Carsh course

for FDL

Théophile Gentilhomme

October 24, 2025

Imports

- You import modules to access functions/classes they provide.
- Common stack for DL: math, random, numpy, torch, pandas, matplotlib.pyplot.

Python Code | Start Over | Run Code

```
1  import math, random
2  import numpy as np
3  import pandas as pd
4  import matplotlib.pyplot as plt
5
6  print("Modules imported!")
```


Downloading package: micropip

Imports (example)

Python Code | Start Over | Run Code

```python
# Use the math module
print("sqrt(16) =", math.sqrt(16))

# Use numpy
arr = np.array([1, 2, 3])
print("mean:", arr.mean())

# Use torch
t = torch.tensor([1., 2., 3.])
print("torch sum:", t.sum().item())
```

Exercise: Imports

Python Code | Start Over | Run Code

```
1  # TODO: Import the statistics module as stats
2  # TODO: Create a list vals = [1, 2, 2, 3, 4]
3  # TODO: Print the mean and median using stats.mean and stats.median
```

Functions

- Functions group reusable logic.
- Signature, arguments, return value

Python Code | Start Over | Run Code

```
1 def add(a, b):
2     return a + b
3
4 print(add(2, 3))
```

Functions (example)

Python Code | Start Over | Run Code

```
1 def relu(x):
2     return x if x > 0 else 0
3
4 print([relu(v) for v in [-2, 0, 3]])
```

Exercise: Functions

Python Code | Start Over | Run Code

```
1  # TODO: Write a function `square(x)` that returns x**2
2  # TODO: Test on x = -3, 0, 5
```

Objects: attributes & methods

- Objects carry data (attributes) and behavior (methods).

Python Code | Start Over | Run Code

```
1  text = "Deep Learning"   # This is an object
2  print(text.upper())      # method
3  print(len(text))         # function
```

Exercise: Objects

Python Code | Start Over | Run Code

```
1  # List s are objects too
2  # TODO: Create a list of 3 numbers
3  # TODO: Use a method to append a new number
4  # TODO: Use len() to print the list length
```

Class (concept)

- A class defines a blueprint for objects.
- Special method `__init__` initializes attributes.
- Methods implement behavior.

Python Code | Start Over | Run Code

```
1  class Animal:
2      def __init__(self, name):
3          self.name = name
4          self.alive = True
5
6      def speak(self):
7          if self.alive:
8              print(f"{self.name} makes a sound.")
9          else:
10             print(f"{self.name} is not with us anymore...")
11
```

main

Inheritance

- Specialized classes can inherit from a base class and override methods.

Python Code | Start Over | Run Code

```python
class Dog(Animal):
    def speak(self):
        print(f"{self.name} barks!")

dog = Dog("Rex") # Create an object
dog.speak() # call a method
# But keeps prent methods
print(dog.is_alive)
```

Exercise: Class & Inheritance

Python Code | Start Over | Run Code

```
1  # TODO: Create a class `Cat` that inherits from Animal
2  # TODO: Override speak() to print "<name> meows!"
```

Arrays (concept)

- numpy.ndarray stores numeric data efficiently.
- Vectorized operations operate on whole arrays at once (fast, concise).

Python Code | Start Over | Run Code

```
A = np.array([[1, 2],
              [3, 4]], dtype=float)

B = np.array([[10, 20],
              [30, 40]], dtype=float)

print("A:\n", A)
print("B:\n", B)
print("A + B:\n", A + B)
print("A * B (elementwise):\n", A * B)
```

Shape and size (concept)

Python Code | Start Over | Run Code

```
1  x = np.random.randn(2, 3, 4)
2  print("Shape:", x.shape)
3  print("Number of elements:", x.size)
```

Batch (first dimension)

- In DL, the first dimension often represents batch size (number of samples).

Python Code | Start Over | Run Code

```
# 4 samples, each with 3 features
batch = np.random.randn(4, 3)
print("Shape:", batch.shape)      # (4, 3)
print("First sample:", batch[0]) # shape (3,)
```

Exercise: Batch

Python Code | Start Over | Run Code

```
1  # TODO: Create an array of shape (5, 2) with random values (e.g., np.random.randr
2  # TODO: Print its shape and first sample
```

Reshape

- Reshape changes the view of the data without copying when possible.

Python Code | Start Over | Run Code

```
1  r = np.arange(6)               # [0..5]
2  print("Original:", r, r.shape)
3
4  R = r.reshape(2, 3)
5  print("Reshaped (2x3):\n", R, R.shape)
```

Pandas (overview + example)

- Tabular data structure (DataFrame), fast I/O, data cleaning, joins, groupby.

Python Code | Start Over | Run Code

```
data = {
    "Name": ["Alice", "Bob", "Carol"],
    "Score": [85, 90, 95]
}
df = pd.DataFrame(data)
df.head()
```

Exercise: Pandas

Python Code | Start Over | Run Code

```
1  # TODO: Create a DataFrame with columns "Gene" and "Expression"
2  # TODO: Display the first 3 rows
```

Matplotlib (plot example)

Python Code | Start Over | Run Code

```
1  x = np.linspace(0, 2*np.pi, 100)
2  y = np.sin(x)
3
4  plt.plot(x, y)
5  plt.title("Sine wave")
6  plt.show()
```

Matplotlib (image visualization)

- Use plt.imshow to display 2D arrays as images.
- cmap="gray" for grayscale (optional).

Python Code | Start Over | Run Code

```
# Create a simple 28x28 "digit-like" blob
img = np.zeros((28, 28), dtype=float)
img[8:20, 10:18] = 128
img[10:18, 12:16] = 255

plt.imshow(img, cmap="gray")
plt.title("Synthetic 28x28 image")
plt.axis("off")
plt.show()
```

Exercise: Matplotlib (image)

Python Code | Start Over | Run Code

```
1  # TODO: Create a 32x32 gradient image where value increases left->right
2  # TODO: Display it with plt.imshow (optionally cmap="gray"), remove axes
```

Python essentials

- f-strings: quick formatting (logging, metrics)
- Comprehensions: concise list/dict creation
- enumerate / zip: clean loops

Python Code | Start Over | Run Code

```python
# f-strings
epoch, loss = 5, 0.1234
print(f"[epoch {epoch}] loss={loss:.4f}")

# Comprehensions
squares = [i*i for i in range(5)]
mapping = {c:i for i, c in enumerate("abc")}
print(squares, mapping)

# enumerate / zip
for (a, b) in enumerate(zip([1,2,3], [10,20,30])):
```

main

Exercise: Python essentials

Python Code | Start Over | Run Code

```
# TODO:
# 1) Using a list comprehension, build cubes = [n**3 for n in range(6)]
# 2) Use enumerate to print index/value pairs from cubes
# 3) Print "epoch=<e> acc=<a>%" using an f-string with e=3, a=97.5
```