Python Language Basics

Learn the basics of programming in Python

These Tutorials are for:

- Those with no Python experience
- Those will little to no programming experience
- Those who want to learn Python

- Side note: there is no OS requirement but I record these tutorials on a mac

What will we Learn?

- What makes Python unique
- How to keep track of data
- How to implement logic
- How to implement responsiveness
- How to create data structures

Topics

1. Intro to Python
2. Variables and operators
3. Collection types
4. Conditionals and loops
5. Functions
6. Classes and objects

How will the Tutorials Work?

- Most topics have conceptual and practical parts
- Conceptual parts will be explained in the slides
- Practical parts will be explored through code examples
- Try to use real-world examples to make the topics easier to understand
- The goal is to get you up to speed with Python basics so you can start programming as soon as possible

Intro to Python

Learn about what Python has to offer

What will we Cover?

- What is Python?
- What are some applications of Python?
- What makes Python unique?
- Why learn Python?

What is Python?

- Python is a high level, general purpose, interpreted programming language
- Considered to be imperative, object oriented, functional, and reflective
- First introduced in 1991 by Guido van Rossum
- Last stable release was v3.8.1 at the end of 2019

Python Applications

- Desktop programs
- Web development with Flask
- Game development with Pygame
- Data science
- Machine learning

What Makes Python Unique?

- Unique syntax that does away with brackets and uses indentation instead
- Dynamically typed variables
- Very flexible but slightly slower runtime
- Code is executed line by line rather than all at once

Why Learn Python?

- Expressive which makes it easy to read and write
- Tons of great frameworks and libraries
- Tons of support forums
- One of the most in-demand languages, especially when it comes to data science

Intro to Variables

Learn about variables and how to use them in Python

What will we Cover?

- What are variables?
- What types of variables does Python have?
- Go over some examples

What are Variables?

- Variables act as placeholders for data
- Store values that can change
- Have name, type, and value
- Takes on “None” value if no value is assigned

Variable Types

- Types dictate the kind of data that a variable can hold
- Types are flexible in Python
- We can create custom types
- We can convert between most types
- Standard library types are:
  - Bools
  - Floats
  - Ints
  - Strings

Booleans

- Represent true or false data
- Basically binary, something is either 1 or 0
- Used when variable can only take on one of two states, such as an on/off switch
- Often stores results of some comparison operation

Ints and Floats

- Represent numerical data
- Ints are whole numbers
- Floats are decimal numbers
- No native Double type in Python

Strings

- Represent text data
- Anything between “” or ‘’
- Use for names, messages, text, etc.
- Can even contain text representation of numbers or true/false values

Variables Examples

Look at some examples of variables in action

What will we Cover?

- Creating different types of variables in Python
- Work with:
  - Booleans (bool)
  - Integers (int)
  - Floats (float)
  - Strings (str)

Conversion Examples

Look at some examples of converting between different variable types in Python

What will we Cover?

- Converting between different variable types
- Talk about what is and isn't allowed
- Go over some interesting tricks

Operators

Explore some of the basic operators available in Python

What will we Cover?

- What are operators?
- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Others

What are Operators?

- Operators enable us to perform operations on variables
- A few operators modify a variable value
- Most operators produce a new result without modifying inputs
- Can be strung together but always follow order of operations

Arithmetic Operators

- Generally used for numerical calculations
- Can be used to glue strings together
- Produce a new numerical or string result
- +, -, *, /, %, //, **

Assignment Operators

- Used to assign values to variables
- Can be used in combination with arithmetic operators
- =
- +=, -=, *=, /=, %=, //=, **=

Comparison Operators

- Used to compare two values
- Can be numerical, string, or boolean variables
- ==, !=, >, >=, <, <=

Logical Operators

- Used to test additional conditions
- Typically used on booleans
- not, or, and

Other Operators

- Ternary operator: if else
- Identity: is, is not
- Membership: in, not in
- Bitwise: &, |, ^, ~, <<, >>

Operators Example 1

Go over some examples of operations performed on variables

What will we Cover?

- Arithmetic operators
- Assignment operators

Operators Example 2

Go over some examples of operations performed on variables

What will we Cover?

- Comparison operators
- Logical operators

Collections

Explore different collection types in Python

What will we Cover?

- What are collections?
- Lists
- Tuples
- Dictionaries
- Ranges

What are Collections?

- Collections provide a way to store multiple values in a single variable
- Items in a collection are considered elements
- Elements can also be collections
- Usually come with additional functionality

Lists

- Lists store 0 or more items at specific indexes
- Access elements based on their index
- Cannot access items at indexes that don’t exist
- Can contain multiple types of variables in Python
- Mutable

Tuples

- Similar to lists
- Access elements based on their index
- Immutable

Dictionaries

- Store key-value pairs at each index
- Each element is a tuple of key-value pair
- Access values based on key, not index
- Can also retrieve list of just keys or just values
- Mutable

Ranges

- Ranges represent lists of consecutive whole numbers but don’t have the functionality of lists
- Specify start and end values
- Can also specify step value
- Wrap in reversed() function to count backwards

Lists

Go over some examples of lists in action

What will we Cover?

- Creating lists
- Accessing/modifying list elements
- Common list operations

Multidimensional Lists

Go over some examples of multidimensional lists in action

What will we Cover?

- Creating 2D lists (matrices)
- Accessing/modifying list elements

Tuples

Go over some examples of tuples

What will we Cover?

- Creating tuples
- Accessing tuple elements
- Common tuple operations
- Compare to lists

Dictionaries

Go over some examples of dictionaries

What will we Cover?

- Creating dictionaries
- Accessing/modifying dictionary elements
- Common dictionary operations

Ranges

Go over some examples of ranges

What will we Cover?

- Create a range with start and end parameters
- Add step parameter
- Reverse a range
- In, not in operators

Conditionals

Explore conditionals in Python and how to add logic

What will we Cover?

- What are conditionals
- If statements
- Elif statements
- Else statements
- If statement variants

What are Conditionals?

- Conditionals allow us to add logic to the code
- We can choose which parts of the code to execute based on the outcome of certain tests
- Tests are typically comparing variables or testing the value of a single variable and always return true or false

If Statements

- Execute some statements if a test evaluates to true
- Do nothing if the test evaluates to false
- The test could check the value of a boolean variable or test equality between values

Elif Statements

- Provide an extra if statements
- Only executed if previous test(s) fail
- Can only come after if statement
- Can chain together as many elif statements as you want

Else Statements

- Provides a failsafe
- Only executed if previous test(s) fail
- Can only come after if statement or elif statement
- Doesn’t actually test anything, just executes code

If Statement Variants

- Can nest if statements to add subsequent tests
- Can add multiple testing conditions in if statement with or/and operators to test multiple conditions at once
- Can add multiple if statements in a row without elif to perform all of the tests

If, Elif, Else Statement Examples

Go over examples of if, elif, and else statements

What will we Cover?

- If statements
- Elif statements
- Else statements
- Ternary operator

If Statements Variants Examples

Go over some if statement variants

What will we Cover?

- Consecutive if statements
- Nested if statements
- Adding multiple test cases in one if statement

Loops

Explore the concepts of while and for loops

What will we Cover?

- What are loops?
- While loops
- For loops

What are Loops?

- Loops provide a way to automate code to run multiple times
- We place statements in a loop and they are executed as continuously until some break condition is met
- We can put whatever we want in the loop body but all code within will be executed each time the loop runs

What are While Loops?

- While loops function similar to if statements but run the code continuously rather than just once
- Break out of while loop once the test condition evaluates to false
- Very useful for when we don’t know exactly how many times the loop should run
- Can run infinitely if we’re not careful

What are For Loops?

- Often referred to as for in loops in Python
- Run a set number of times with specific start and end points
- Visits every element in some range or list, starting at the beginning and finishing at the end

While Loop Example

Go over an example of a while loop in action

What will we Cover?

- Creating a while loop
- Running a while loop
- Breaking out of a while loop

For Loop Example

Go over an example of a for loop in action

What will we Cover?

- Creating a for in loop
- Running a for in loop
- Working with ranges and lists coupled with for in loops

Functions

Explore the concept of functions

What will we Cover?

- What are functions?
- What are parameters?
- What are return values?

What are Functions?

- Functions contain code to be executed exactly when we choose
- Code remains hidden until we run the function
- Run the code by calling on the function
- Can also receive inputs and produce outputs

What are Parameters?

- Parameters are inputs into a function
- Functions can have 0 or more parameters
- We use these like regular variables within a function
- Pass these values into the function when calling it
- Can use default values if we don’t want to pass in a value every time but need to use the variable in the function body

What are Return Values?

- Return values are outputs from a function
- Specify these in the function body with a return statement
- Function breaks and outputs a value (or not) when it reaches this
- If not return statement, function breaks after the last statement
- Can use the output like a variable when calling the function

Functions Examples

Go over some examples of implementing and calling functions

What will we Cover?

- Implementing functions
- Calling on functions
- Variable scope

Parameters and Return Values

Go over some examples of using parameters and return values in functions

What will we Cover?

- Adding parameters
- Adding return statements
- Passing in parameters and return values when calling functions

Classes and Objects

Explore the concept of classes and how to use them as objects

What will we Cover?

- What are objects?
- What are classes?
- How do we use classes and objects?
- Inheritance
- Static variables and functions

What are Objects?

- Objects are entities with state/properties and behaviour
- State is a mix of all of the values of an object
- Behaviour typically modifies the state
- Modelled after real life objects

What are Classes?

- Classes are implementations of objects
- State is represented with variables (fields)
- Behaviour is represented with functions (methods)
- Special initializer function helps to set up the initial state (set variable values)
- Use the initializer to create instances of classes (objects)

How do we use Classes and Objects?

- Class simply acts as a blueprint for an object
- Use the initializer to create instances of classes (objects)
- An object's initial state is set upon instantiation
- We can use the object to access fields or execute methods

What is Inheritance?

- Sometimes classes need to be similar but have some unique fields or methods
- For these we use inheritance where one class can inherit everything from another but also add its own stuff
- A subclass will inherit from a superclass
- Sometimes we override a superclass implementation of a variable or function to provide a new one

What are Static Members?

- Static variables and functions belong to a whole class rather than just an instance
- Static variables hold their values across all instances of the class
- Don’t have to create an instance to get the value
- Static functions follow a similar concept to variables

Class Example

Create a custom class

What will we Cover?

- Create a custom class to represent an object
- Add some fields
- Add an initializer
- Add some methods

Object Example

Create an instance of our class and play around with it

What will we Cover?

- Create an object from our class
- Access the object's fields
- Execute the object's methods

Inheritance Example

Create a subclass for our superclass

What will we Cover?

- Create a subclass
- Compare it to the superclass
- Go over the difference in usage between the two

Static Members Example

Create and use some static members

What will we Cover?

- Create some static variables
- Create some static functions
- Access and use the static members

Python Language Basics

Learn the basics of programming in Python

What did we Learn?

- What is Python?
- How do we store data in Python?
- How do we add logic and responsiveness in Python
- How do we automate code in Python?

Topics

1. Intro to Python
2. Variables and operators
3. Collection types
4. Conditionals and loops
5. Functions
6. Classes and objects

Where to go from here

- Go over the topics and come up with more examples
- Write some simple Python programs
- Explore more Python concepts
- Explore popular Python libraries