IEEE
UCF

Introduction to Python Programming

(for Reinforcement Learning)

UCF College of Engineering and Computer Science

Table of Contents

1. Personal Introduction
2. Installing Python
3. Introduction to Python
   a. Why Python for Reinforcement Learning (RL)?
   b. Control Flow
   c. Functions
   d. Object Oriented Programming (OOP)
4. Python for (Deep) Reinforcement Learning
   a. Numpy
   b. TensorFlow / Pytorch
   c. OpenAI Gymnasium
5. How to Debug on the Fly

Personal Introduction

- Senior at UCF studying Computer Engineering
- Co-authored Deep RL research with Professor Enyioha
- Done Deep Learning and robotics research and engineering at MIT and NASA for internships

Installing Python

Setup Instructions (Windows) - Python Install

1. Go to the official download page:
   a. https://www.python.org/downloads/release/python-31011/
   b. Scroll down to Files, and download:
      i. Windows installer (64-bit) — Windows installer (64-bit) (.exe)
2. Run the installer:
   a. Go to File Explorer and open “Downloads”
   b. Double-click “python-3.10.11-amd64.exe”
   c. Uncheck “Use Admin Privileges”
   d. Check "Add Python 3.10 to PATH"
   e. Click “Install Now”
   f. If this does not work or an error comes up, let me know

Setup Instructions (Windows) - Environment Setup

1. Verify Python
   a. Open Command Prompt
   b. Run: python --version
      i. Should say “Python 3.10.11”
2. Make requirements.txt
   a. Open Notepad
   b. Write these two lines EXACTLY:

```
gymnasium[classic-control,atari]
stable-baselines3[extra]
```

   c. Save the file, call it “requirements.txt” and save it to your Downloads folder
3. Install Libraries
   a. python -m pip install -U pip
   b. cd Downloads
   c. python -m pip install -r .\requirements.txt

Setup Instructions (Linux)

1. Install Python 3.10.11
   a. sudo apt update && sudo apt upgrade -y
   b. sudo apt install software-properties-common -y
   c. sudo add-apt-repository ppa:deadsnakes/ppa
   d. sudo apt install python3.10
2. Setup Environment
   a. python3.10 -m venv cca-env
   b. source cca-env/bin/activate
   c. pip install --upgrade pip
   d. pip install -r requirements.txt

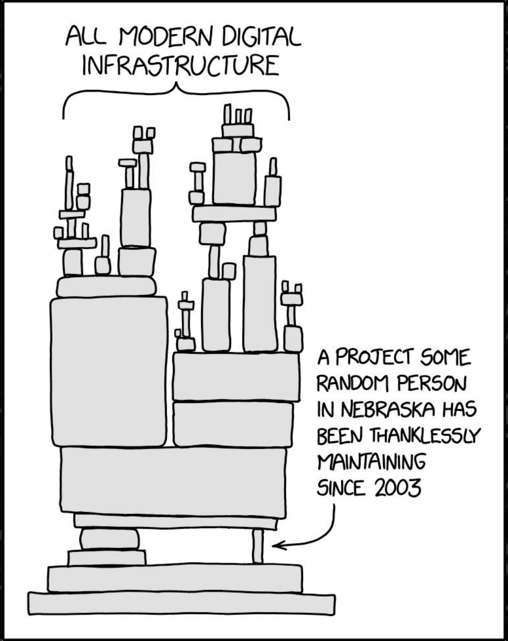
Introduction to Python Programming

IEEE UCF

UCF College of Engineering and Computer Science

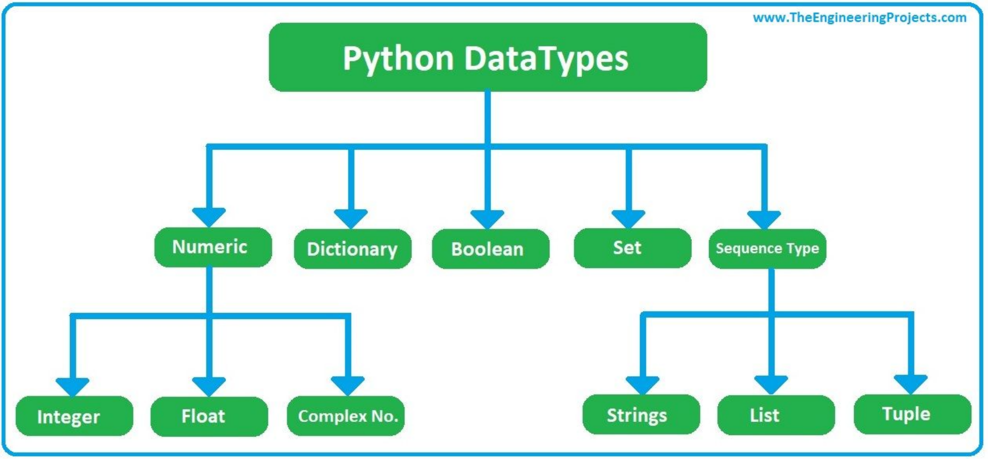
Why Python for Reinforcement Learning?

- Python is one of the easiest programming languages to learn
- Decades of development in machine learning in Python means there are plenty of tools readily available to use in Python
- Researchers and Engineers don't have to code everything from scratch if we use these tools


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Relevant XKCD Comic

Variables



- All variables can be any kind of data type in Python

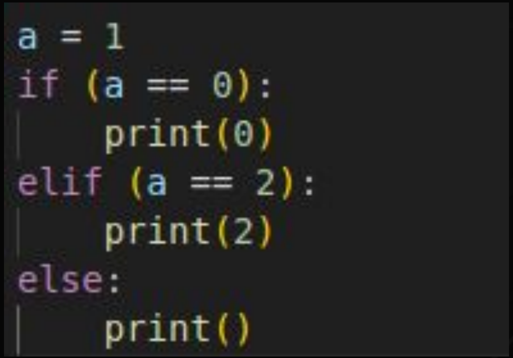
Operators

How we change the value of variables

- Arithmetic
    - + (Addition)
    - - (Subtraction)
    - * (Multiplication)
    - / (Division)
- Logical
    - “and”
    - “or”
    - “Not”
- Comparison
    - a == b (are a and b equal?)
    - a != b (are a and b <u>not</u> equal?)

Control Flow - If Statements

- If Statements
  - "If a equals 0, print "0".
    Else if a equals 2, print "2".
    Else print nothing"



```
a = 1
if (a == 0):
    print(0)
elif (a == 2):
    print(2)
else:
    print()
```

Control Flow - Loops

- For Loops
  - “For each element in this sequence, do this”

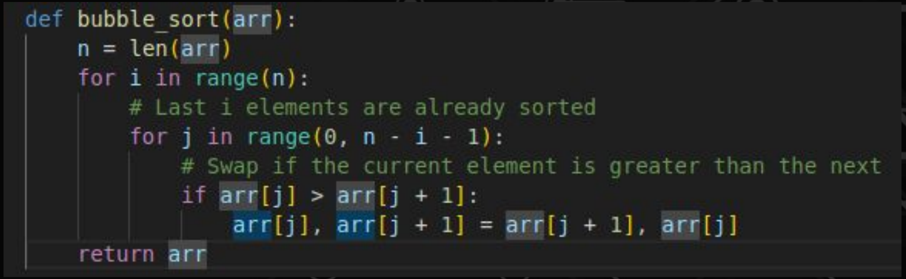
```
number_sequence = [1, 2, 3, 4 ,5]
for number in number_sequence:
    print(number, end=", ")
# Output:
# 1, 2, 3, 4, 5,
```

- While Loops
  - “Loop until this condition is not true”

```
number = 0
while (number != 5):
    number += 1
    print(number, end=", ")
print("\nexiting, number now equals 5")
# Output:
# 1, 2, 3, 4, 5,
# exiting, number now equals 5
```

Functions

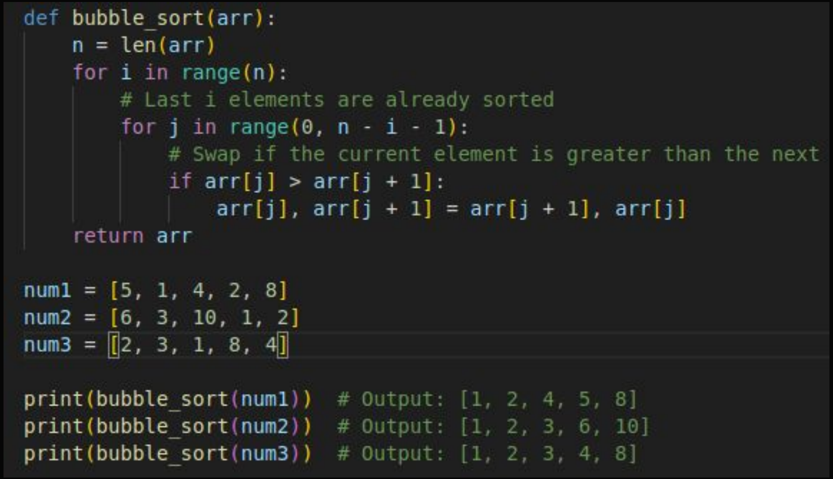
- This is how programmers abstract away large amounts of code
  - If you want to run the same 10 lines of code 3 times, you don’t want to write the 10 lins 3 times. A function lets you write it once and call it 3 times.
- Exact same as the mathematical function notation
  - y = f(x), where f is the function
- Syntax:
  - **‘def’**: declare function definition
  - **‘bubble_sort’**: function name
  - **‘arr’:** the only argument for this function, there can be more
  - **‘return’**: what value to return
    (‘y’ in y = f(x))



```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        # Last i elements are already sorted
        for j in range(0, n - i - 1):
            # Swap if the current element is greater than the next
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```

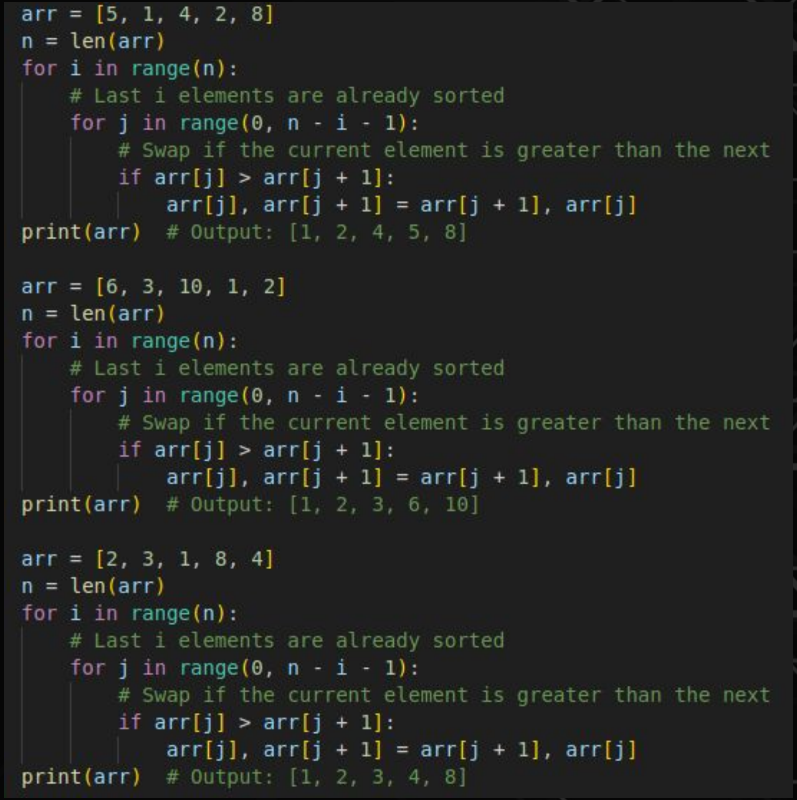
Functions (Example)

Using Functions:



```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        # Last i elements are already sorted
        for j in range(0, n - i - 1):
            # Swap if the current element is greater than the next
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

num1 = [5, 1, 4, 2, 8]
num2 = [6, 3, 10, 1, 2]
num3 = [2, 3, 1, 8, 4]

print(bubble_sort(num1))  # Output: [1, 2, 4, 5, 8]
print(bubble_sort(num2))  # Output: [1, 2, 3, 6, 10]
print(bubble_sort(num3))  # Output: [1, 2, 3, 4, 8]
```

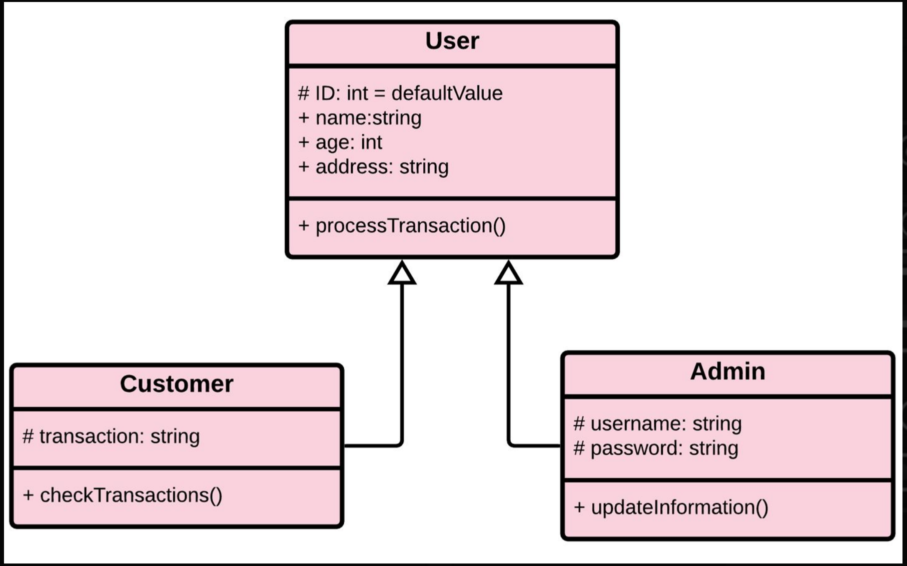
Not Using Functions:



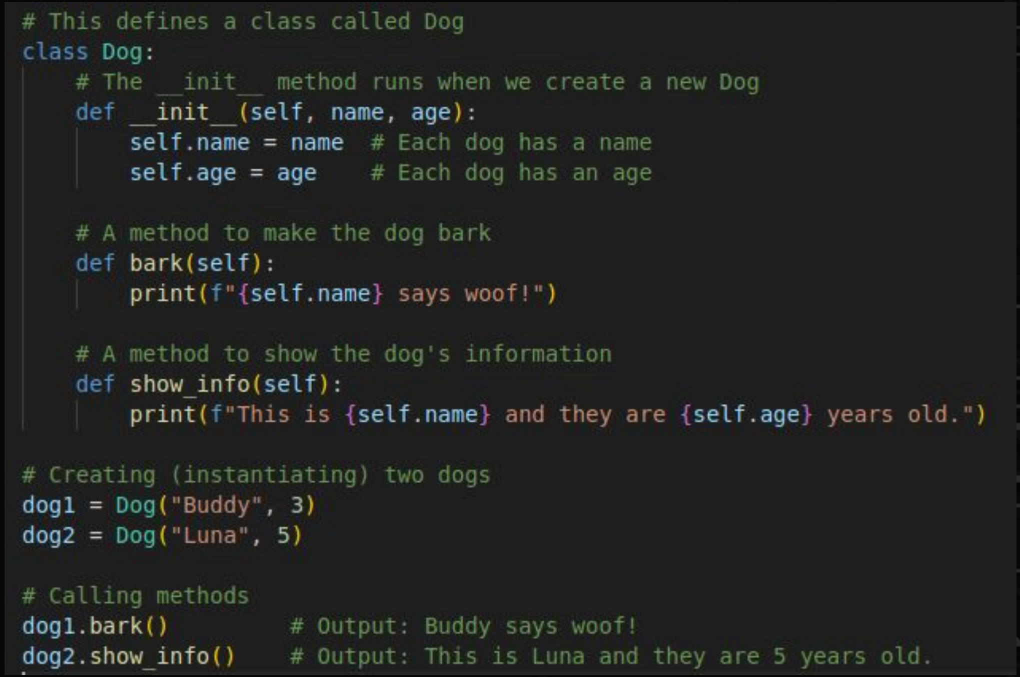
```
arr = [5, 1, 4, 2, 8]
n = len(arr)
for i in range(n):
    # Last i elements are already sorted
    for j in range(0, n - i - 1):
        # Swap if the current element is greater than the next
        if arr[j] > arr[j + 1]:
            arr[j], arr[j + 1] = arr[j + 1], arr[j]
print(arr)  # Output: [1, 2, 4, 5, 8]

arr = [6, 3, 10, 1, 2]
n = len(arr)
for i in range(n):
    # Last i elements are already sorted
    for j in range(0, n - i - 1):
        # Swap if the current element is greater than the next
        if arr[j] > arr[j + 1]:
            arr[j], arr[j + 1] = arr[j + 1], arr[j]
print(arr)  # Output: [1, 2, 3, 6, 10]

arr = [2, 3, 1, 8, 4]
n = len(arr)
for i in range(n):
    # Last i elements are already sorted
    for j in range(0, n - i - 1):
        # Swap if the current element is greater than the next
        if arr[j] > arr[j + 1]:
            arr[j], arr[j + 1] = arr[j + 1], arr[j]
print(arr)  # Output: [1, 2, 3, 4, 8]
```

Object Oriented Programming

- A way to make custom data types that have functions and variables that they own



Object Oriented Programming (Example)



```python
# This defines a class called Dog
class Dog:
    # The __init__ method runs when we create a new Dog
    def __init__(self, name, age):
        self.name = name   # Each dog has a name
        self.age = age     # Each dog has an age

    # A method to make the dog bark
    def bark(self):
        print(f"{self.name} says woof!")

    # A method to show the dog's information
    def show_info(self):
        print(f"This is {self.name} and they are {self.age} years old.")

# Creating (instantiating) two dogs
dog1 = Dog("Buddy", 3)
dog2 = Dog("Luna", 5)

# Calling methods
dog1.bark()          # Output: Buddy says woof!
dog2.show_info()     # Output: This is Luna and they are 5 years old.
```

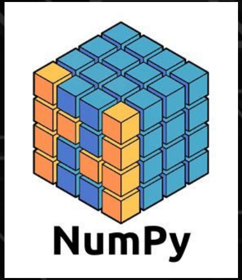
Python for (Deep) Reinforcement Learning

IEEE UCF

UCF College of Engineering and Computer Science

NumPy

- NumPy is the backbone of most major machine learning libraries and frameworks due to its incredibly fast implementations of matrix and statistics math
- Given machine learning’s extreme amount of matrix operations it’s incredibly important to be able to do these operations fast
    - e.g . training chatGPT can reach trillions to quadrillions of matrix calculations (1,000,000,000,000 - 1,000,000,000,000,000)

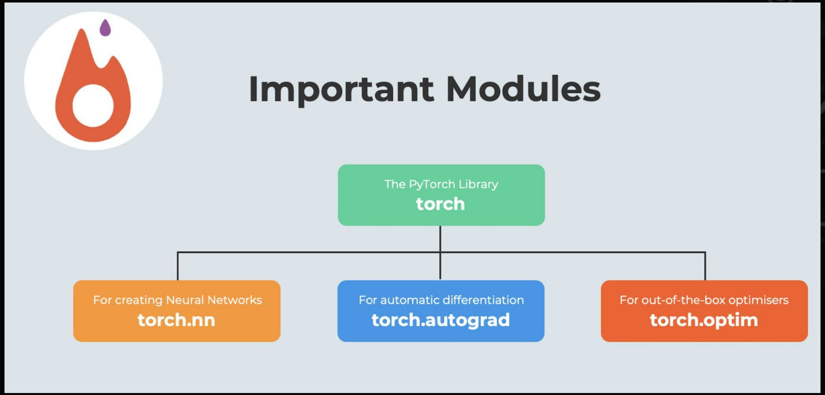


NumPy - Main Functionality

- Efficient storage and manipulation of large multidimensional arrays (ndarray)
- Supports fast element-wise operations
- Built-in fast operators for matrix math (dot product, mean, add, multiply, etc)
- Enables vectorized computations, reducing the need for explicit loops
- Offers advanced indexing, slicing, and reshaping for flexible data access
- Seamlessly integrates with other scientific libraries (e.g., SciPy, Pandas, TensorFlow)

PyTorch

- With PyTorch, you can build an entire neural network with a single line of code
- Holds efficient implementations of almost every machine learning algorithm you can think of, both classical and modern
- Builds on top of numpy to implement those models



Important Modules

The PyTorch Library
torch

For creating Neural Networks
torch.nn

For automatic differentiation
torch.autograd

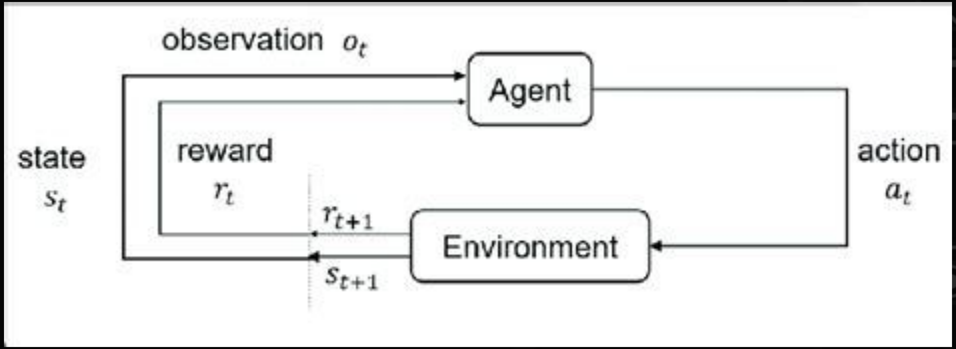
For out-of-the-box optimisers
torch.optim

PyTorch - Main Functionalities

- Tensors for efficient multidimensional array operations with optional GPU acceleration
- Automatic differentiation for gradient computation
- A modular neural network API (torch.nn) for building and training deep learning models easily
- Built-in optimizers (e.g., SGD, Adam) for updating model parameters
- Easy GPU and CPU interoperability for data and model computation
- Strong ecosystem integration with libraries for vision, audio, and natural language processing

Gymnasium

- Gymnasium is an open source library that makes experimenting with Reinforcement Learning easy
- Has built in environments, models, and more to let you set up a full RL experiment in under 20 lines of code



Gymnasium Examples

Custom Environment

- Declare your action space
  - What kind of actions are you expecting?
  - E.g. one value from -100 to 100 representing left or right turn intensity
- Declare your observation space
  - What kind of observations should the agent be able to see to learn from?
  - E.g. how close the agent is to the landing zone in meters
- Defining a reward function
  - Define a reward function that lets the agent know when they're doing something good.
  - E.g. more reward for staying upright in a cartPole environment

Custom Agent

- What algorithm are you using to train your agent?
  - Many, many options:
  - DQN
  - DDPG
  - TD3
  - SAC
  - etc…

How to Debug on the Fly

IEEE UCF

UCF College of Engineering and Computer Science