Python for Data Analytics

Module 1: Programming fundamentals for data analytics

DeepLearning.AI

Python for Data Analytics

Welcome to Python for Data Analytics

DeepLearning.AI

Python for Data Analytics

Generative AI in this course

DeepLearning.AI

In this course, you'll...

💡 **Learn how to use LLMs to:**

- Find and fix code errors
- Discover new Python features
- Customize data visualizations
- Interpret inferential statistics
- And more!

Our philosophy

💡 LLMs are:

- A complement to skills, not a replacement
- Excellent coding companions for coding and fix bugs

🧠 You'll get the most from LLMs if you know how to code

📚 In this course, you'll:

- Learn fundamentals of coding and LLM collaboration
- Build code interpretation skills
- Develop intuition about what tasks LLMs excel at and can go wrong
- Use Coursera's built-in lab chatbot to completement code writing

LLMs in this course

🗓️ Demonstrates the most up-to-date capabilities as of 2025

🌲 Evergreen principles:

- How to **think about** and **use** generative AI regardless of product

🧠 Develop a mindset of iteration and skepticism

- New models and features constantly released

Changes you should expect

- More advanced and specialized features
- Cheaper tools
- Faster tools
- Higher quality outputs overall

LLM options

- ❌ You **won't** need to purchase any additional products
- ❌ This course **does not** recommended any single tool
- ✅ You'll develop confidence experimenting and selecting tools
- ✅ You'll see several tools throughout the modules
- ✅ You'll learn core principles to work with LLMs: free and paid, now and in the future

Programming fundamentals for data analytics

Module 1 introduction

DeepLearning.AI

Module 1 outline

Library revenue

Safety inspection scores

Role of programming

- Compare Python to spreadsheets
- Use Jupyter notebook environment

Fundamentals of Python

- Data types
- Variables
- Lists
- Functions

Control flow

- Conditionals
- Loops

Modify your code using an LLM

Programming fundamentals for data analytics

Computer programming

DeepLearning.AI

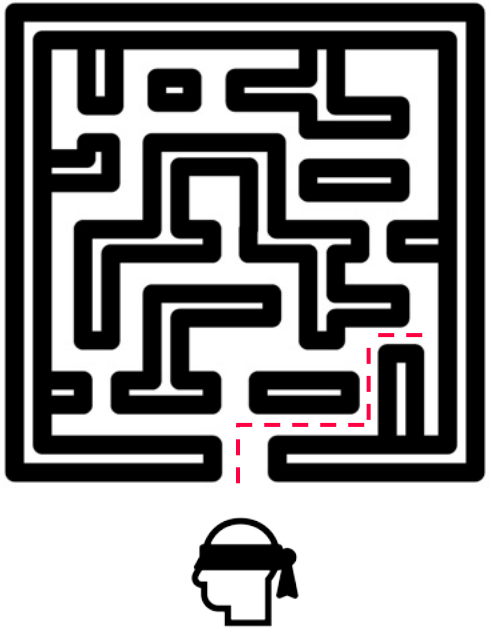
What is computer programming?

🧠 **Your role:**

- Tell computers very precisely what you want them to do

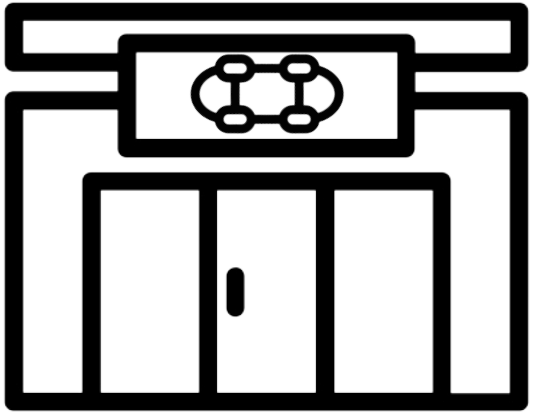
💻 **Computer's role:**

- Carry out commands to the letter
- If you don't tell them exactly how to do what you want, they might:
  - ⛔ Make a mistake
  - ⛔ Only do some of your commands



Why computer programming?

- Automates thinking work like:
  - Math
  - Data analysis
- Computers excel in:
  - Efficiency
  - Traceability
  - Repeatability

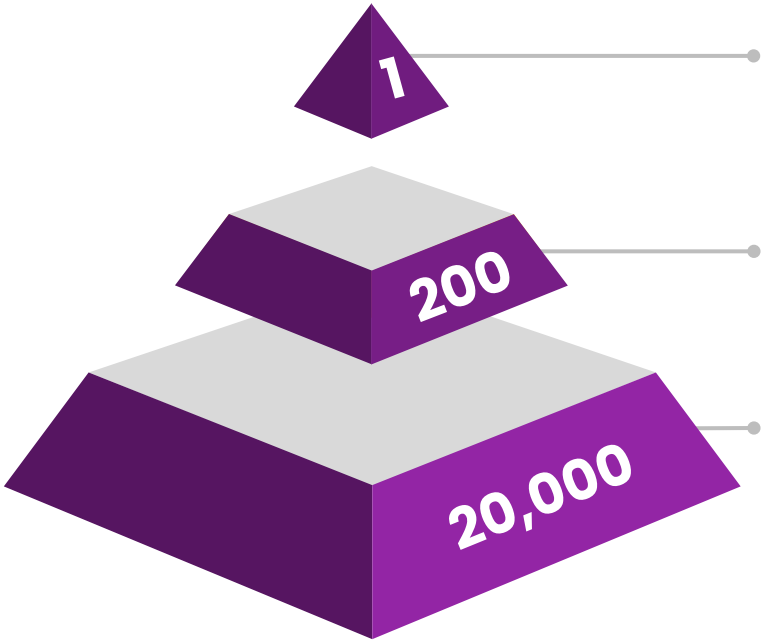


Adds up all figures at the end of the month to create sales report

Intern

Efficiency

Completing tasks with minimal time and resources



Compute average daily sales for:

A particular item

- 📱 **Intern:** 20 seconds
- 💻 **Python:** < 0.001 seconds

200 products

- 📱 **Intern:** > 1 hour
- 💻 **Python:** < 0.5 seconds

100 stores

- 📱 **Intern:** > 2 weeks
- 💻 **Python:** A few seconds 🏆

"at scale"

Traceability

The ability to track **how** tasks are completed.

Steps might be:

1. Opening dataset of sales
2. Selecting rows for each product
3. Averaging the sales
4. Saving or displaying that average

✅ When something goes wrong, you can quickly identify & fix it

Your averages look too low:

may2031.csv **may2032.csv**

Intern averages suspiciously low:

- ❌ Might not have record of what they did
- ❌ Probably have to redo the analysis

Repeatability

Doing the same task in the same way many times.

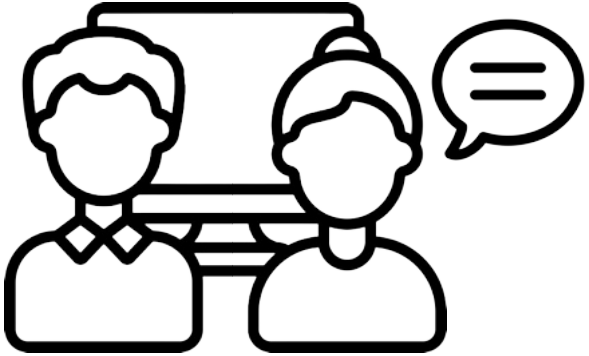
Easy to repeat the same steps over and over again

For next year's analysis:

💾 Change the filename:

may2033.csv

▶️ Rerun the code



- Intern might teach how she did the analysis before
- They may not execute it the same way

Quick note about Python

- It has a **broad user base**:
  - 👩‍💻 Individual programmers
  - 🏢 Companies
- It has a **great support** for:
  - 📊 Visualizations
  - ⚙️ Automation
  - 🌐 Web development
- It has a **great readability**
  - ⏰ Often faster to write compared with many other languages
- It has some **great technical features**

Programming fundamentals for data analytics

Navigating the Jupyter notebook environment

DeepLearning.AI

Programming fundamentals for data analytics

Input, processing, output

DeepLearning.AI

Input, processing, output

- Computer programs basically do math
- Do math on numbers that represent data, whatever the format:

Change a color image to black and white

Calculate confidence interval for daily sales

Determine whether Yelp review is positive or negative

Type of data	Numerical representation
Number	1 2 3 4 5 6 7 8 9 0
Text	104 H, 101 e, 108 l, 108 l, 111 o
Digital image	 = rgb(151,175,72)
Video	

Information → Math → New information

What code does

Input

- Numbers
- Dataset
- Folder of images
- Something else

} Program needs to expect data in that format

Processing

- Run code on the input
- Execute each step by step

Output

- Displaying something on screen
- Altering or saving a new file
- Sending results to another program or computer

Programming fundamentals for data analytics

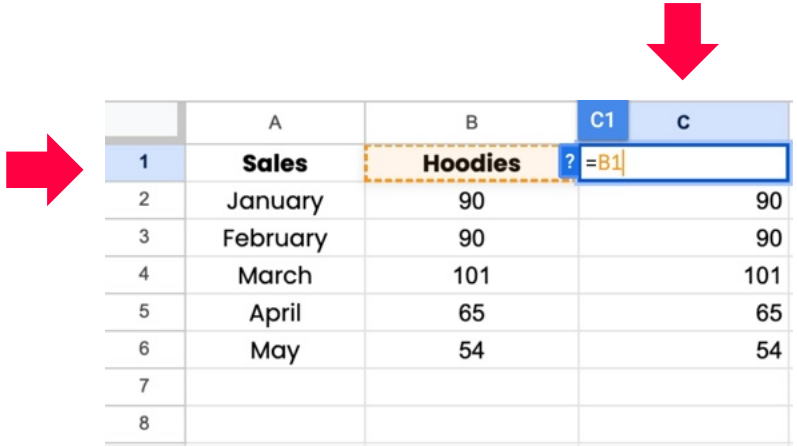
Python or a spreadsheet?

DeepLearning.AI

Interface

Spreadsheets

- Manually managing the data
- Easily format data with colors and fonts
- Store data in one format



	A	B	C1	C
1	**Sales**	**Hoodies**	? =B1	
2	January	90		90
3	February	90		90
4	March	101		101
5	April	65		65
6	May	54		54
7				
8				

Python

- Processing happens in background
- Inspect data, but won't be managing it as directly
- Format the markdown

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the CSV file
file_path = 'filmlengthsratings.csv'
df = pd.read_csv(file_path)
```

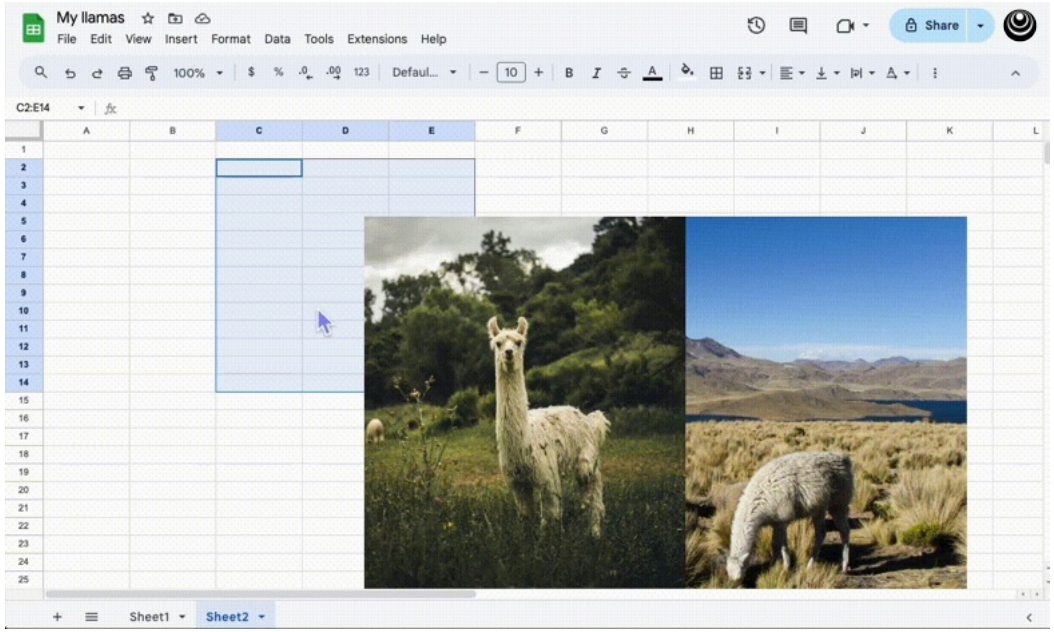
Interface

Spreadsheets

- Manually managing the data
- Easily format data with colors and fonts
- Store data in one format

Python

- Processing happens in background
- Inspect data, but won't be managing it as directly
- Format the markdown



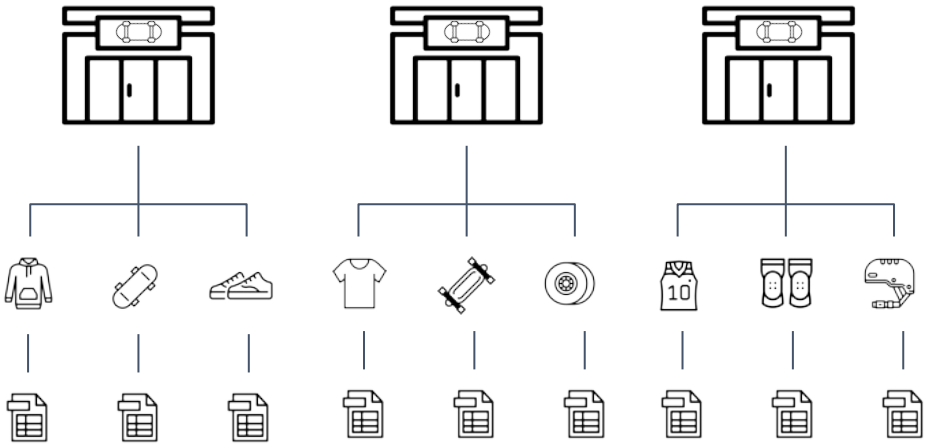
Interface

Spreadsheets

- Manually managing the data
- Easily format data with colors and fonts
- Stores data in one format

Python

- Processing happens in background
- Inspect data, but won't be managing it as directly
- Format the markdown



Complexity of analysis

Spreadsheets

- Sorting
- Filtering
- Hypothesis testing

Python

- Expands those capabilities
- Perform sophisticated statistics
- Run simulations
- Create machine learning models
- More custom analysis

Sales	Hoodie	Complimentary Product
January	900000	1282250
February	900000	1366400
March	1012500	918000
April	900000	918000
May	937500	990000

```
=T.TEST(E3:E7,F3:F7)
```

Returns the p-value

You could also manually calculate:

- Test statistic
- Degrees of freedom
- Confidence interval

```
t_stat, p_value = ttest_ind(sample1, sample2)
```

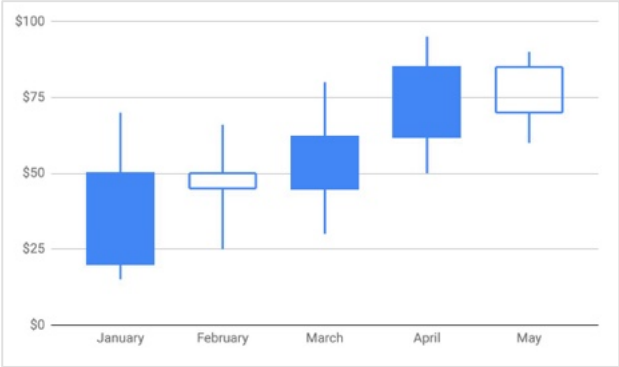
Visualization

Spreadsheets

- Sorting
- Filtering
- Hypothesis testing

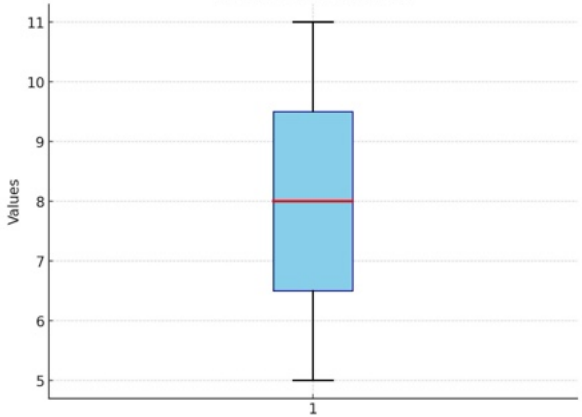
Python

- Expands those capabilities
- Perform sophisticated statistics
- Run simulations
- Create machine learning models
- A more custom analysis
- Create box plots



Source: Google Docs Editor Help

❌ Not possible to create boxplot in Google Sheets



✅ Customize every aspect of plot
✅ Apply unique color schemes
✅ Create several plots at once

Traceability & Repeatability

Spreadsheets

- You can use version history

Python ⭐

- Each step completed line by line
- Make small changes to produce many similar outputs quickly
  - Running the same analysis on similar data
  - Producing many analyses or visualizations

raw_data.csv → ? → **organized_data.csv**

1. Import data
2. Filter out rows with missing price
3. Add new column to track discount amount

15 of sales data sets formatted in same way

Dataset with 20 features

Change 1 line of code, then run same analyses

95% confidence intervals for each feature

Summary

Spreadsheets

If you're looking to:

- Quickly put together an analysis, especially at a small scale
- Be able to see the full the dataset
- You're not concerned with traceability or repeatability
- You want a lot of formatting and customization options

Python

If you're looking for:

- A highly efficient analysis at scale with larger datasets
- Complex or customized analysis or visualizations
- A traceable process that makes it easy to repeat tasks across different datasets or features

Programming fundamentals for data analytics

Types and expressions

DeepLearning.AI

Data types

Determines what kinds of operations you can perform on the data

Integers

- Whole counting numbers
- Negatives

```
-1
0
-177
```

Floats

- Real numbers including a decimal point
- Can be negative

```
0.0
1.7
-200.03
```

Strings

- Text
- Must be contained inside quotation marks

```
"oranges and apples"
"(づ ◕‿◕ )づ"
```

Integer

```
400/2
```

```
200
```

String

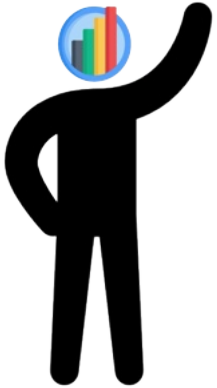
```
"400"/2
```

```
Error!
```

Data types in action

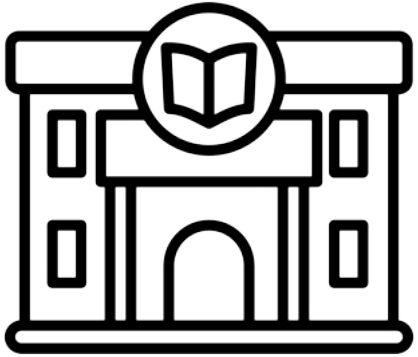
In this lesson, you'll work with:

- A real world data set of yearly library data for one library



You
Data Analyst

📍 **Connecticut, USA**



Summary

Integers

Whole counting numbers

```
-1
0
4000
```

Floats

Numbers with decimals

```
0.0
1.7
-200.03
```

Strings

Text

```
"oranges and apples"
"(づ ◕‿◕ )づ"
```

Can be used in mathematical expressions as you would in a spreadsheet

Programming fundamentals for data analytics

Printing and commenting

DeepLearning.AI

Comments

On their own line

```
# This is a comment
print("Hello, it's me")
```

```
# Total coffee cups in a year
print("Cups per year:") # Display message
print(1.5 * 365)
```

End of a line that has commands

Programming fundamentals for data analytics

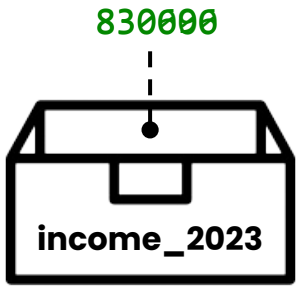
Storing information: variables

DeepLearning.AI

Variables

💾 Used to store values you expect to change or reuse

📦 A box that can store some information



Create a variable:

```
income_2023 = 830696
```

1 Declare **only once**

2 Assign

Overwriting:

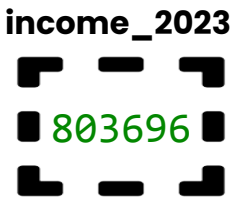
```
income_2023 = 830000
```

- Don't need the previous value
- Otherwise, create a new variable

Variables

```
income_2023 = 830696
print(35602)
```

```
35602
```



Programming fundamentals for data analytics

Debugging with variables

DeepLearning.AI

Variable naming

- Stick with:
  - Letters uppercase or lowercase
  - Numbers
  - Underscores

```
INCOME = 803696
income = 803696
income2023 = 803696
income_2023 = 803696
```

- Start your variable with a letter

- Keep in mind that variables are case sensitive

```
income_2023 = 803696
Income_2023 = 803696
```

The coder's mindset

💪 Be resilient

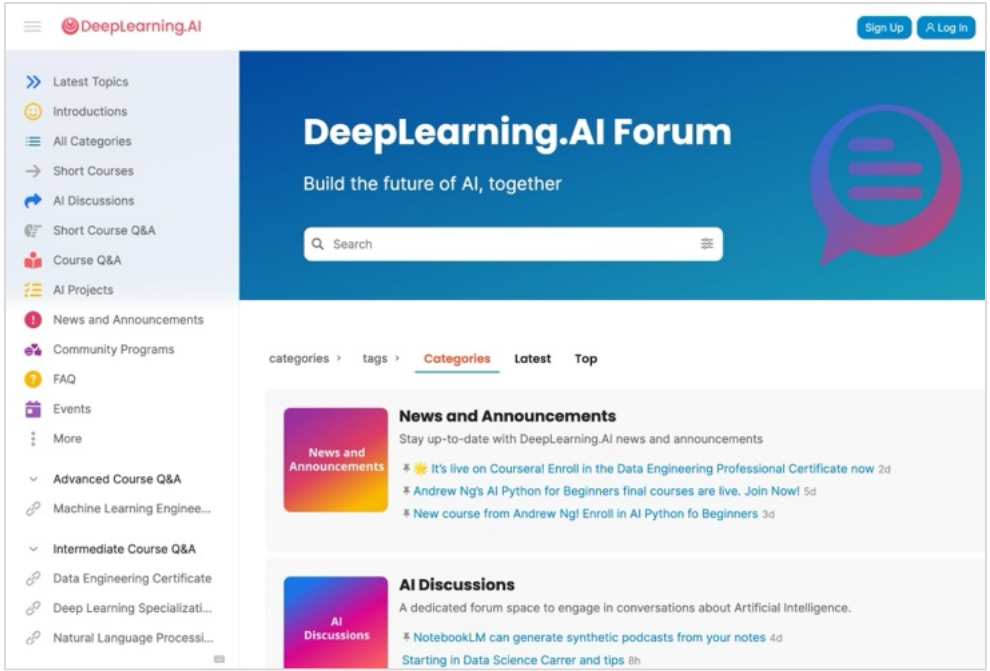
🔍 Treat errors as part of the process

Key strategies:

🔄 Try code changes

💬 Ask an LLM

🌐 Search the web



Sean Barnes

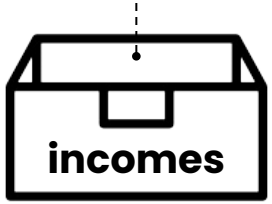
Programming fundamentals for data analytics

Creating lists

DeepLearning.AI

Lists

```
[737487, 740967, 756155, 763415, 830696]
```



- A type of **sequence**
- A collection of values that has an order
- Values inside a list are also called **items**

Declares a variable and assigns value

Brackets create a list

Plural

```
incomes = [737487, 740967, 756155, 763415, 830696]
```

Individual values separated by commas

One value

```
income_2023 = 830696
```

Programming fundamentals for data analytics

List operations

DeepLearning.AI

Recap

length of list - 1

- Lists are **zero-indexed**

0	1	2	3	4

```
incomes = [737487, 740967, 756155, 763415, 830696]
```

- To access items:

Name **Index**

```
incomes[4]
```

- To update an item:

Name for 3rd item **New value**

```
incomes[2] =  830000
```

- To append an item:

Dot allows you to perform an action

```
incomes.append(837404)
```

Programming fundamentals for data analytics

Taking action: calling functions

DeepLearning.AI

Functions

Input you provide → Processes it → Produce output

```
print("Income from 2023:")
```

Values or expressions → Processes input → Displays to the screen

```
incomes = [737487, 740967, 756155]
sum(incomes)
```

A list → Adds everything → Total

```
incomes.append(837404)
```

List & item to append → Add item to list → New, longer list

Functions

Inputs to a function: **arguments**

`sum()`

```
sum(incomes)
```

Name of the function

Inputs

`.append()`

```
incomes.append(837404)
```

List variable name

Name of the function

Item to append

Programming fundamentals for data analytics

State

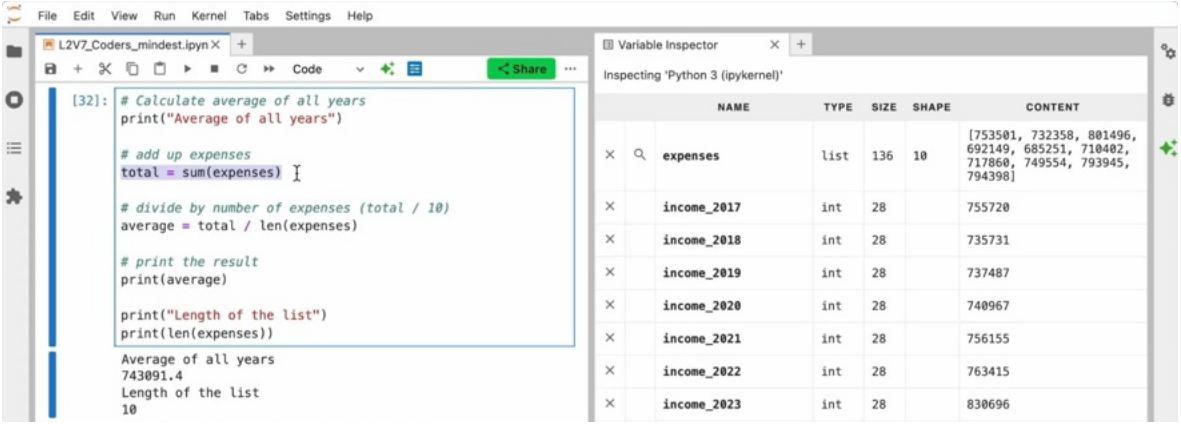
DeepLearning.AI

State

- Computer's "state" is like a snapshot of all current info
- Until you close notebook or restart kernel, variables are saved
- Helps focus on output of program, rather than steps

	A	B	C
1	**Price**	**Quantity Sold**	**Earnings**
2	1.25	4	5
3	2.75	3	25
4	3.99	1	=A4*B4

3.99



Hidden state

✅ 100% reliability: don't always need to see intermediate steps

✅ Makes calculations fasts

Problems:

⛔ Think variable has a value, but it has a different one

⛔ A dataset loaded correctly when it didn't

```
from helper_functions import get_list
incomes = get_list("Total Income")
print(incomes)
```

- Can't actually see what values are in list
- Calculate without seeing data itself:
  - `max()`
  - `min()`
  - `sum() / len()`
- Use `print()` to:
  - Make sure data is loaded
  - Form intuition about what values should look like

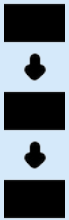
Programming fundamentals for data analytics

Control flow

DeepLearning.AI

Control flow

- Order in which individual lines of code are executed
- Determines the path computer takes through the code
- Four main types:
  - 1️⃣ Sequential
  - 2️⃣ Conditional
  - 3️⃣ Repetitive
  - ⬜ Function calls



Sequential control flow

- Default way code runs
- Lines of code will execute in order

Conditional control flow

- Executes certain lines of code only if a condition is true
- Creates branching paths
- Code goes down **one** path, not both

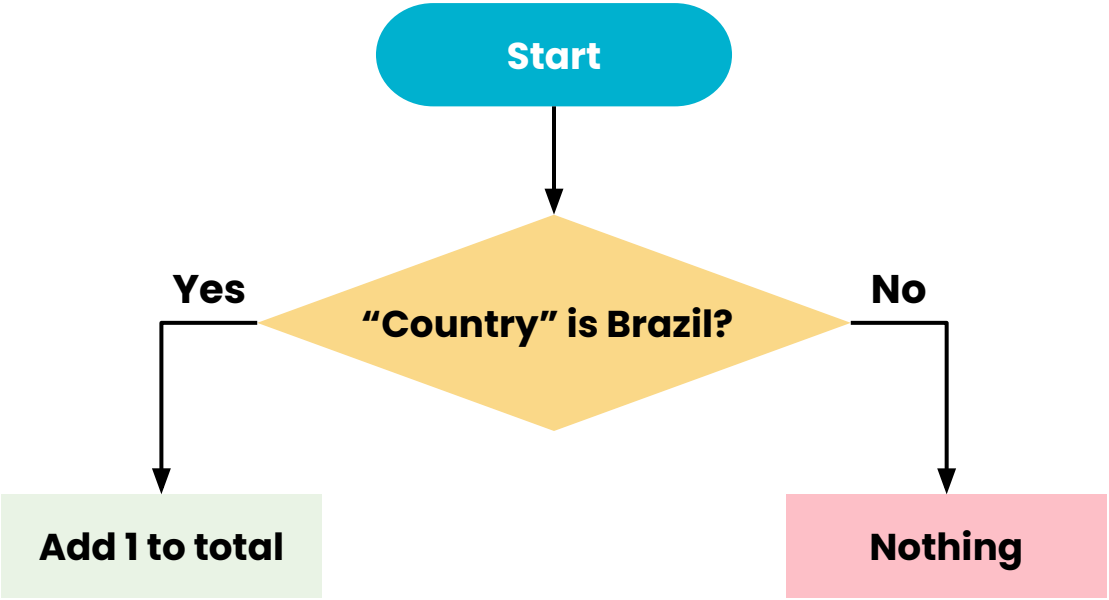
Conditional control flow

- **Dataset**: Libraries from all around the world
- **Goal**: Count number of libraries in Brazil 🇧🇷

COUNTIF

SUMIF

AVERAGEIF



There are over **75,000 libraries** in Brazil! 🤯

Repetitive control flow

- **Data**: A 100 items in a list
- **Goal**: Divide them all by 2
- Allows you to write a few lines of code that repeat an action
  - "Print all the numbers between 1 and 100"
  - "Add each item in this list to my total variable"

```
items = [item0, item1, item2,..., item99]
item[0] = item[0] / 2
item[1] = item[1] / 2
item[2] = item[2] / 2
...
item[99] = item[99] / 2
```

ARRAYFORMULA

- Repeat the operation for each cell with just one formula

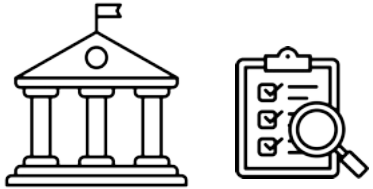
Sean Barnes

Programming fundamentals for data analytics

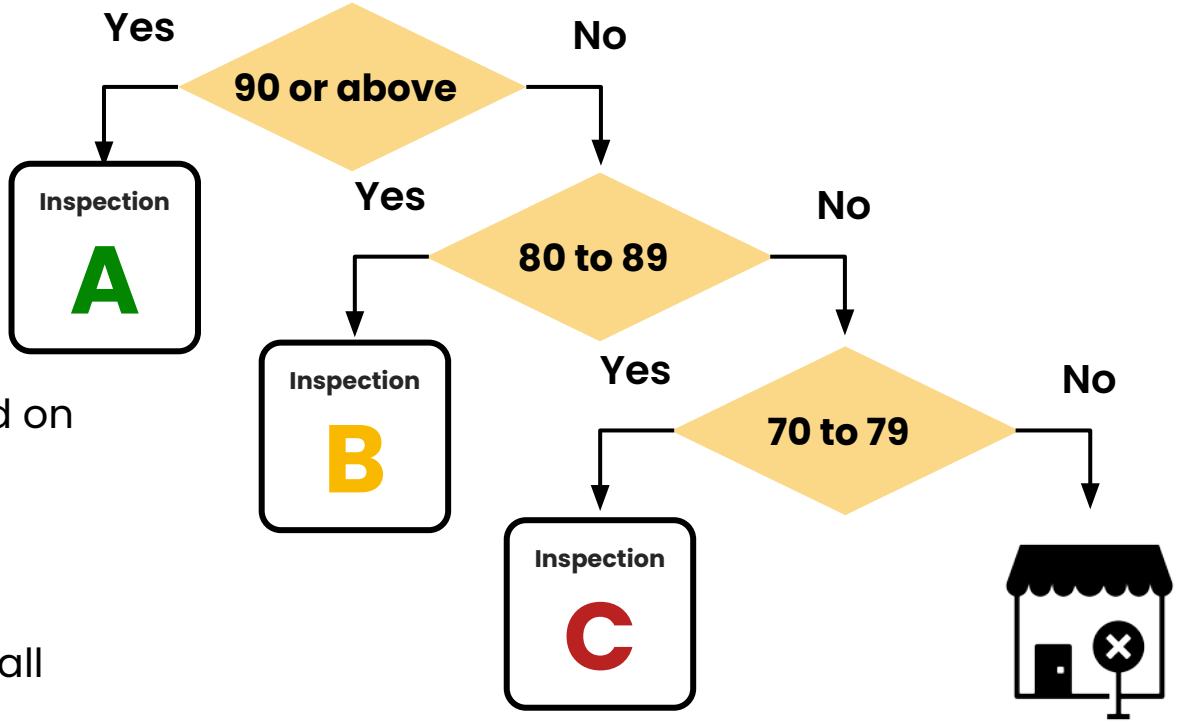
Comparison

DeepLearning.AI

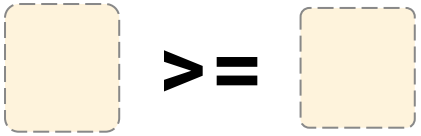
Scenario



- Give score from 0 to 100 based on its sanitation
- Raw score are analyzed to:
  - Better understand how restaurants perform overall
  - Inform consumers about health and safety violations



Comparison operators



- Answers question about values
- Two possible answers: True False **Type:** Boolean

- Look exactly the same as in spreadsheets
- Compute a Boolean value based on the inequality

<	<=	>	>=
Less than	Less than or equal to	Greater than	Greater than or equal to

- = is taken by the assignment operator
- If you're checking for equality, always write ==

==	!=
Equals	Not equals

Comparison operators

A	B	C	Failure
90-100	80-89	70-79	0-69

Beverly Falafel	Pasta Roma	The Melrose Shrimp	Modern Eats	Alfred's Coffee

```
scores = [ 96,  91,  79,  93,  86 ]
```

Question	Inequality
Did Beverly Falafel score an A?	`scores[0] >= 90`
Did Modern Eats fail their inspection?	`scores[3] < 70` **or** `scores[3] <= 69`
Did Pasta Roma score exactly 90?	`scores[1] == 90`

Knowledge check

1. Which of these expressions answers the question **“Did Beverly Falafel score a 100?”**?

```
scores[0] >= 100
```

```
scores[0] == 100
```

```
scores[0] > 99
```

```
scores[0] < 101
```

Programming fundamentals for data analytics

Branching code: if & else

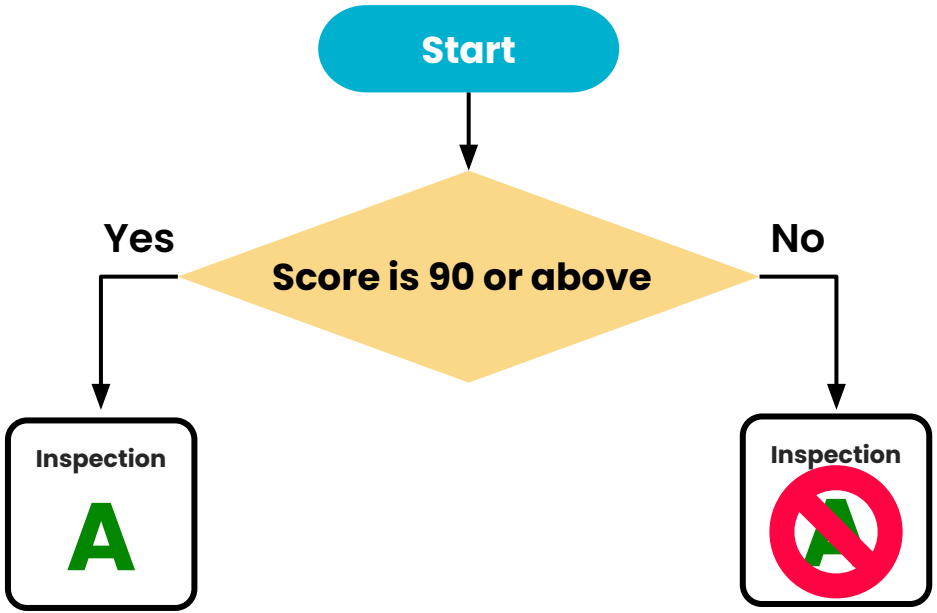
DeepLearning.AI

Recall

- **Task**: Assign a grade based on the restaurant's score

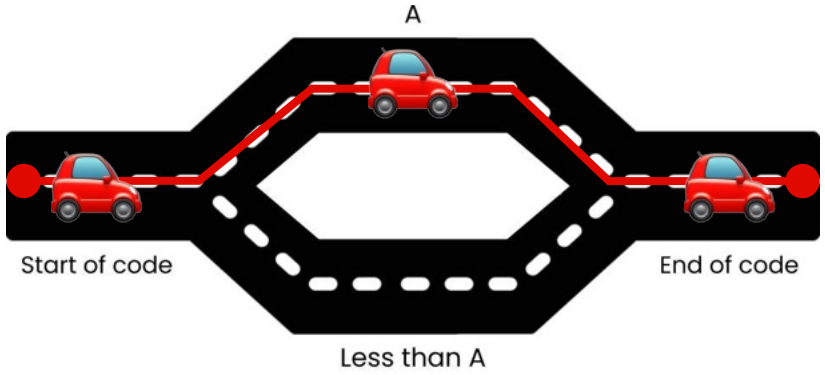
```
scores[0] >= 90
```

- To take a particular action based on decision, use an **if statement**:
  - Print whether grade is A
  - Or not an A



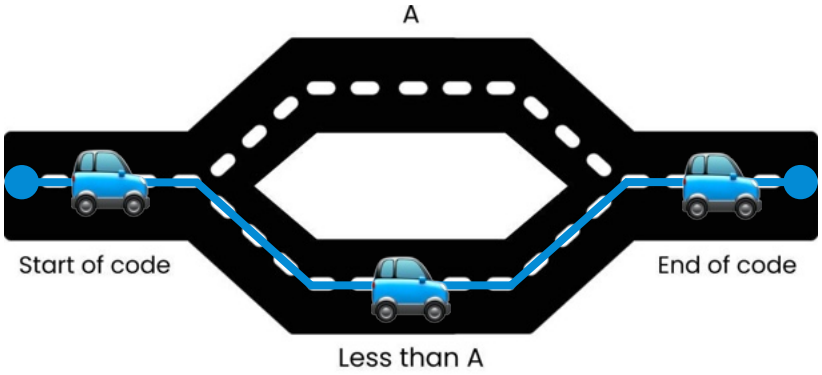
Score: 100

```
print("Start of code")  # 1
if score >= 90:
  print("A") # 2
else:
  print("Less than A") # 3
print("End of code")   # 4
```



Score: 73

```
print("Start of code")  # 1
if score >= 90:
  print("A") # 2
else:
  print("Less than A") # 3
print("End of code")   # 4
```



Programming fundamentals for data analytics

Repeating actions: for loops

DeepLearning.AI

1. Creates **new variable** of type `list`

2. Creates a **loop**
   - `for` keyword - Indicates to repeat block of code for a certain number of times
   - Creates a new variable `score`
     - Changes each time the loop runs
     - Takes on each values in `scores` in turn
   - Run same lines of code for each item in list
   - Once the loop reaches last item:
     - Loop stops
     - Moves onto next line of code

```
scores = [96, 91, 79, 93, 86]

print("Starting loop")

for score in scores:
    if score >= a:
        print("A")
    else:
        print("Less than A")

print("Done with the loop!")
```

for loop

Getting Started with Python

Indentation

DeepLearning.AI

Indentation

Tells Python which lines belong with which control structure

```
if score >= A:
    print("You got an A!")
```

```
for score in scores:
    print(score)
```

- Colon indicates Python should expect a **block of lines**
  - A group of lines that **belong** with that if statement or loop

Outside

Inside

Outside

```
scores = [96, 91, 79, 93, 86]

print("Starting loop")

for score in scores:

    if score >= a:

        print("A")

    else:

        print("Less than A")

print("Done with the loop!")
```

Recap: Indentation

- Conditionals and loops needs an indented block of code
- Python will throw an error if you don't include it
- Indentation sets aside the code you want to execute when:
  - ✅ Condition is true
  - 🔄 Inside the loop

```
if score >= A:
    print("You got an A!")
```

```
for score in scores:
    print(score)
```

- Can cause silent errors:
  - 💬 Try debugging with LLM
  - 🖨️ Print statements

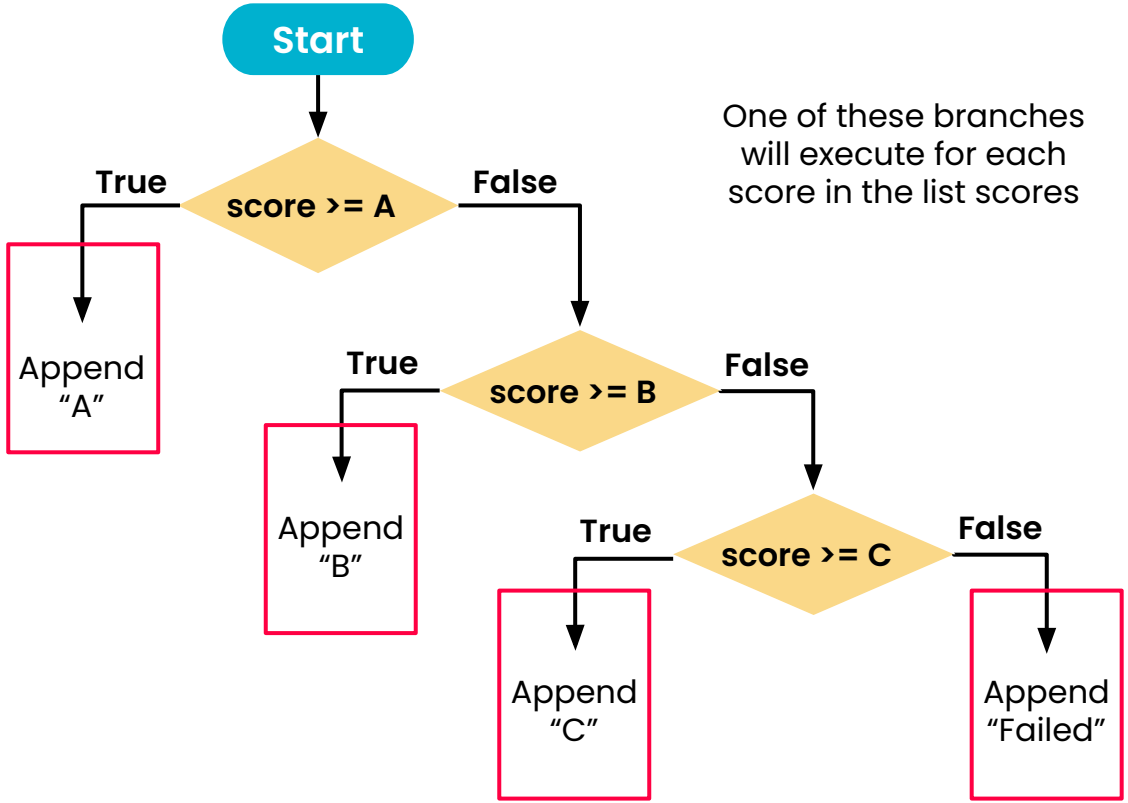
Programming fundamentals for data analytics

Branching code: elif

DeepLearning.AI

elif statement

```
grades = []
for score in scores:
    if score >= A:
        grades.append("A")
    elif score >= B:
        grades.append("B")
    elif score >= C:
        grades.append("C")
    else:
        grades.append("Failed")
```



One of these branches will execute for each score in the list scores

elif statement

- You can create as many branches as you need
- **Always** have one if statement
- **Optionally** an else statement
- Add elifs in the middle
- Each condition implies the previous one was `False`.
    - Avoid checking both high and low boundaries

```
if score >= A:
    print("You got an A!")
elif score >= 80:
    print("You got a B.")
elif score >= 70:
    print("You got a C.")
elif score >= 60:
    print("You got a D.")
else:
    print("You got an F.")
```

Programming fundamentals for data analytics

Repeating actions: range

DeepLearning.AI

Access items in list

1. Using their index:

```
scores = [96, 91, 79, 93, 86, ...]
item[0] # 1st item
item[100]  # 101st item
```

2. Looping through single list

```
scores = [96, 91, 79, 93, 86, ...]
for score in scores:
  print(score)
```

3. Loop over **numbers** by:

- Getting a list of numbers
- Use each number to access multiple lists at once
- Access same position in multiple lists

Recap: range

1. `range()`

   Creates list of numbers, from 0 to number you specify

```
scores = [96, 91, 79, 93, 86, ...]
for i in range(len(scores)):
```

2. Looping over lists with an index

```
for i in range(len(scores)):
    print(scores[i])
    print(names[i])
```

Use `i` to access each item

Access corresponding items with the same index

If you need to:

- ⬜ Access to a **single** list:

  Use `for score in scores`

- ⬜ More flexibility or access to **multiple** lists:

  Use index-based loops

Programming fundamentals for data analytics

Execution order

DeepLearning.AI

```python
A = 90
B = 80
C = 70
scores = [96, 91, 79, 93, 86, ...]

num_As = 0
num_Bs = 0
num_Cs = 0

for score in scores:
    if score >= A:
        num_As += 1
    elif score >= B:
        num_Bs += 1
    elif score >= C:
        num_Cs += 1

num_scores = len(scores)
```

❌

✅

✅

Variable	Value
num_As	3
num_Bs	1
num_Cs	1
score	86

Programming fundamentals for data analytics

Your first graded lab

DeepLearning.AI

Labs in the course

💻 **Autograded**

- We test output of your code against the correct answer
- Test your code as many times as you like before you submit

💡 **To make sure you get best grade:**

- Use provided variables names
- Replace instances of “None”
- Only need modify cells that include comment “Graded cell”

```
# GRADED CELL: Exercise 1

### START CODE HERE ###

num_observations = None

### END CODE HERE ###
```