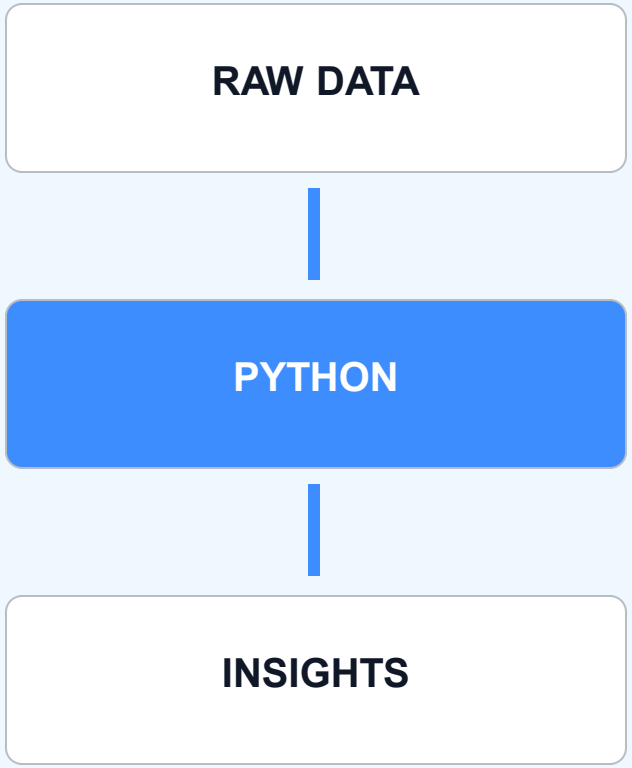
DATA SCIENCE

Python Foundations for Data Science

Language Fundamentals & Files Handling



A readable, practical path from data to decisions

UNC GREENSBORO

Python fits the data-science learning curve

It lets beginners express an idea clearly, then grow into serious analysis.

1

Readable

Syntax is close to plain English.

2

Versatile

One language for analysis, automation, and visualization.

3

Ecosystem

NumPy, pandas, matplotlib, scikit-learn, and more.

4

Community

Tutorials, examples, and answers are easy to find.

Python is often the glue between data, analysis, and communication.

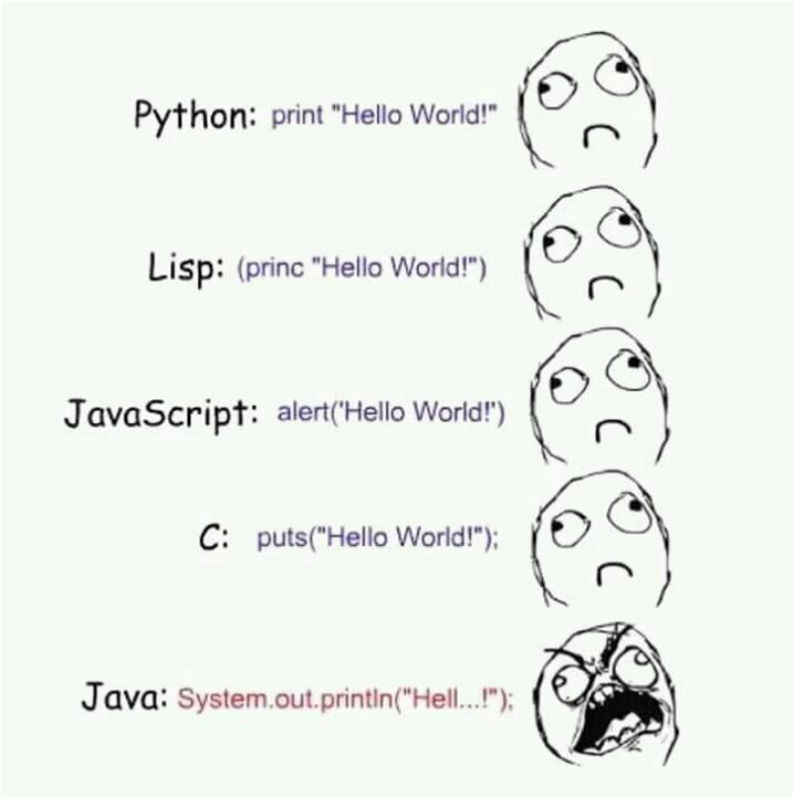
Readable code beats clever code

The Zen of Python is a useful guide when writing code for other people—and your future self.

> **Simple is better than complex.**

> **Readability counts.**

> **Errors should never pass silently.**



"Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered."

— Guido van Rossum

Choose the workspace that fits the task

All three run Python; they differ mainly in how you interact with code.

Python interpreter

Run one command at a time.

```
python
```

Jupyter Notebook

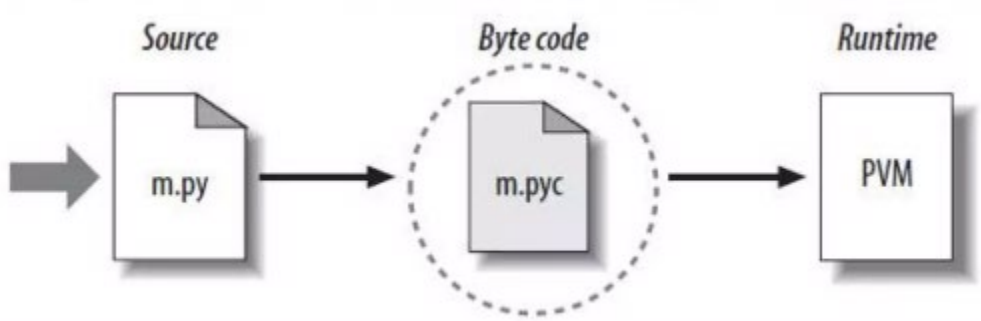
Mix code, results, and explanations.

```
In [1]:
```

VS Code

Edit files, run scripts, and debug.

```
.py
```



Python's traditional runtime execution model

Source code extension is .py
Byte code extension is .pyc

A Python program is a sequence of clear instructions

Indentation groups code blocks—there are no braces around a conditional or loop.

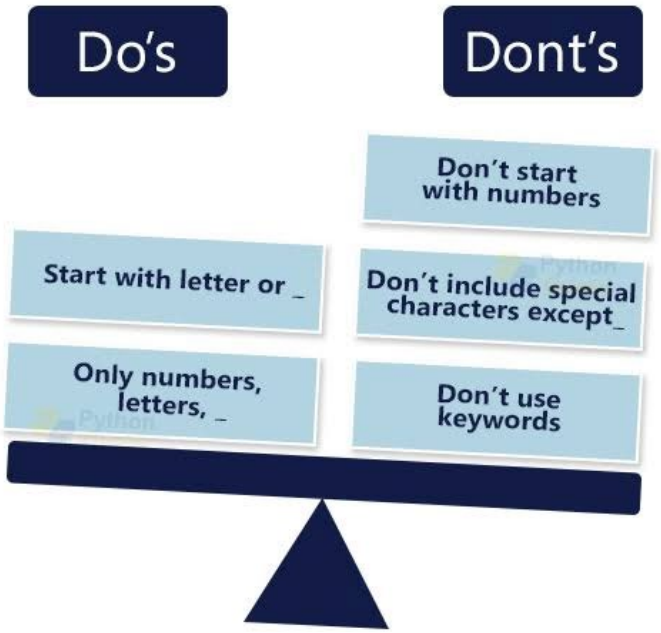
While the number of spaces can vary, all lines in the same block must be indented consistently.

```
1  num = 3
2  if num < 0:
3      print("Negative number")
4  else:
5      if num == 0:
6          print("Zero")
7      else:
8          print("Positive number")
9
10 for x in "ABC":
11     for y in "123":
12         print(x, y)
```

Python Identifiers

Rules for writing identifiers

- Identifier is the name given to entities like class, functions, variables etc.
- Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore( ).
- An identifier cannot start with a digit.
- Keywords cannot be used as identifiers.
- We cannot use special symbols like !, @, #, $, % etc.



Python Keywords

Keywords: There are 33 keywords in Python 3.3.

Keywords in Python programming language				
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

They have special meaning and cannot be used as variable names.

Python Variables

Variables:

- A variable is a location in memory used to store some data (value).
- They are given unique names to differentiate between different memory locations. The rules for writing a variable name is same as the rules for writing identifiers in Python.
- We don't need to declare a variable before using it. In Python, we simply assign a value to a variable and it will exist.
- We don't even have to declare the type of the variable. This is handled internally according to the type of value we assign to the variable.

```
student_count = 28
average_score = 84.6
course_name = "Data Science"
value_a = value_b = 10
```

Data Types in Python

Data Types in Python

- Every value in Python has a data type.
- Since everything is an object in Python
- Data types are actually classes and variables are instance (object) of these classes.
- type() function can be used to know which class a variable or a value belongs to.

Name	Data Type	Description
Integer	int	a number that can be written without fractions: 22 10 0 -300
Boolean	bool	logical value that indicates **True** or **False**
Floating-point	float	a number that has a decimal component: 3.14 2.73 10.0
String	str	sequence of characters: "hello world" "hey88340" '2018'

Data Collections in Python

String	List	Tuple	Set	Dictionary
Immutable	Mutable	Immutable	Mutable	Mutable
Ordered/Indexed	Ordered/Indexed	Ordered/Indexed	Unordered	Unordered
Allows duplicate members	Allows duplicate members	Allows duplicate members	Doesn't allow duplicate members	Doesn't allow duplicate keys
Empty string = ""	Empty list = []	Empty tuple = ()	Empty set = set()	Empty dictionary = {}
String with single element = "A"	List with single item = ["Achal"]	Tuple with single item = {"Achal"}	Set with single item = {"Achal"}	Dictionary with single item = {"Achal":1}
---	It can store any data types: str, list, set, tuple, int and dictionary	It can store any data types: str, list, set, tuple, int and dictionary	It can store int, str, tuple, but not list, set and dictionary	Inside the dictionary, **Key** can be int, str and tuple only. **Values** can be of any data types str, list, set, tuple, int and dictionary

Arithmetic Operators in Python

Python evaluates arithmetic expressions from right to left, following standard precedence.

Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

```
mean = total / count
percent = passed / total * 100
```

Comparison Operators in Python

Python comparison operators compare two values and evaluate down to a single boolean value: True or False.

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
!=	Not equal to	5!=2	True

```
x = 10
y = 5

print(x == y)  # False (10 is not equal to 5)
print(x != y)  # True  (10 is different from 5)
print(x > y)   # True  (10 is greater than 5)
print(x <= y)  # False (10 is not less than or equal to 5)
```

Assignment Operators in Python

Assignment operators in Python are used to assign values to variables.

Operator	Example	Equivalent Expression
=	$m = 10$	$m = 10$
+=	$m += 10$	$m = m + 10$
−=	$m -= 10$	$m = m - 10$
*=	$m *= 10$	$m = m * 10$
/=	$m /=$	$m = m/10$
%=	$m \% = 10$	$m = m\%10$
<<=	$a <<= b$	$a = a << b$
>>=	$a >>= b$	$a = a >> b$
&=	$a \&= b$	$a = a \& b$
^=	$a \hat{} = b$	$a = a \hat{} b$

```
x = 10          # Assigns the integer 10 to variable x
name = "Alice"  # Assigns the string "Alice" to variable name
```

Logical Operators in Python

Python comparison operators compare two values and evaluate down to a single boolean value: True or False.

logical operator	Description	Syntax
and	True if both statements are true	a and b
or	True if one of the statements is true	a or b
not	Reverse the result. If the result is true it will return false and viceversa	not a not b

```
age = 25
has_license = True

# Both conditions must be True
if age >= 18 and has_license:
    print("You are allowed to drive.")  # This will print
```

Strings Operations in Python

Text can be indexed, sliced, cleaned, split, joined, and formatted.

Input	Method	Output
'hello WORLD'	.capitalize()	Hello world
'HELLO WORLD'	.lower()	hello world
'hello world'	.upper()	HELLO WORLD
'Python'	.center(10, '*')	**Python**
'HELLO WORLD'	.count('L')	3
'HELLO WORLD'	.index('O')	4
'HELLO WORLD'	.find('OR')	7
'31/01/2022'	.replace('/', '-')	31-01-2022
'31/01/2022'	.split('/')	['31', '01', '2022']
'abc123'	.isalnum()	True
'12345'	.isnumeric()	True
'hello world'	.islower()	True
'HELLO WORLD'	.isupper()	True

```
city = "Chicago"
city.lower()        # "chicago"
city[0:3]           # "Chi"
"a,b,c".split(",")  # ["a", "b", "c"]
```

Conditional Statements in Python

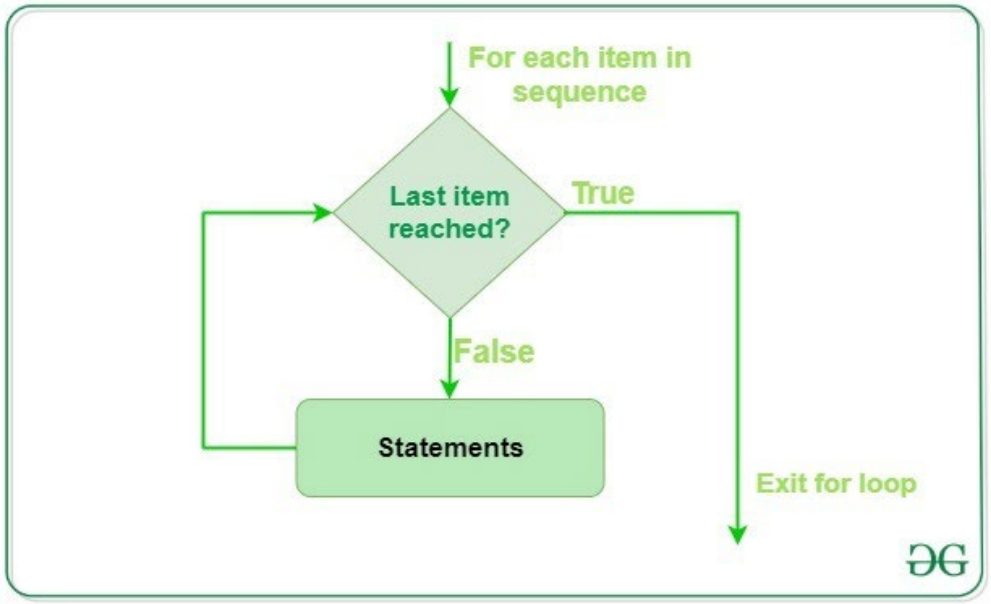
Use if / elif / else to take different actions based on a condition.

```
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "Needs review"
```

Loops in Python

for loops iterate through a collection; while loops continue until a condition changes.

```
scores = [72, 88, 91]
for score in scores:
    if score >= 80:
        print(score, "meets target")
```



Control words

break stop the loop early

continue skip to the next item

range() make a sequence of numbers

Python’s libraries add data-focused superpowers

NumPy

fast numerical arrays

pandas

tables and data cleaning

matplotlib

charts and visualizations

scikit-learn

machine-learning tools

File Handling in Python

- File is a named location on disk to store related information
- It is used to permanently store data in non-volatile memory
- In Python, a file operation take place in the following order
  - Open a file
  - Read or write (perform operation)
  - Close the file

1

Read

bring TXT or CSV data into Python

2

Inspect

check rows, columns, and missing values

3

Transform

filter, calculate, and organize

4

Write

save a clean result for someone else

TXT files Operations

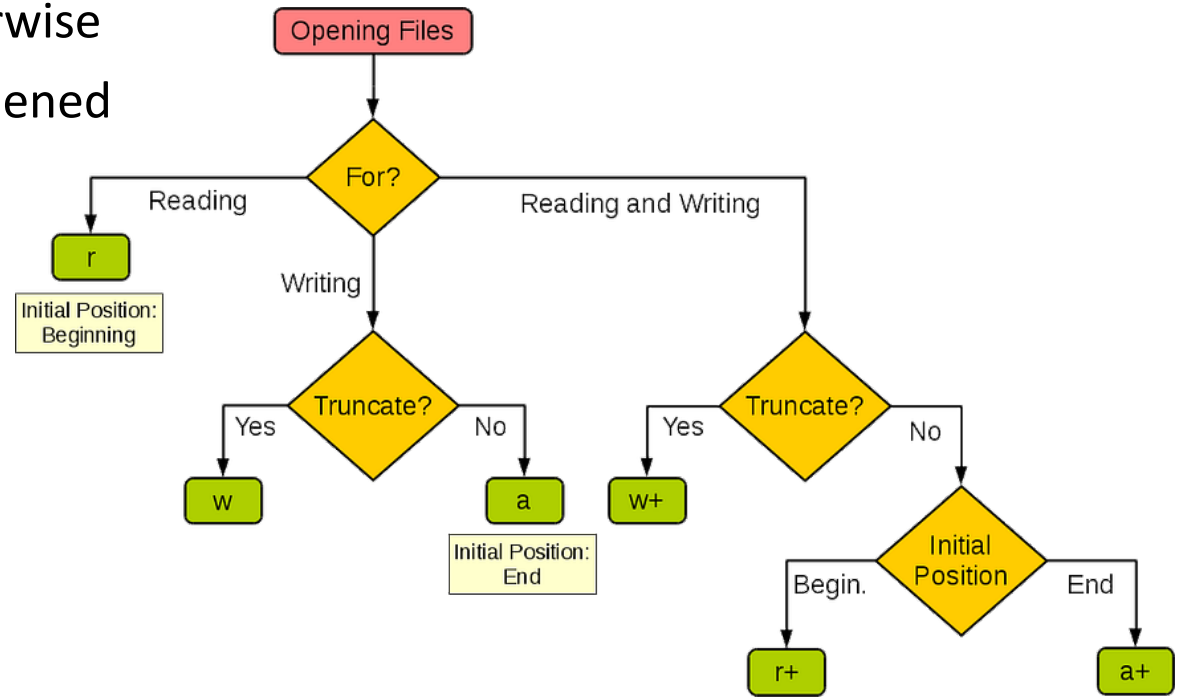
Once a file is opened ,we have one file object, which contain various info related to that file

1. file.closed-Returns true if the file is closed, False otherwise
2. file.mode-Returns access mode with which file was opened
3. file.name- Name of the file

```
with open("notes.txt", "r") as file:
    text = file.read()

print(text)
```

```
with open("summary.txt", "w") as file:
    file.write("Analysis complete!\n")
```




```
f = open('test.txt', 'w')
f.close()
```

TXT files Operations

File object includes the following methods to read data from the file.

- read(chars): reads the specified number of characters starting from the current position.
- readline(): reads the characters starting from the current reading position up to a newline character.
- readlines(): reads all lines until the end of file and returns a list object.

TXT files Operations

- The os module Python provides methods that help to perform file-processing operations, such as renaming and deleting.
- To rename an existing file, rename() method is used
- It takes two arguments, current file name and new file name

```
import os
os.rename(current file name, new file name)
```

- We can delete a file by using remove() method.

```
import os
os.remove(file name)
```

CSV files Operations

- CSV (stands for comma separated values) format is a commonly used data format used by spreadsheets.
- Each row is a record; commas separate fields such as name, major, and score.
- The csv module in Python's standard library presents classes and methods to perform read/write operations on CSV files.

```
import csv

with open("students.csv", mode="r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row["name"], row["score"])

with open("students.csv", mode="w") as file:
    writer = csv.writer(file)
    writer.writerows(data)
```

name	major	score
Ana	Stats	91
Lee	Biology	84
Mia	CS	88

Test Your Understanding…

Exercise: CSV Data Processing Practice

1. Download ‘fast_food_consumption_health_impact_dataset.csv’ from the course website.
2. Read the CSV file using Python.
3. Extract the first numeric column (‘Age’ attribute) and convert it into a list of integers.
4. Compute the maximum, minimum, and average values of the Age list.
5. Extract the second categorical column (‘Gender’ attribute) into a list.
6. Count how many times each gender category appears.
7. Write all the results into a .txt document.

Do not use external libraries such as pandas.
Use Python basics: csv, list, dict, for loops, and if statements.