Python 101

Created by Avery Reed. Find the pdf here.

Sites & Downloads

Python.org - Download - Compiler

Introduction to Python

This is the first step in learning Python, just printing a simple “Hello, Python!” message to the console.

```
print("Hello, Python!")
```

Basic data types (strings, integers, floats, booleans)

In this topic, you’ll learn about the basic data types in Python, such as strings, integers, floats, and booleans, and how to work with them.

```
name = "John"
age = 30
height = 1.8
is_male = True

print(name, age, height, is_male)
```

User Input

In this topic, you’ll learn how to get input from the user in a Python program using the input() function.

```
name = input("What is your name? ")
print("Hello, " + name + "!")
```

Control flow (if/else statements, for loops, while loops)

Control flow statements are used to control the flow of execution in a program. You’ll learn about if/else statements, for loops, and while loops in this topic.

```
x = 5

if x > 0:
    print("x is positive")
else:
    print("x is negative")

for i in range(5):
    print(i)

i = 0
while i < 5:
    print(i)
    i += 1
```

Functions and modules

Functions are used to group a set of related code together, and modules are used to organize code and reuse it across multiple programs. You'll learn about functions and modules in this topic.

```
def add(a, b):
    return a + b

import math
print(math.sqrt(16))
```

Lists, tuples, and dictionaries

Lists, tuples, and dictionaries are all used to store multiple values in Python. You'll learn about the differences between these data structures and how to work with them.

```
my_list = [1, 2, 3, 4]
my_tuple = (1, 2, 3, 4)
my_dict = {"name": "John", "age": 30}

print(my_list)
print(my_tuple)
print(my_dict)
```

File input/output

In this topic, you'll learn how to read from and write to files in Python.

File writing

```
with open("example.txt", "w") as file:
    file.write("Writing to a file")
```

File reading

```
with open("example.txt", "r") as file:
    print(file.read())
```

Exception handling

Exceptions are events that are triggered when an error occurs in a program. You'll learn how to use try/except statements to handle exceptions in this topic.

```
try:
    x = 5 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

Object-oriented programming

Object-oriented programming (OOP) is a programming paradigm that is based on the concept of objects, which can have properties and methods. You'll learn about classes, objects, inheritance, and polymorphism in this topic.

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        print("Woof woof!")

dog = Dog("Max", 3)
print(dog.name)
dog.bark()
```

Regular expressions

Regular expressions are a powerful tool for searching and manipulating text. You'll learn how to use regular expressions to match patterns in strings in this topic.

```
import re

text = "My phone number is 555-555-5555"
phone_number = re.search(r"\d{3}-\d{3}-\d{4}", text)
print(phone_number.group())
```

Debugging and testing

Debugging is an important part of the development process, and testing is a way to ensure that your code is working correctly. In this topic, you'll learn about different tools and techniques for debugging and testing your code.

```
import pdb

def add(a, b):
    pdb.set_trace()
    return a + b

print(add(5, 7))
```

List comprehension

List comprehension is a way to create a new list by applying an operation to each element of an existing list. This is a concise way to write for loops.

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = [x**2 for x in numbers]
print(squared_numbers)
```

Lambda functions

A lambda function is an anonymous function that can be created without a name. They are useful for short, throw-away functions.

```
add = lambda x, y: x + y
print(add(5, 7))
```

Map and filter

map and filter are built-in functions that allow you to apply a function to each element of an iterable and filter the elements of an iterable based on a condition.

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x**2, numbers))
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(squared_numbers)
print(even_numbers)
```

List slicing

List slicing is a way to extract a subset of elements from a list.

```
numbers = [1, 2, 3, 4, 5]
print(numbers[1:3])
```

List sorting

This topic covers how to sort the elements of a list in ascending or descending order.

```
numbers = [3, 1, 5, 4, 2]
numbers.sort()
print(numbers)
```

String formatting

String formatting is a way to insert values into a string. In this topic you will learn about different ways to format strings in python.

```
name = "John"
age = 30
print("My name is {} and I am {} years old.".format(name, age))
```

String methods

Strings are one of the most widely used data types in Python, and they come with a number of built-in methods for working with them. This topic covers some of the most commonly used string methods, such as upper(), lower(), and split().

```
text = "Hello, World!"
print(text.upper())
print(text.lower())
print(text.split(","))
```

JSON

JSON is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. You'll learn how to work with JSON data in Python in this topic.

```
import json
data = {"name": "John", "age": 30}
json_data = json.dumps(data)
```

```
print(json_data)


with open("data.json", "r") as file:
    data = json.load(file)
print(data["name"])
```

Here is the json.data file

```
{
    "name": "John Doe",
    "age": 35,
    "address": {
        "street": "123 Main St",
        "city": "Anytown",
        "state": "CA",
        "zip": "12345"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "555-555-5555"
        },
        {
            "type": "work",
            "number": "555-555-5556"
        }
    ]
}
```

Date and Time

In this topic, you'll learn how to work with dates and times in Python, including how to create, format, and manipulate datetime objects.

```
from datetime import datetime

now = datetime.now()
print(now)
print(now.year)
```

Enumerate

enumerate() is a built-in function that allows you to iterate over a list and also keep track of the index of the current item. This topic covers how to use this function to iterate over a list.

```
fruits = ["apple", "banana", "cherry"]
for index, fruit in enumerate(fruits):
    print(index, fruit)
```

These examples are just a small subset of the many built-in functions and commands available in Python. As you continue to learn and use Python, you will discover many more powerful tools at your disposal.