

Practical Business Python

Lecture 2: Python Language Basics

Igor Vyshnevskyi

Woosong University

September 14, 2023

Agenda

1. Python Basics
2. Why Python for Business Analytics
3. Why not Python
4. Python Application in Business Analytics
5. Installation and Setup
6. Python Language Basics
7. In-class Assignment

1. Python Basics

A Small Introduction

Python is a programming language that was created in the 1980s.

It is named after the Monty Python comedy group.

Python is a general-purpose programming language and finds application in a broad range of domains, including web development, artificial intelligence and data science.

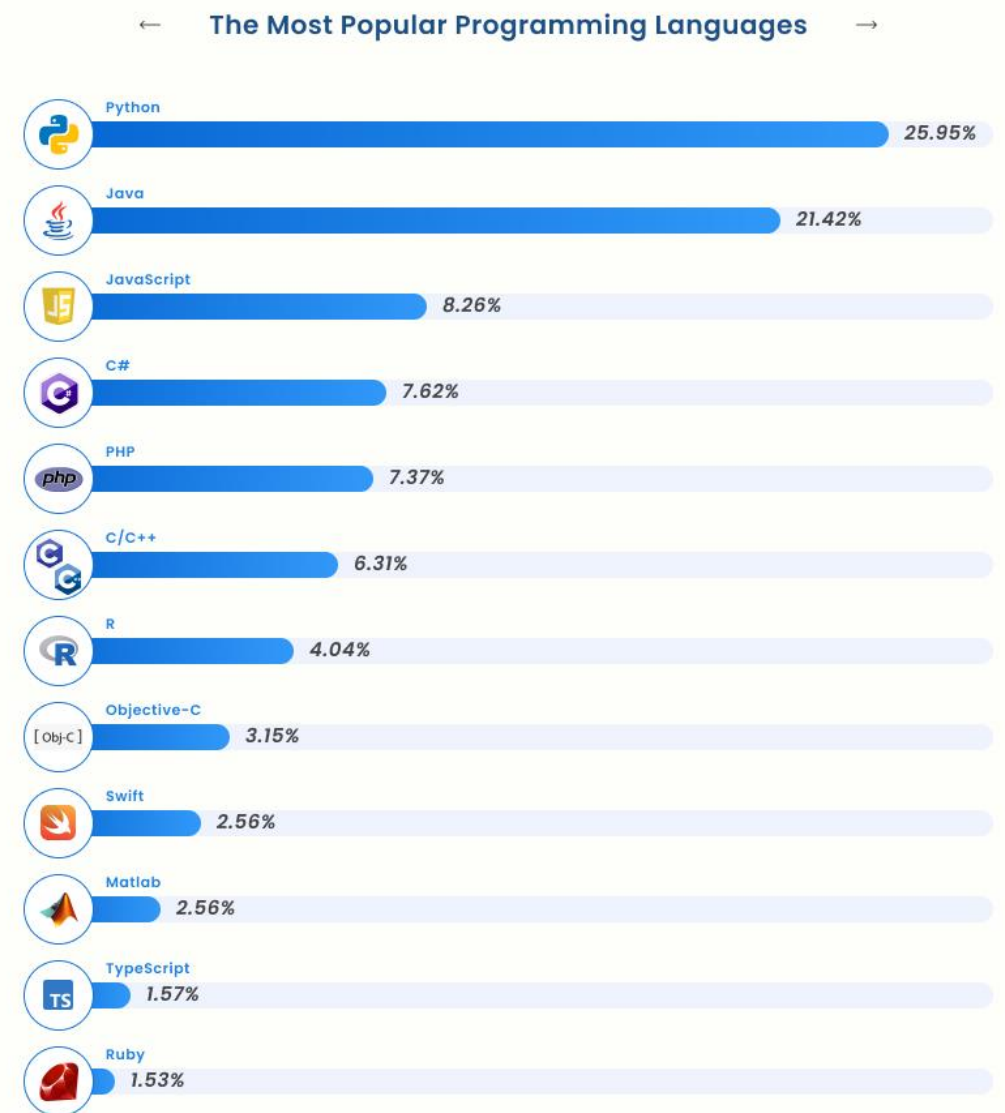
Since it is characterized as a high-level programming language, it is considered relatively easy to learn.

Python is one of the *most popular languages for business analytics* today and continues to grow at an astonishing rate.



Ranking

- The TIOBE index for July 2023 emphasizes Python's hegemony in the programming industry with a rating of 13.42%.
- Python continues to keep the top spot with a share of 27.43%, according to the PYPL index as of July 2023, produced by examining how frequently language tutorials are searched on Google.



Source: *Python for Business Analytics [A New Era of Revolution]*, V. Bhagat.;
How Python and R Dominate the Data Science Landscape? Zaveria.

Comparison with...

Python vs R for Data Science



Matt Dancho (Business Science) @mdancho84 · May 26, 2022



Which language should you pick to land a 6-figure data science job: Python or R?

Here's what I think.

14 of 130

Everything You Should
Already Know About Data Science.

Python Strengths	R Strengths
Machine Learning	Statistics
Deep Learning	Econometrics
Apps	Statistical Modeling (& Machine Learning)
APIs	Reporting (& Communication)
	Web Apps
	APIs
	Integrates Python



44



132



674





Read more here: [Python vs R for Data Science: Which Should You Learn? DataCamp](#)

Comparison with...

Python vs C# and Java

Python vs C#

		
For beginner programmers	✓	✗
For high salary	✓	✗
Speed	✗	✓
For web development	✓	✗
For game development	✗	✓

Advantages of 'Python' over 'Java'



Doesn't use semicolon, Use indentation instead of curly brackets and Dynamically typed	Use semicolon (the hide and seek champion), Use curly brackets {} , Statically typed
Simple like plain English , More focus on the tasks and Easy to read other's code	Verbose, contain more words, More concern to the code to write
Python programmers type fewer lines of code and produce less syntax errors	Must use enterprise IDE to increase coding speed
Python code run slower , but time to market is more important than time to run the app	Java code run faster
Have an extensive amount of third party libraries . Ready to be used building almost anything	Limited to Open JDK libraries or move to Enterprise Libraries

Comparison with...

Which Software Should I Choose?	Python	R	SAS	SQL
Best for:	General programming; Data analysis; Deep learning; Repeated tasks	Statistical analysis; Data analysis; Single passes of data	Statistical analysis; Data analysis	Database manipulating, updating, querying; Extracting, wrangling data
Availability	Free, open source	Free, open source	Paid (free for university edition); Closed source	Open and closed source versions available (free and paid)
Easy to learn?	Yes, especially for software engineers	Steep learning curve; Relatively easier if no prior coding experience	Yes, especially if you already know SQL	Relatively easy for basic level; Learning curve for more complex tasks
Advantages	Easy to deploy; General purpose language; Widely used by corporations	Minimal coding required for statistical models	Highly reliable, secure and stable	Very readable
Disadvantages	Requires rigorous testing	Very statistics oriented; Not a general-purpose program	Relatively expensive	Not general purpose: very specific, limited capability

2. Why Python for Business Analytics

Features Python



graphical user
interface (GUI)

Advantages of Python for business analytics (1)

- **Library Support:** it comes with many libraries. These libraries provide various functions and methods that can carry out different data analysis and processing tasks. Some of the most popular Python libraries for business analytics include NumPy, Pandas, and SciPy.
- **Speed:** it is fast. This means that the code written in Python can be executed quickly compared to other languages.
- **Easy to Learn:** It has quite a simple syntax that beginners can easily understand. Additionally, Python comes with many resources that can help you learn the language quickly.

Advantages of Python for business analytics (2)

- **Good Visualization Support:** This language comes with various libraries for data visualization. Some of the most popularly used libraries include Matplotlib and Seaborn.
- **Scalable:** This means the code written in Python can be easily scaled up or down as per the requirements. This is because Python makes use of an object-oriented programming approach. This approach helps in making the code more flexible and scalable.

3. Why not Python

Two main reasons

While Python is an excellent environment for building many kinds of analytical applications and general-purpose systems, there are a number of uses for which Python may be less suitable.

1. As Python is an interpreted programming language, in general most Python code will run substantially slower than code written in a compiled language like Java or C++.
2. Python can be a challenging language for building highly concurrent, multithreaded applications, particularly applications with many CPU-bound threads.

CPU Bound (Central processing unit, compute-bound) means the rate at which process progresses is limited by the speed of the CPU.

Sum up

Advantages and Disadvantages of Python



Advantages



Easy-to-learn and use



Improved productivity



Interpreted language



Open-source



Easily portable



Massive libraries



Easy-to-integrate



Disadvantages

Low speed



Inefficient memory consumption



Weak in mobile devices



Difficult to access database



Prone to cause runtime errors

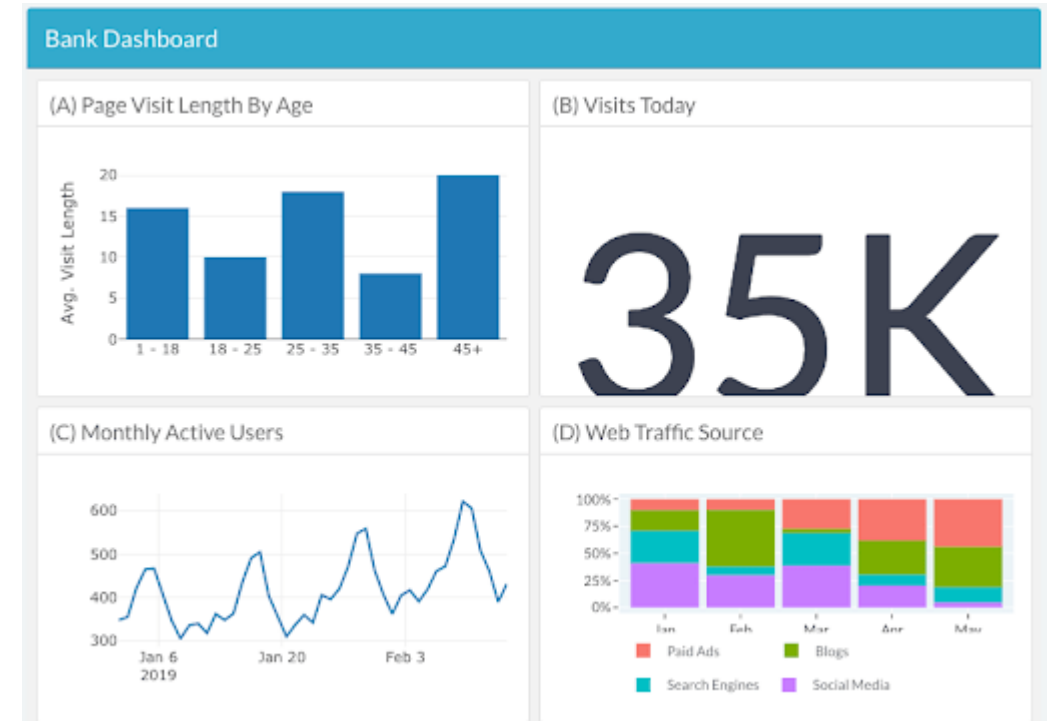


4. Python Application in Business Analytics

BI and dashboards (Descriptive analytics)

One of the primary goals for business analytics is **describing** what has happened in order to understand trends and evaluate metrics over time. This field is called descriptive analytics and is typically performed by data analysts.

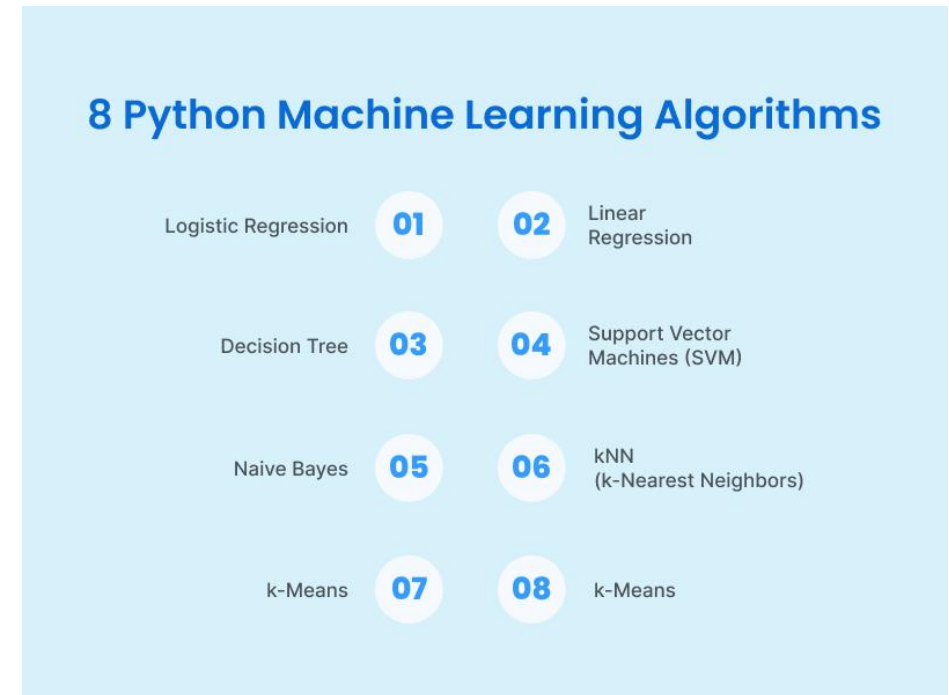
1. Data analysts often use Python to describe and categorize the data that currently exists. They engage in exploratory data analysis, which includes profiling the data, visualizing results, and creating observations to shape the next steps in the analysis.
2. Python can be used to manipulate data (using libraries such as pandas), streamline workflows, and create visualizations (using Matplotlib).



Machine learning (Predictive analytics)

Another objective of business analytics is to prepare for the future by predicting what will happen. This field is known as predictive analytics. Machine learning is the branch of predictive analytics that uses streamlined statistical algorithms to predict the future based on existing information and identify relationships and insights—think Netflix's recommendation engine.

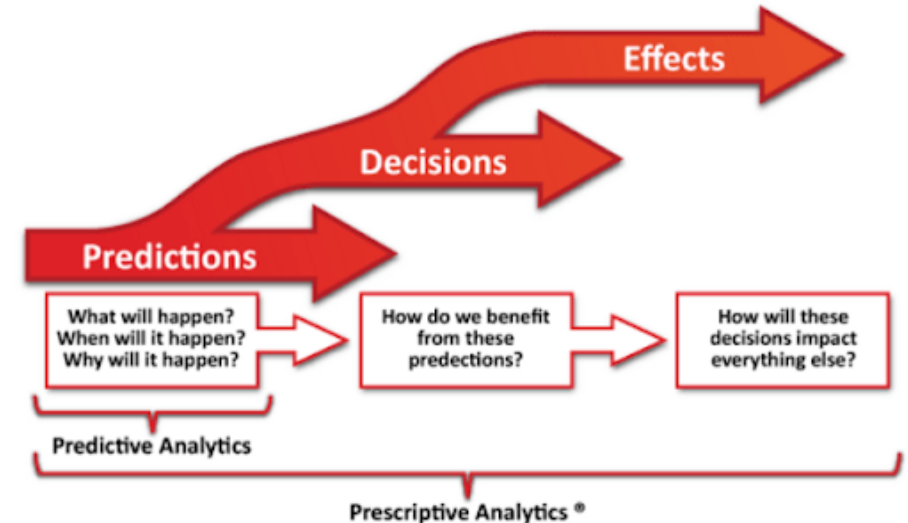
1. Python is quickly becoming the go-to language of machine learning and is used to create models for Bayesian networks, decision trees, and much more.
2. Google's **TensorFlow** is a popular Python library that many data scientists use to quickly access many supervised and unsupervised machine learning algorithms.



Decision science (Prescriptive analytics)

Prescriptive analytics, also known as decision science, is the final phase of business analytics that anticipates what, when, and why certain outcomes will happen—and determines what to do with that information. It applies data to the decision-making process.

1. Decision scientists frame their analysis of data around business problems and use many of the same techniques and tools as data scientists. Their goal is to make insights usable, so their models and visualization methods must be built to communicate those insights.
2. Python is commonly used to create prescriptive analytics tools via deep learning, which uses artificial neural networks to optimize outcomes ([Keras](#)).



And many more...

- **Data Mining**: the process of extracting valuable information from large data sets. Python can play a role in data mining by helping businesses to filter and process data, identify patterns, and more.
- **Business Process Automation**: Python can be used to automate repetitive and time-consuming tasks. This can free up employees to focus on more critical tasks and improve efficiency. Automation can also help to reduce errors and improve accuracy.

5. Installation and Setup

Installation

Since everyone uses Python for different applications, there is no single solution for setting up Python and obtaining the necessary add-on packages.

- Download & install Python (choose the correct Operation System) from <https://www.python.org/downloads/>
 - See tutorials for Windows ([link](#)) and Mac ([link](#)).

Code editor or an integrated development environment (IDE)

Python IDEs and Python code editors offer their own distinct features and user interfaces.

Which IDE to Use When? (1)

1. Based on Your Level of Knowledge

- Beginner: IDLE, Thonny would be the perfect choice for first-time programmers who are just getting into Python.
- Intermediate: For intermediate-level users, PyCharm, VS Code, Atom, and Sublime Text 3 are good options.

2. Based on Your End Goal

- Data Science: Spyder, Jupyter Notebook, PyCharm professional (Paid).
- Web Development: VS Code, PyCharm professional (Paid).
- Scripting: Atom, PyDev, Sublime Text 3, PyCharm Community (Free).

3. Based on the Hardware You Use

- Basic (Pentium, Celeron): IDLE, Atom, Sublime Text 3, Online IDEs.
- Developer (Intel core series): PyCharm, Jupyter, Spyder, VS Code, Eclipse + PyDev.

You may use any you prefer. Even Posit, Google Colab, etc.

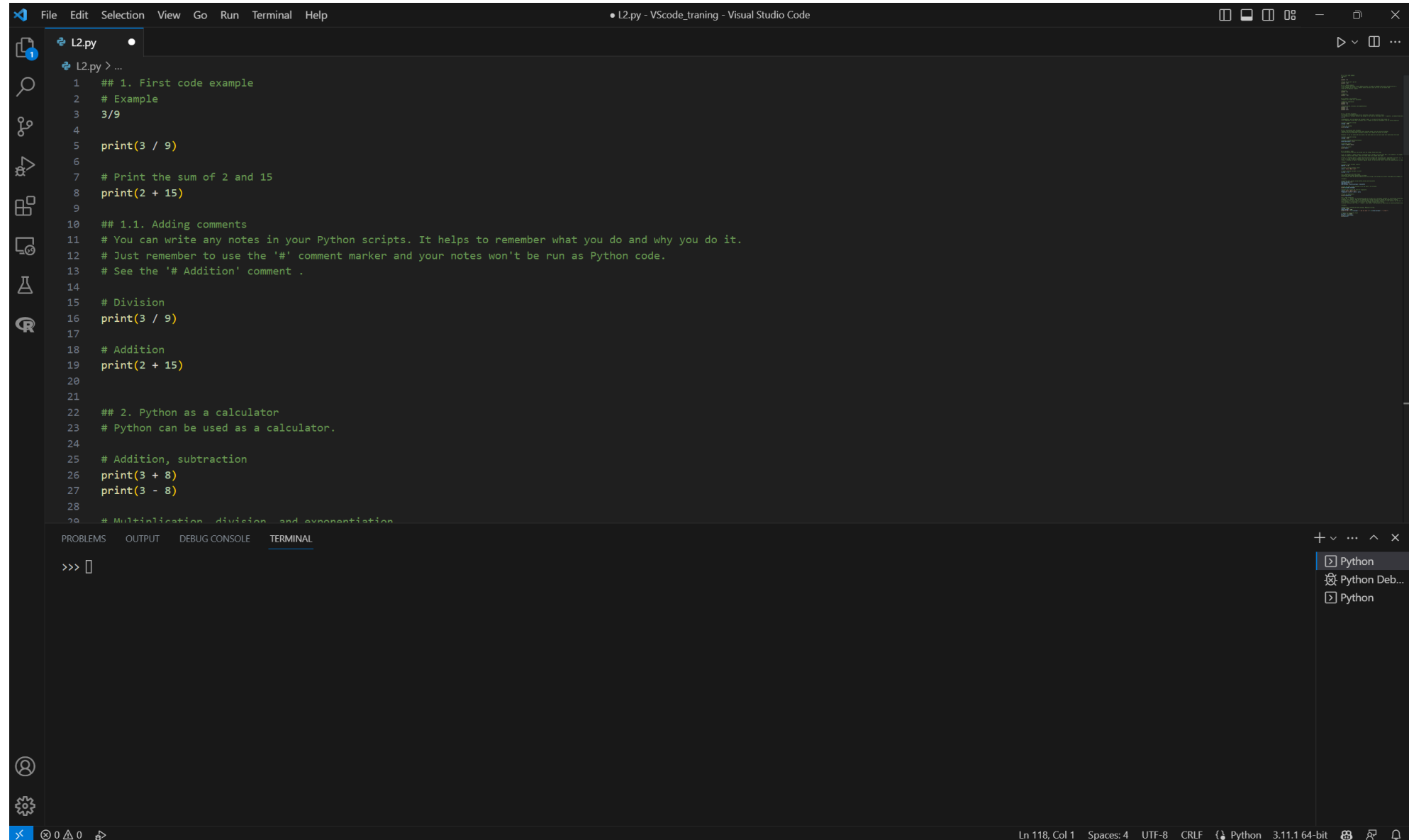
Which IDE to Use When? (2)

- For beginners, there are several Python IDEs that allow you to build a comfort level with the environment as you become steadily familiar with each feature.
- For ease of installation, IDLE is a strong choice for beginners as it's bundled with Python on your system. If you'd like to explore other options, Thonny and Wing 101 are excellent choices as they've both been designed with beginners in mind.
- But if you'd like to code in a more professional environment, you could consider PyCharm as your next IDE, or if you'd prefer portability, VS Code is a solid choice as one of the best Python code editors.
- Experienced and professional developers find themselves spoiled for choice. If you're after an IDE that offers strong performance with larger projects, then Pycharm or PyDev (Eclipse) are popular options. Similarly, VS Code is often chosen when a code editor is preferred.
- If you're part of the ever-expanding data science community then Jupyter should be at the top of your list. Equally, Spyder is one of the best Python IDEs for scientific computing, so this may be a good choice if your development aligns with its strengths.
- Then again, perhaps you're a little more 'old school' and you'd prefer the look and feel of GNU/Emacs or Sublime, both of which are very popular with Linux developers.

Free Python IDE	Python IDE for Mac	Python IDE for Windows
<ul style="list-style-type: none">• Pycharm• PyDev• Visual Studio Code• Jupyter Notebook• Spyder• Thonny	<ul style="list-style-type: none">• PyDev• Pycharm• Visual Studio Code• Jupyter Notebook• Spyder• Wing• Thonny	<ul style="list-style-type: none">• PyDev• Pycharm• Visual Studio Code• Jupyter Notebook• Spyder• Wing• Thonny

6. Python Language Basics

The Python Interface (VScode)



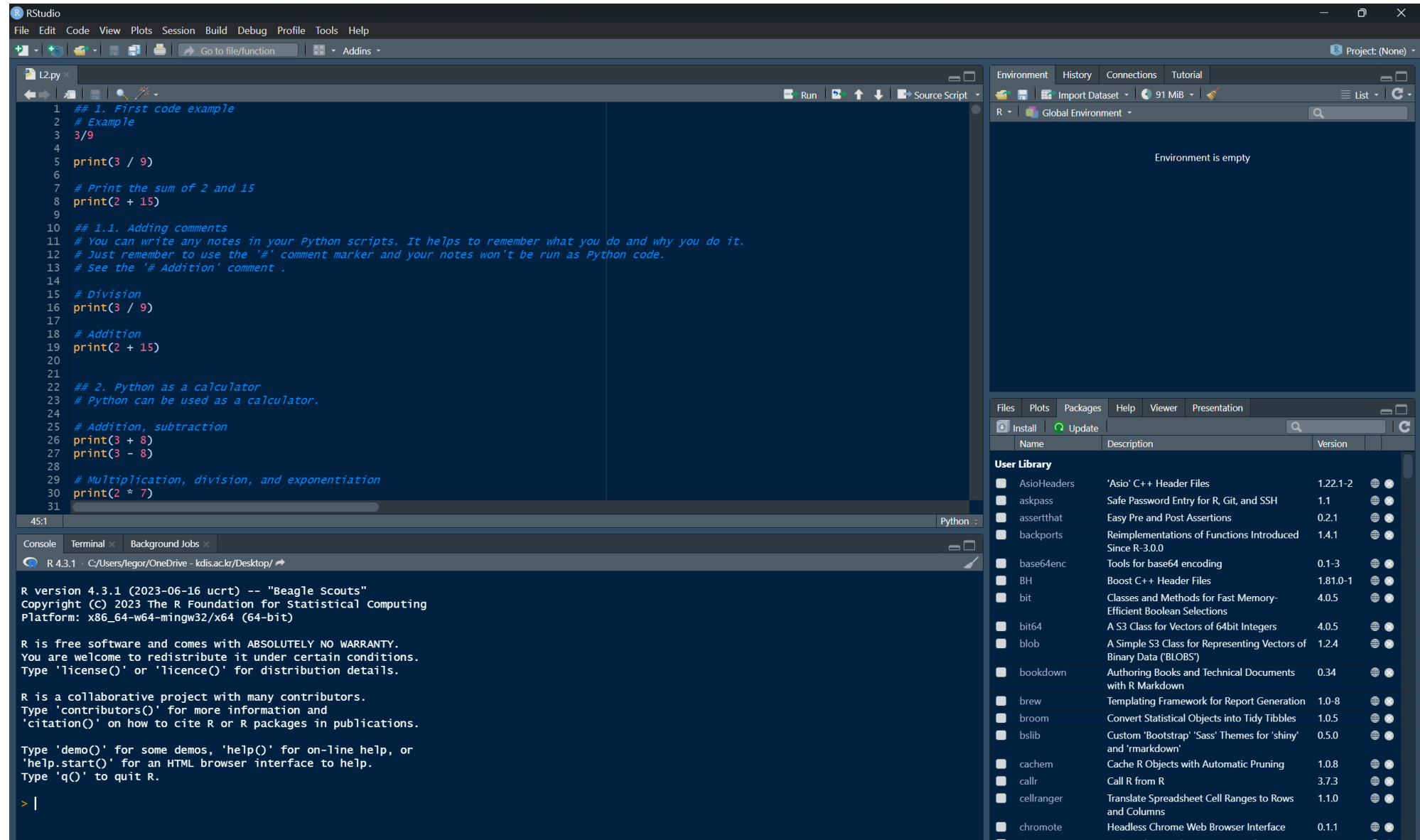
The screenshot displays the Visual Studio Code interface with a dark theme. The main editor window shows a Python file named `L2.py` with the following content:

```
1  ## 1. First code example
2  # Example
3  3/9
4
5  print(3 / 9)
6
7  # Print the sum of 2 and 15
8  print(2 + 15)
9
10 ## 1.1. Adding comments
11 # You can write any notes in your Python scripts. It helps to remember what you do and why you do it.
12 # Just remember to use the '#' comment marker and your notes won't be run as Python code.
13 # See the '# Addition' comment .
14
15 # Division
16 print(3 / 9)
17
18 # Addition
19 print(2 + 15)
20
21
22 ## 2. Python as a calculator
23 # Python can be used as a calculator.
24
25 # Addition, subtraction
26 print(3 + 8)
27 print(3 - 8)
28
29 # Multiplication, division, and exponentiation
```

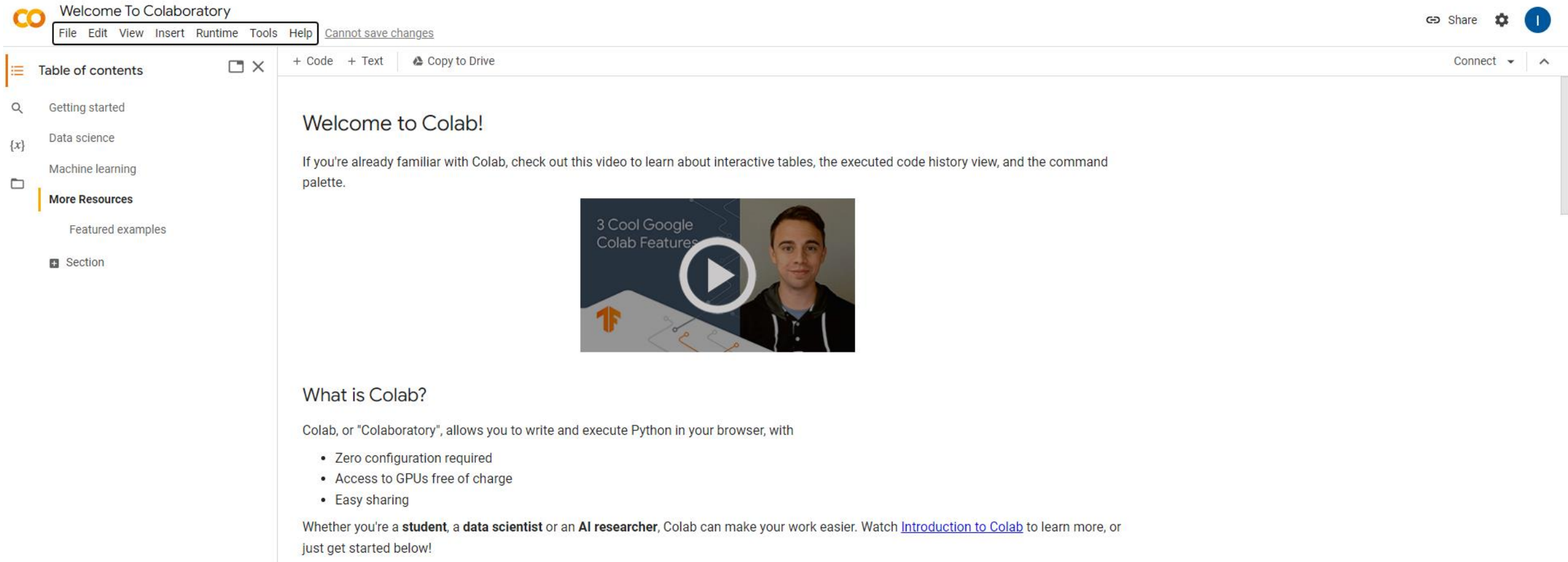
The bottom panel shows the `TERMINAL` tab, which is currently empty and displays the prompt `>>> |`. The right sidebar shows the `Python` tab selected, with a list of Python-related files and folders.

The status bar at the bottom indicates the current line and column: `Ln 118, Col 1`, and the file encoding: `UTF-8`.

The Python Interface (Posit/RStudio)



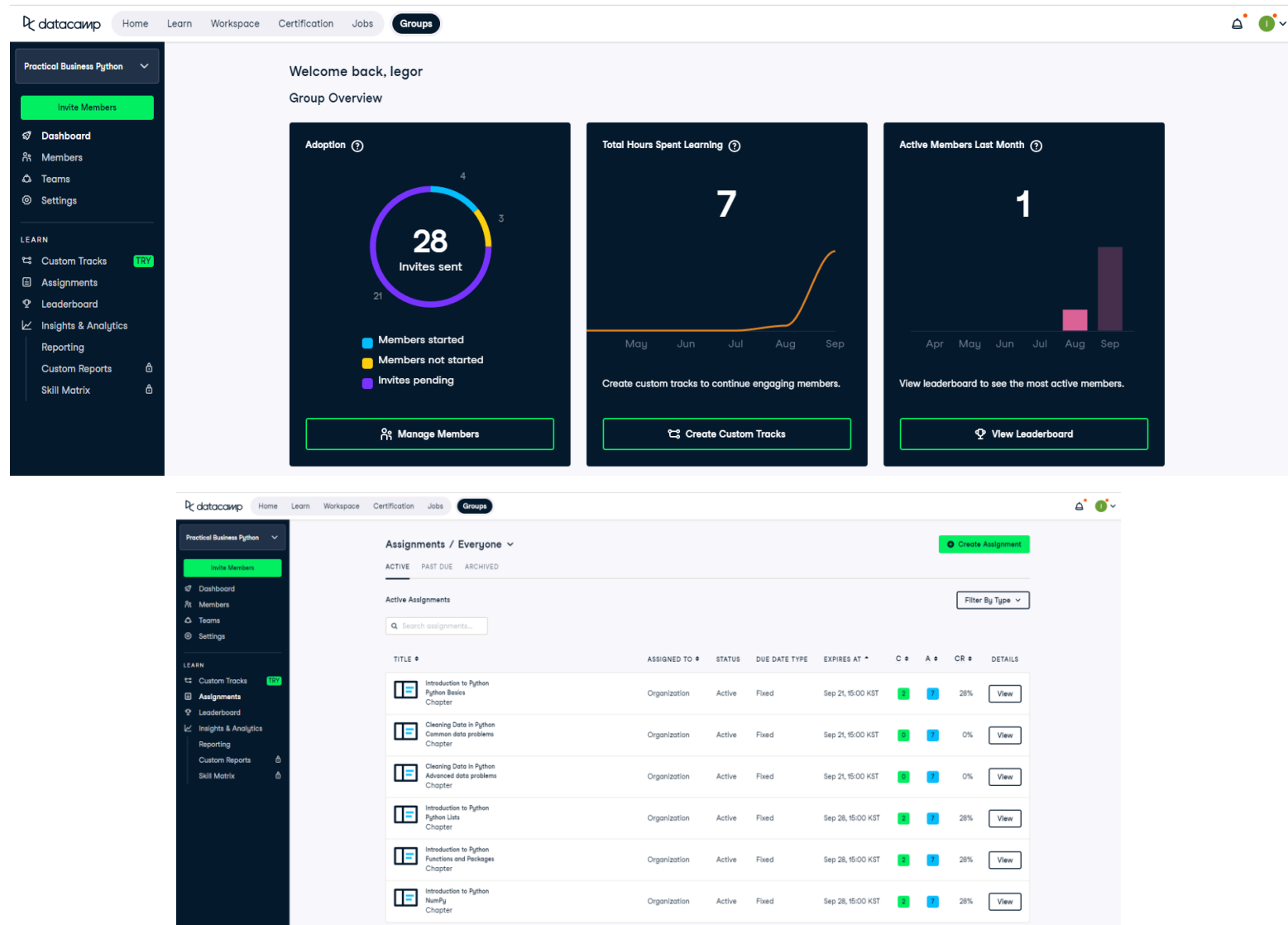
The Python Interface (Google Colab)



The screenshot displays the Google Colaboratory (Colab) web interface. At the top, a navigation bar includes the Colab logo, a "Welcome To Colaboratory" message, and a menu with options: File, Edit, View, Insert, Runtime, Tools, and Help. A status message "Cannot save changes" is visible next to the Help menu. On the right side of the top bar are links for "Share", a settings gear icon, and a user profile icon. Below the top bar, a sidebar on the left contains a "Table of contents" section with links to "Getting started", "Data science", and "Machine learning". Under "More Resources", there are links for "Featured examples" and a "Section" button. The main content area is titled "Welcome to Colab!" and contains a paragraph: "If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette." Below this text is a video player showing a thumbnail with the text "3 Cool Google Colab Features" and a play button. Further down, the section "What is Colab?" is followed by a paragraph: "Colab, or 'Colaboratory', allows you to write and execute Python in your browser, with" and a bulleted list of features: "Zero configuration required", "Access to GPUs free of charge", and "Easy sharing". At the bottom of the main content area, a paragraph states: "Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](\"#\") to learn more, or just get started below!" data-bbox="10 191 987 813"/>

<https://colab.research.google.com/notebooks/intro.ipynb>

Data Camp Classroom



<https://app.datacamp.com/groups/practical-business-python/dashboard>

First code example

- Highlight these lines and press 'Run':

```
# Example
3/9

print(3 / 9)

# Print the sum of 2 and 15
print(2 + 15)
```

- You will get this output in the terminal:

```
>>> 3/9
0.3333333333333333
>>> print(3 / 9)
0.3333333333333333
>>> # Print the sum of 2 and 15
>>>
>>> print(2 + 15)
17
```

Adding comments

- You can write any notes in your Python scripts. It helps to remember what you do and why you do it.
- Just remember to use the '#' comment marker and your notes won't be run as Python code.
- See the comments below:

```
# Division  
print(3 / 9)  
  
# Addition  
print(2 + 15)
```

Python as a calculator

- Python can be used as a calculator.
- See the code below:

```
# Addition, subtraction
print(3 + 8)
print(3 - 8)

# Multiplication, division, and exponentiation
print(2 * 7)
print(2 / 3)
print(2 ** 3)
```

```
>>> print(3 + 8)
11
>>> print(3 - 8)
-5
>>> # Multiplication, division, and exponentiation
>>>
>>> print(2 * 7)
14
>>> print(2 / 3)
0.6666666666666666
>>> print(2 ** 3)
8
```


Variable Assignment

- In Python, a variable enables you to associate a name with a specific value.
- To establish a variable called x and assign it the value 23, you employ the '=' operator, as demonstrated below

```
x = 23
```

- Subsequently, you can employ the variable's name, x, in place of the actual value, 23.
- It's important to note that in Python, the '=' symbol is used for assignment, not for testing equality!

Variable Assignment: Example

- Create a variable savings = 1000 and print it

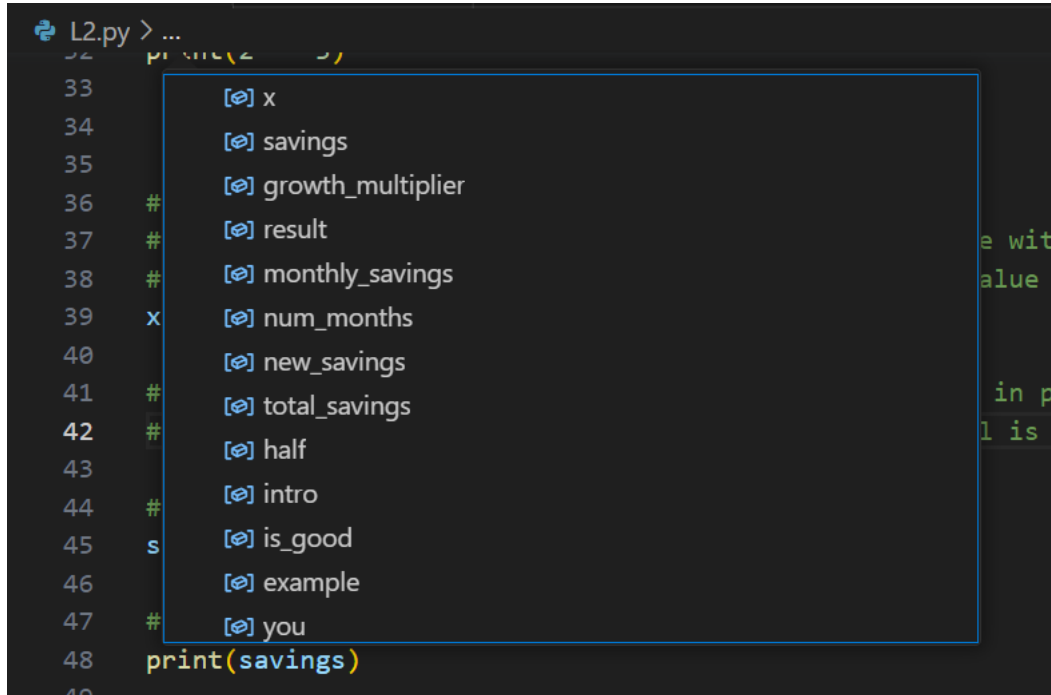
```
# Create a variable savings
savings = 1000

# Print out savings
print(savings)
```

```
>>> savings = 1000
>>> # Print out savings
>>>
>>> print(savings)
1000
```

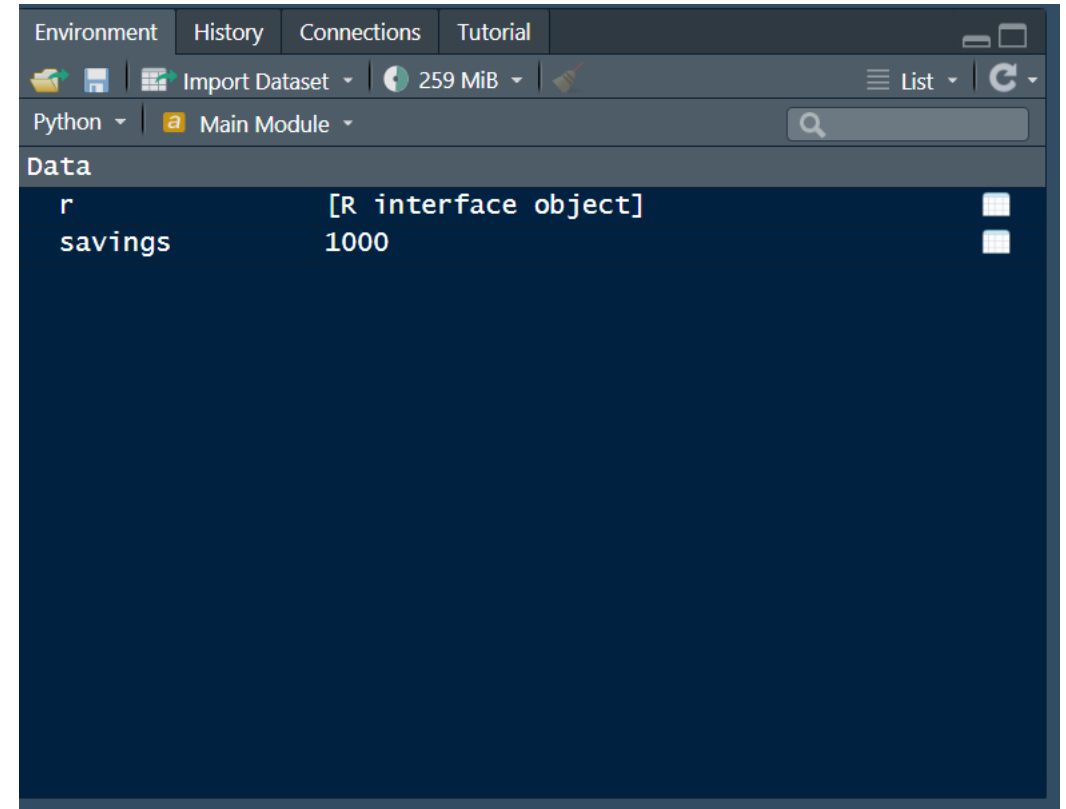
Where to find 'Savings'

- VScode: click on ... after the script name



The screenshot shows a VS Code editor window with a file named 'L2.py'. A dropdown menu is open over the variable 'savings' on line 48. The menu lists several variables: 'x', 'savings', 'growth_multiplier', 'result', 'monthly_savings', 'num_months', 'new_savings', 'total_savings', 'half', 'intro', 'is_good', 'example', and 'you'. The 'savings' variable is highlighted in the menu. The code in the background includes comments like '# e with value' and '# in p l is'.

```
33 print(x)
34
35
36 #
37 #
38 #
39 x
40
41 #
42 #
43 #
44 #
45 s
46
47 #
48 print(savings)
```



- RStudio: check the 'Environment'

Calculations with variables

- Rather than performing calculations with concrete values, you can utilize variables.
- Now that you've established a savings variable, let's begin the process of saving.

Example

- If you set aside \$10 every month, how much money will you have saved four months down the line?

```
# Create a variable savings
savings = 1000

# Create a variable growth_multiplier
growth_multiplier = 1.01

# Calculate result
result = 1000*(1.01**4)

# Print out result
print(result)
```

```
>>> savings = 1000
>>> # Create a variable growth_multiplier
>>>
>>> growth_multiplier = 1.01
>>> # Calculate result
>>>
>>> result = 1000*(1.01**4)
>>> # Print out result
>>>
>>> print(result)
1040.60401
```

Variables' types

- In the preceding exercise, we dealt with the integer data type in Python:
 - `int`, or integer: This type represents whole numbers without fractional parts. For instance, the variable "savings" with a value of 1000 is an example of an integer.
- Apart from numerical data types, there are three other widely used data types:
 - `float`, or floating point: This type represents numbers with both an integer and fractional part, separated by a decimal point. For instance, 1.1 is an example of a float.
 - `str`, or string: This type is used to represent text. You can create strings using either single or double quotes.
 - `bool`, or boolean: This type is employed to denote logical values, and it can only take on two possible values: True or False (note the capitalization!).

Example

- Let's create three new variables:

```
# Create a float variable 'quarter'
quarter = 0.25

# Create a string variable 'intro'
intro = "Hello! What's up?"

# Create a boolean variable 'is_fine'
is_fine = True
```

```
quarter = 0.25
# Create a string variable 'intro'

intro = "Hello! What's up?"
# Create a boolean variable 'is_fine'

is_fine = True
```

Operations with other types

- Different types behave differently in Python.
- For instance, when you perform addition with two strings, the outcome will differ from adding two integers or two booleans.

Example

```
# Calculate year_savings using monthly_savings and num_months
monthly_savings = 10
num_months = 24
year_savings = monthly_savings * num_months

# Print the type of year_savings to see the type of the variable
print(type(year_savings))

# Assign sum of intro and intro to tripleintro
intro = "Hello! What's up?"
tripleintro = intro + intro + intro

# Print out doubleintro
print(tripleintro)
```

```
>>> monthly_savings = 10
>>> num_months = 24
>>> year_savings = monthly_savings * num_months
>>> # Print the type of year_savings to see the type of the variable
>>>
>>> print(type(year_savings))
<class 'int'>
>>> # Assign sum of intro and intro to tripleintro
>>>
>>> intro = "Hello! What's up?"
>>> tripleintro = intro + intro + intro
>>> # Print out doubleintro
>>>
>>> print(tripleintro)
Hello! What's up?Hello! What's up?Hello! What's up?
```

Type conversion

- Using the '+' operator to concatenate/merge two strings can be extremely valuable for constructing custom messages.
- Imagine, for instance, that you've computed your savings and wish to present the results as a string.
- To achieve this, you'll need to convert the types of your variables explicitly. To be precise, you can use 'str()' to convert a value into a string.
- For instance, 'str(savings)' will convert the integer savings into a string.
- Similar functions like 'int()', 'float()', and 'bool()' are available to assist you in converting Python values into various data types as needed.

Example

```
# Definition of savings and total_savings. Merging in string.
savings = 1000
total_savings = 1430
print("I had $" + str(savings) + " and now have $" + str(total_savings) + ". Great!")

# Convert pi_string into float
pi_float = 3.1415926
pi_int = int(pi_float)
print(pi_int)
```

```
>>> savings = 1000
>>> total_savings = 1430
>>> print("I had $" + str(savings) + " and now have $" + str(total_savings) + ". Great!")
I had $1000 and now have $1430. Great!
>>> # Convert pi_string into float
>>>
>>> pi_float = 3.1415926
>>> pi_int = int(pi_float)
>>> print(pi_int)
3
```


Logical Operators

- Logical operators return boolean values of true or false and can be used to link a set of conditions.
- When combining multiple operators, we need to use parentheses to facilitate their correct evaluation. Parentheses have the highest precedence and cause the expressions inside parentheses to be evaluated first. If two operators have the same precedence, the expression is evaluated from left to right.

Example

&, and	TRUE if both Boolean expressions are TRUE
, or	TRUE if either Boolean expression is TRUE
^, xor	TRUE if either Boolean expression is TRUE
in	TRUE if the operand is equal to one of a list of expressions
~, not	Reverses the value of any other Boolean operator

7. In-class Assignment

Q & A

Thank you!