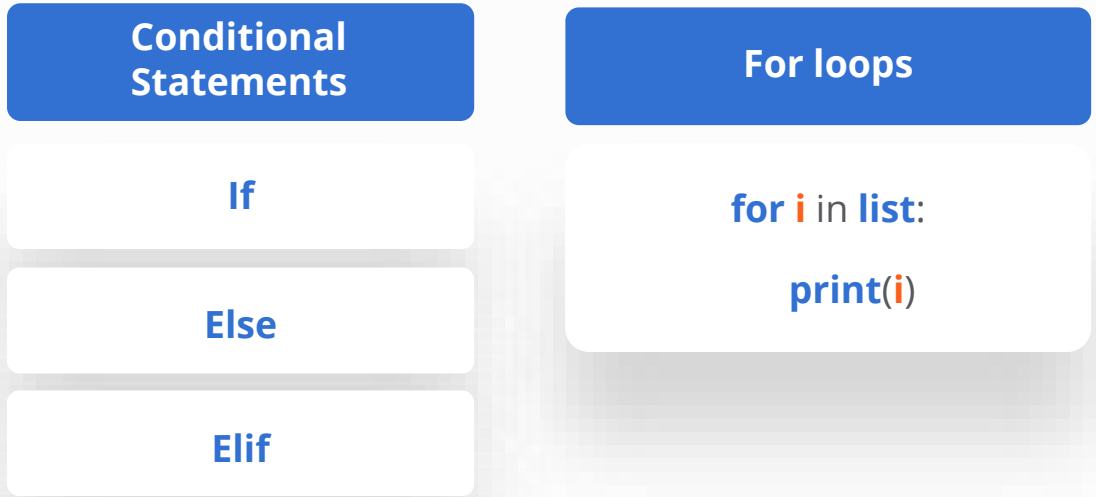
Functions

Exploratory Data Analysis	Data Transformation
type()	abs()
max()	sum()
min()	round()
len()	pow()

Operators and **functions** are crucial because they can save us time - allowing us to use logic that is already defined.

Conditional Statements & For Loops

In this section, we will start to combine some of the earlier concepts to build more complex logic into our code.



Conditional statements and **for loops** are key concepts for any application of programming in Python - and throughout the rest of the course, we'll see how important they are to coding Python for data analysis and data science.

Chapter Review - Python Concepts

In this chapter, we reviewed four foundational concepts that will allow us to generate more complex code in later chapters.

Create and define **variables**

Set the **data type** of our variables and data

Combine our data into different **data structures**

Use **operators** and **functions** to run calculations and produce output

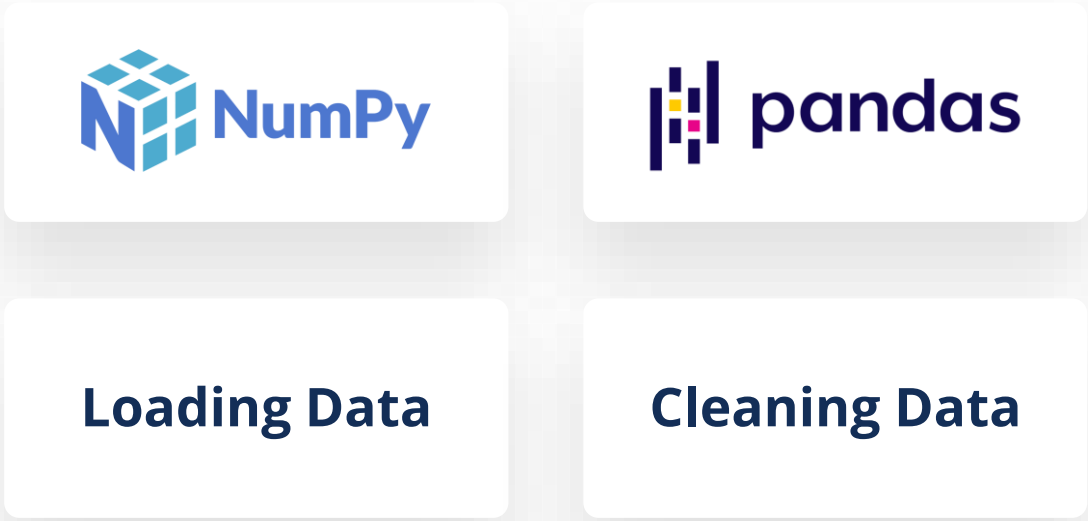
Build more complex logic by leveraging **conditional statements** and **for loops**

We are now ready to take the next step in our learning journey, using Python to analyze our data and generate insights!

2 – Loading & Cleaning Data

Loading & Cleaning Data

In this chapter, we will discuss in detail two popular Python packages - **NumPy** and **Pandas** – and how to utilize these packages to efficiently load and clean our data.



NumPy & Pandas

In this section, we will introduce the functionalities of **NumPy** and **Pandas** in more detail.

- **Arrays**
- **Operations with NumPy Arrays**

- **Series**
- **Operations with Series**
- **DataFrames**

Loading Data

In this section, we will dive deeper into **NumPy** and **Pandas** functions that are used for creating and loading data.

Generate Data

- NumPy.Arange()
- NumPy.random.rand()
- Numpy.randint()

Import Data

- Pandas_datareader.data.DataReader()
- Pandas.read_csv()

Cleaning Data

In this section, we will be focusing on one of the most important concepts in data science – **cleaning data**.

Missing Data	Duplicate Data	Incorrect/Error Data
isna() – identifies null values	**nunique**() – identifies duplicate values	**str.split**() – modifies errors
dropna() – removes null values	**drop_duplicates**() – removes duplicated values	
fillna() – replaces null values		

Cleaning data is important, as we want our data to be as free from errors as possible when inputting our data into a data science model.

Chapter Exercise – Loading & Cleaning Data

John owns a cake store that specializes in cheesecake. He wants to utilize Python to extract insights from his customer data. We can aid John by doing the following:

1. **Create a DataFrame by importing from a csv. file**
2. **Perform mathematical operations within the DataFrame**
3. **Identify null values and help him deal with it accordingly**

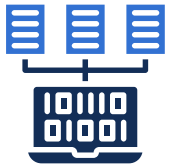
The exercise can be found in the **Chapter Exercise folder**, in Chapter 2 of the Student Files. A starting Jupyter notebook and **"Chapter Exercise Data.csv"** have been provided.

We will review a completed version of the exercise in the next lesson, but encourage you to try on your own to practice your skills!

3 – Analyzing Data

Analyzing Data

In this chapter, we will utilize the packages we learned in the previous chapter to **modify** and **analyze our data**.



Transforming Data

Statistical Analysis

This is crucial in the process of data science, as the **analysis** and **transformation of data provides immediate insights.**

Transforming Data

In this section, we will cover how to **transform data** within our **Pandas DataFrame**.

Selecting Data	Add/Remove Data	Merging Data
head() & **tail**() – preview data	Adding rows/columns with Pandas	Grouping Data for aggregation
loc() – select data using column names	Removing rows/columns with Pandas	Concatenating/merging multiple DataFrames
iloc() – select data using column indexes	Creating/resetting indicies	

These methods of transforming data are useful, especially when trying to find insights that were not discovered initially.

Statistical Analysis

In this section, we will go over the use of **statistical functions** to conduct **statistical analysis**.

Single Variable Statistics	Analysis
mean(), **median**(), **mode**()	**NumPy.percentile**() - identify outliers
var(), **std**(), **describe**()	**NumPy.where**() – conditional analysis
	corr() – generate a Correlation Matrix

Statistics are crucial in the world of data science, as it is the core of machine learning algorithms, and tells us more about our data for us to review and draw conclusions from our data.

Chapter Exercise – Analyzing Data

John wants to make decisions based on insights from his data. He wants us to help him with the following tasks:

1. **Add new information about his customers to the dataset**
2. **Filter for specific customer information to identify loyal customers**
3. **Remove unnecessary data**
4. **Find the most optimal number of cakes he sells per order**

The exercise can be found in the **Chapter Exercise folder**, in Chapter 3 of the Student Files. A starting Jupyter notebook and **"Chapter Exercise Data.csv"** have been provided.

We will review a completed version of the exercise in the next lesson, but encourage you to try on your own to practice your skills!

4 – Visualizing Data

Visualizing Data

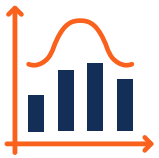
In this chapter, we will use the **Matplotlib.pyplot (plt)** and **Seaborn (sns)** packages to visualize data

Visualizing Data

matplotlib
Matplotlib.pyplot

seaborn
Seaborn

Exploratory Data Analysis


Histograms

Box Plots

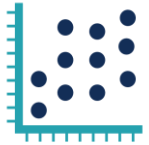
Pairplots

Heatmaps

Sharing Insights

Line Charts

Bar Plots

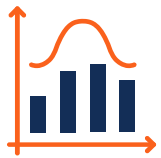
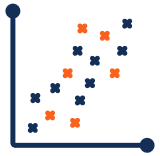

Scatter Plots

Custom Formatting

We will cover the most common visuals for **exploratory data analysis** and for **sharing insights.**

Visualizing Data for Exploratory Analysis

In this section, we will analyze a **DataFrame** with popular visuals used for **exploratory analysis**.

Column Distributions	Column Relationships	Functions
 Histograms	 Pairplots	sns.histplot()
Box Plots	Heatmaps	sns.boxplot()
		sns.pairplot()
		sns.heatmap()

We will focus on exploring **column distributions** and **relationships between columns** in our DataFrame.

Visualizing Data for Sharing Insights

In this section, we will explore powerful **formatting options** to **control the appearance of our visuals**.

Visuals

Line Charts

Bar Plots

Scatter Plots

Formatting

- Titles, Labels, & Annotations
- Size, Limits, & Orientation
- Color Palettes
- Data Style, Hue, & Size

Functions

- **plt.plot**()
- **sns.barplot**()
- **sns.scatterplot**()

We will create **line charts**, **bar plots**, and **scatter plots** to visualize trends, compare categories, and show relationships in our data

Chapter Exercise – Visualizing Data

John wants to share the insights that we have generated from the data. To do this, we can:

1. **Create a histogram of the Quantity column**
2. **Create a bar plot – aggregation Customer by number of OrderIDs**
3. **Create a scatter plot of Quantity vs Amount**

The exercise can be found in the **Chapter Exercise folder**, in Chapter 4 of the Student Files. A starting Jupyter notebook and **"Chapter Exercise Data.csv"** have been provided.

We will review a completed version of the exercise in the next lesson, but encourage you to try on your own to practice your skills!

5 – Case Study – Portfolio Optimization

Case Study – Portfolio Optimization

In this case study, we will build two $10,000 investment portfolios containing four stocks. The first portfolio will have an equal weighting between the stocks. The second portfolio will be optimized with a weighting allocation that provides the best return, adjusted for risk. To build these two portfolios, we will:

1. **Import two years of data for four stocks**
2. **Build the initial portfolio with equal weighting to each of the stocks**
3. **Analyze and visualize the equal-weighted portfolio**
4. **Generate 10,000 portfolio scenarios with random weighting to each of the stocks**
5. **Identify the optimal portfolio from the scenarios and visualize the results**