# INF 354
# Python
## Introduction

Lecture 10
25 April 2019

# The contents of this lecture are as follows:


- Introduction to Python
- Why we're teaching you Python
- Benefits of Python Programming
- IDE's and Web Frameworks
- Getting Started
- Syntax – variables and types, indentation, comments, lists, dictionaries, conditionals, loops, functions, and classes

# Introduction to Python


Python is

- A general purpose programming language (Web, GUI, Scripting, …)
- Open Source
- Object oriented
- Interpreted
- Strongly and Dynamically Typed
- Uses an elegant syntax, making the programs you write easier to read.
- Comes with a large standard library that supports many common programming tasks
- Current stable version of Python is 3.7.3 (April 2019)

# Features of Python



Some programming-language features of Python are:

- A **variety of basic data types** are available: numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
- Python **supports object-oriented programming** with classes and multiple inheritance.
- Code can be grouped into **modules and packages**.
- The language supports raising and catching **exceptions**, resulting in cleaner error handling.
- Data types are **strongly and dynamically typed**. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised, so errors are caught sooner.
- Python's **automatic memory management** frees you from having to manually allocate and free memory in your code.

# Why we're teaching you Python


- Python has an active community with a vast selection of libraries and resources.
- As of Sep 2018, the Python Package Index had grown to over **70,000 libraries**
- Many packages are **geared toward data science** (collection, processing, cleaning, exploratory analysis, statistical modelling and deep analytics, and visualisation)
- Examples of popular libraries for Data Science are NumPy, Matplotlib, and Pandas
- Python is also considered **easy-to-learn and very accessible**
- These characteristics make Python a very popular platform for Data Scientists (up to 48% of Data Scientists* prefer Python)
- Python is programming platform that makes sense to use with emerging technologies like machine learning and data science.
- Many of you might wish to learn Data Science in your post graduate studies given the tremendously high demand.
- Learning Python will make it easier for you.

# Integrated Development Environments



- Visual Studio Code
- PyCharm
- Thonny
- Spyder (specifically for Data Science)
- Atom
- and many more…

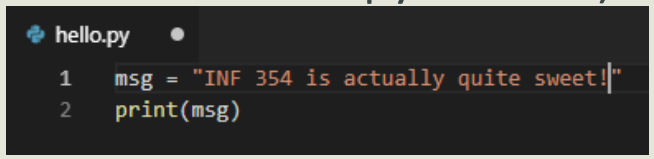
# Web Frameworks


- Django
- Flask
- web2py
- Pyramid
- TurboGears

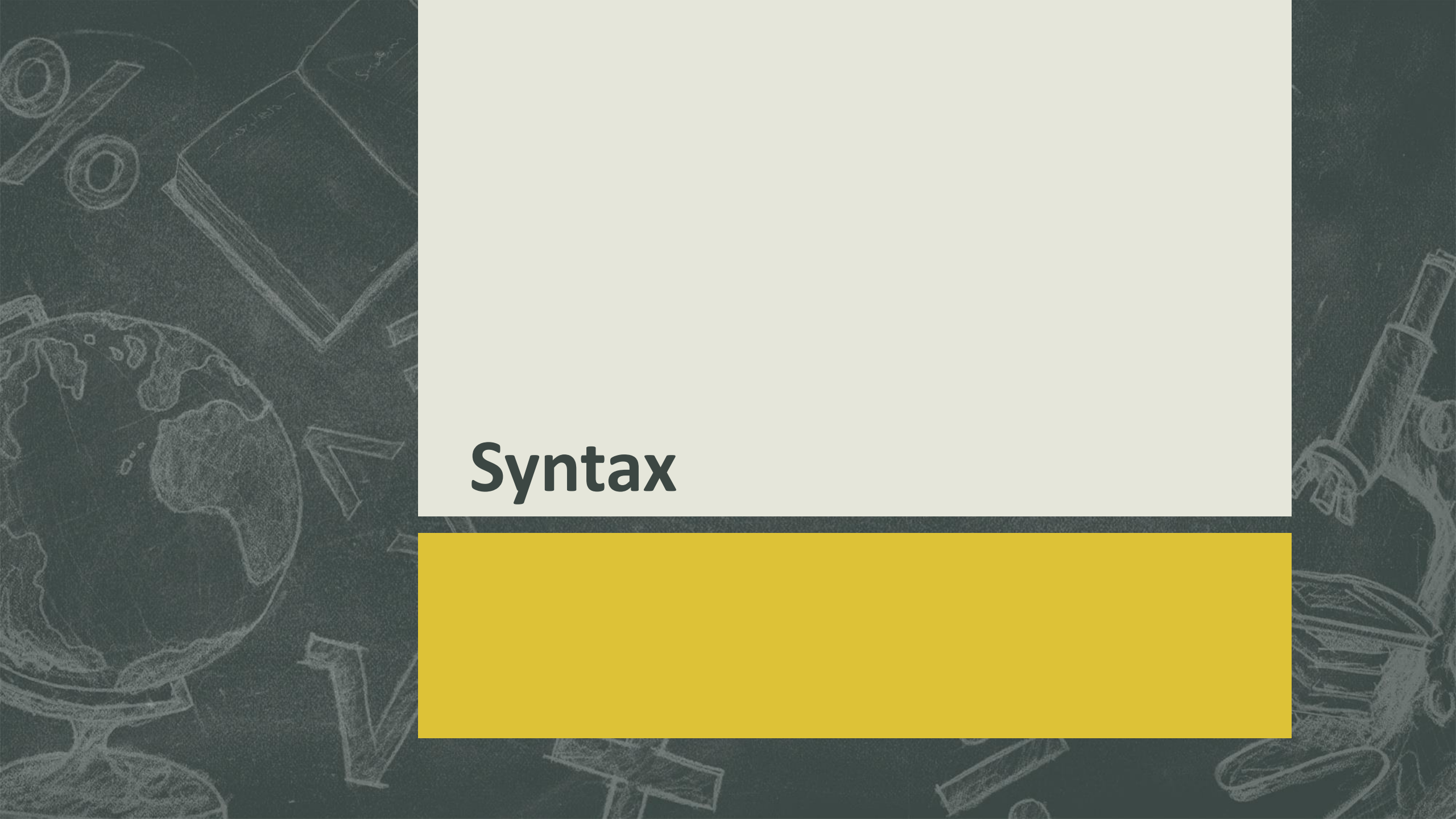
# Getting started with Python

# Steps:

- Download and Install Python: https://www.python.org/downloads/
- If you don't have **VS Code** already, download and install that as well: https://code.visualstudio.com/download
- Install the Python extension in VS Code: https://code.visualstudio.com/docs/python/python-tutorial#_prerequisites
- Create folder (wherever you like) for your first Python project.
- Right-click in your newly created empty folder and click on 'Open with Code'
- Select Python interpreter: https://code.visualstudio.com/docs/python/python-tutorial#_select-a-python-interpreter
- Create a new file called 'hello.py' (the .py extension makes it a python file)
- Write some python code in the file, such as




- Save
- Run by right-clicking file and selecting 'Run Python File in Terminal'

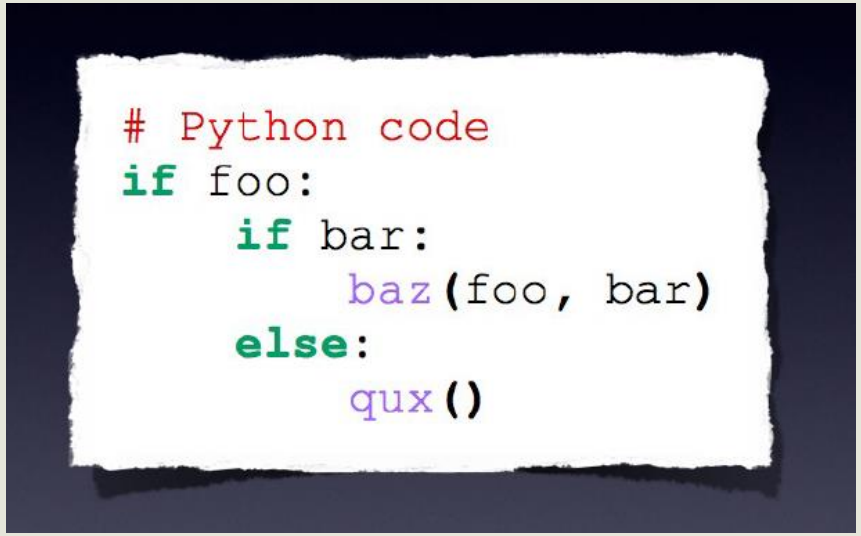


# Syntax

# Output

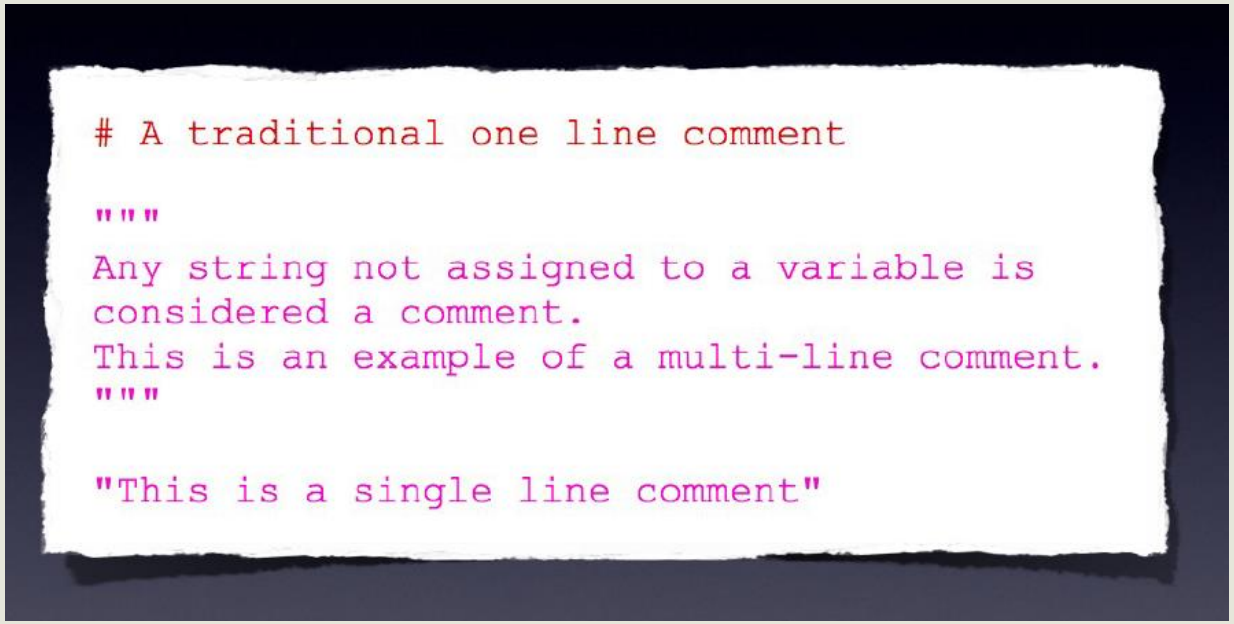

**Note:** print requires parentheses in Python 3: print(some_var)

# Indentation



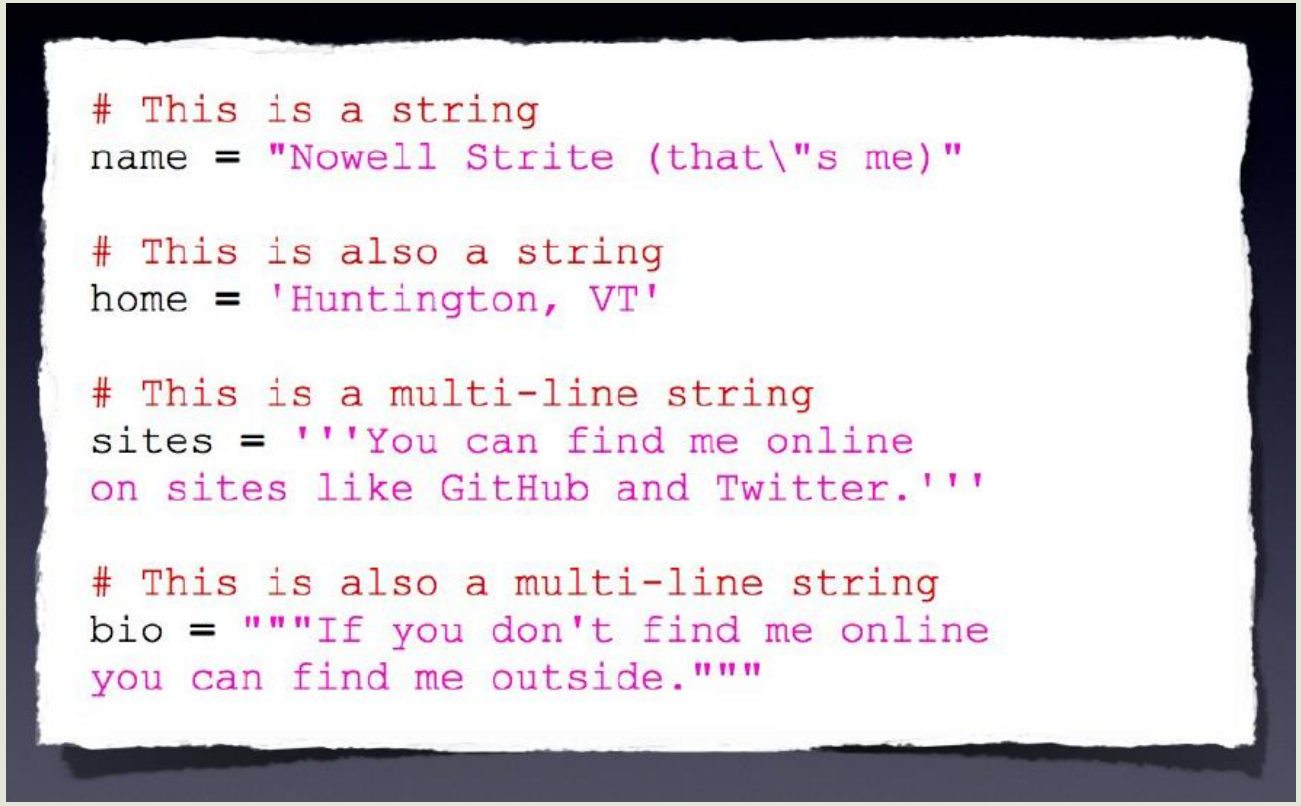

Indentation is very important in Python.
It does not use brackets to determine code blocks, it uses indentation

# Comments



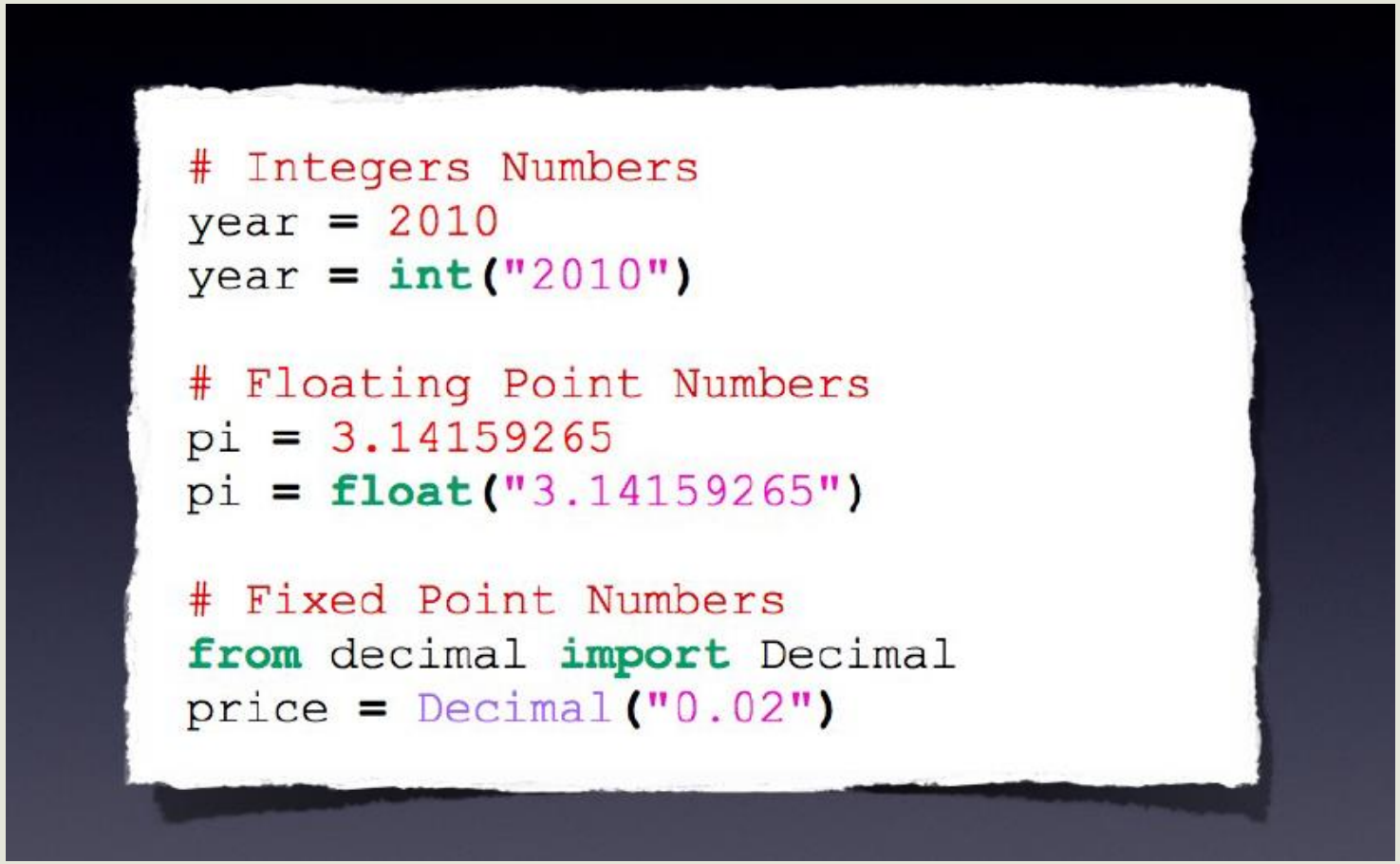
```
# A traditional one line comment

"""
Any string not assigned to a variable is
considered a comment.
This is an example of a multi-line comment.
"""

"This is a single line comment"
```

# Strings



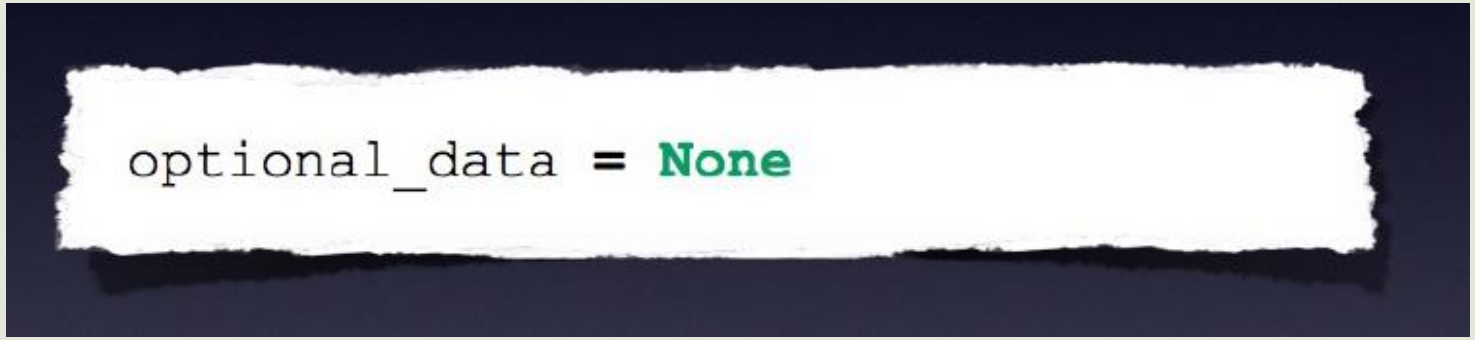
```
# This is a string
name = "Nowell Strite (that\"s me)"

# This is also a string
home = 'Huntington, VT'

# This is a multi-line string
sites = '''You can find me online
on sites like GitHub and Twitter.'''

# This is also a multi-line string
bio = """If you don't find me online
you can find me outside."""
```

# Numbers

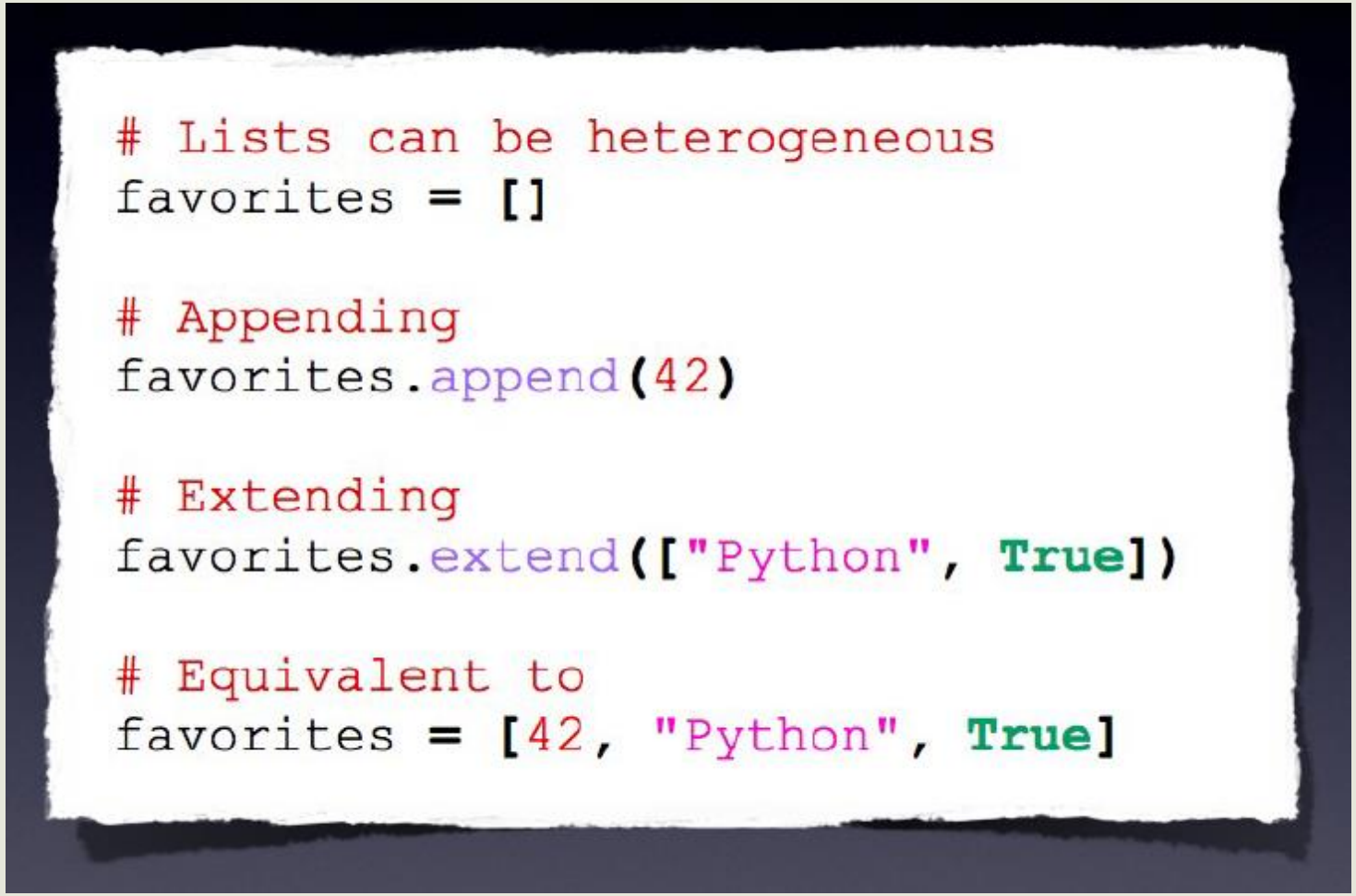


```
# Integers Numbers
year = 2010
year = int("2010")

# Floating Point Numbers
pi = 3.14159265
pi = float("3.14159265")

# Fixed Point Numbers
from decimal import Decimal
price = Decimal("0.02")
```

# Null



# Lists



```
# Lists can be heterogeneous
favorites = []

# Appending
favorites.append(42)

# Extending
favorites.extend(["Python", True])

# Equivalent to
favorites = [42, "Python", True]
```

# Lists (2)



```
numbers = [1, 2, 3, 4, 5]

len(numbers)
# 5

numbers[0]
# 1

numbers[0:2]
# [1, 2]

numbers[2:]
# [3, 4, 5]
```

# Dictionaries



```
person = {}

# Set by key / Get by key
person['name'] = 'Nowell Strite'

# Update
person.update({
    'favorites': [42, 'food'],
    'gender': 'male',
    })

# Any immutable object can be a dictionary key
person[42] = 'favorite number'
person[(44.47, -73.21)] = 'coordinates'
```

# Dictionary Methods

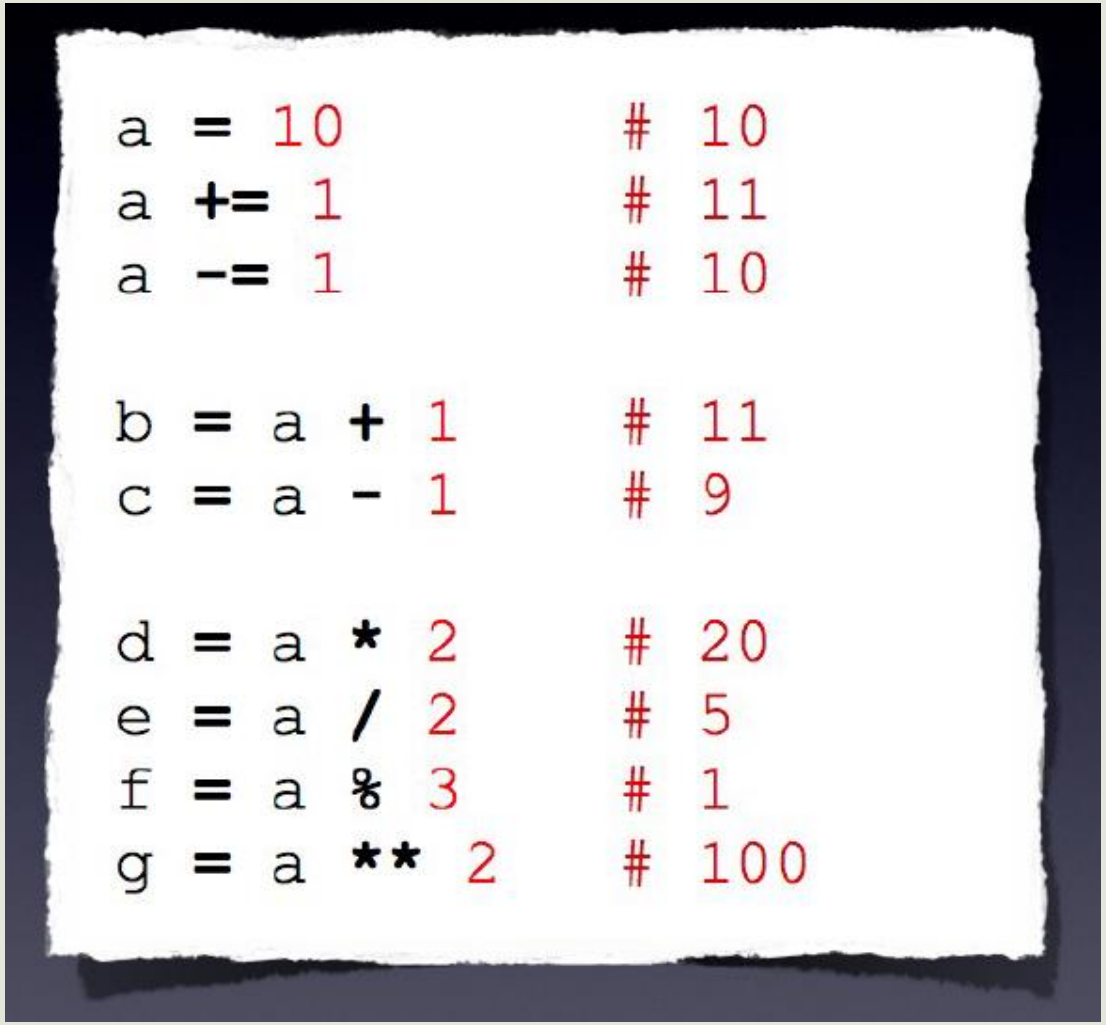


```
person = {'name': 'Nowell', 'gender': 'Male'}

person['name']
person.get('name', 'Anonymous')
# 'Nowell Strite'

person.keys()
# ['name', 'gender']

person.values()
# ['Nowell', 'Male']

person.items()
# [['name', 'Nowell'], ['gender', 'Male']]
```

# Booleans

```
# This is a boolean
is_python = True

# Everything in Python can be cast to boolean
is_python = bool("any object")

# All of these things are equivalent to False
these_are_false = False or 0 or "" or {} or []
or None

# Most everything else is equivalent to True
these_are_true = True and 1 and "Text" and
{'a': 'b'} and ['c', 'd']
```

# Arithmetic Operators



```
a = 10          # 10
a += 1          # 11
a -= 1          # 10

b = a + 1       # 11
c = a - 1       # 9

d = a * 2       # 20
e = a / 2       # 5
f = a % 3       # 1
g = a ** 2      # 100
```

# String Manipulation



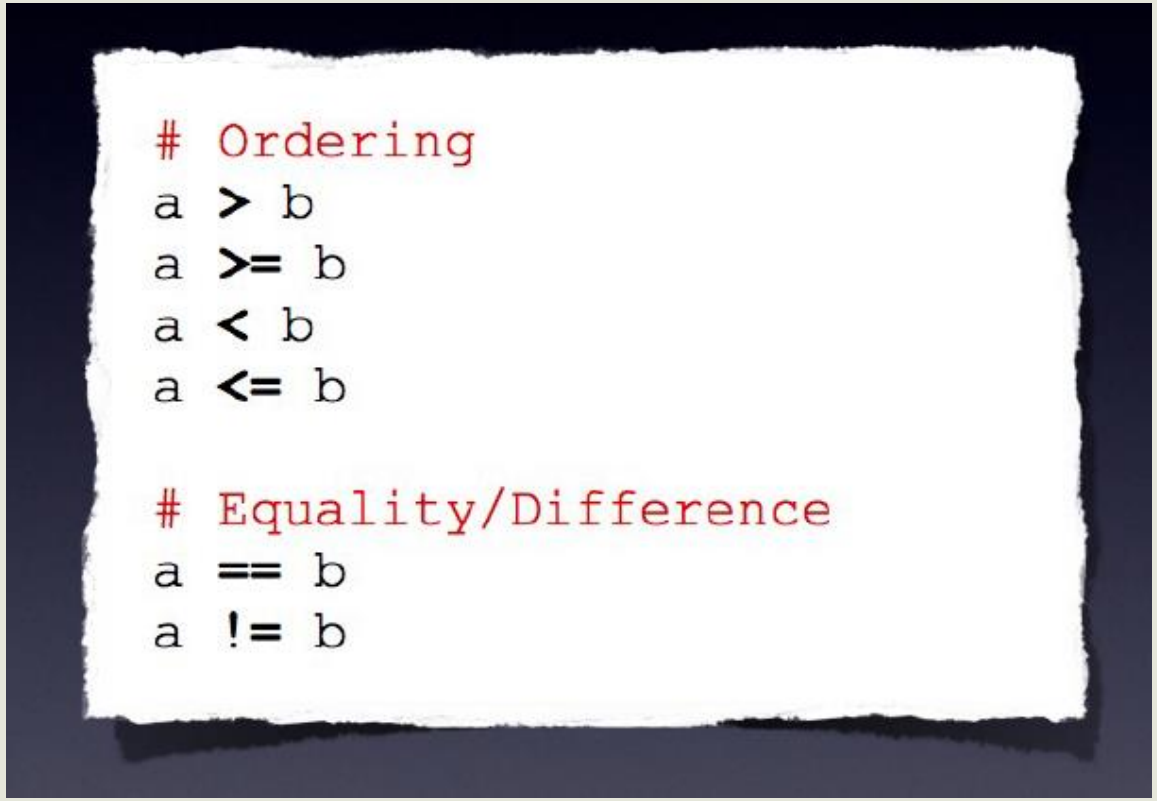
```
animals = "Cats " + "Dogs "
animals += "Rabbits"
# Cats Dogs Rabbits

fruit = ', '.join(['Apple', 'Banana', 'Orange'])
# Apple, Banana, Orange

date = '%s %d %d' % ('Sept', 11, 2010)
# Sept 11 2010

name = '%(first)s %(last)s' % {
    'first': 'Nowell',
    'last': 'Strite'}
# Nowell Strite
```

# Identity Comparison – (Compare objects as a whole?)



```
# Identity
1 is 1 == True

# Non Identity
1 is not '1' == True

# Example
bool(1) == True
bool(True) == True

1 and True == True
1 is True == False
```

# Arithmetic Comparison – (Compare values?)

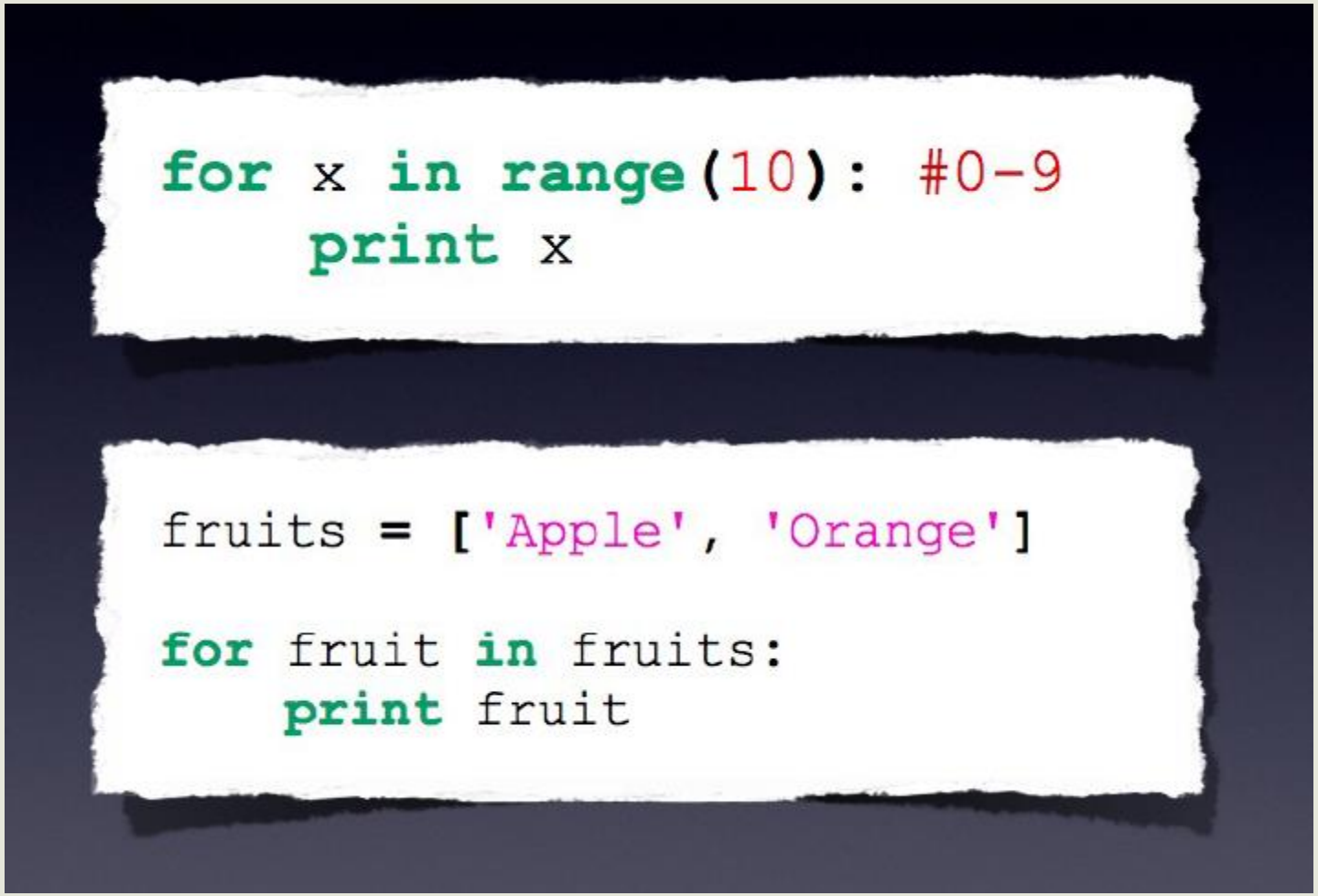


# Conditionals

```
grade = 82
if grade >= 90:
    if grade == 100:
        print 'A+'
    else:
        print "A"
elif grade >= 80:
    print "B"
elif grade >= 70:
    print "C"
else:
    print "F"

# B
```

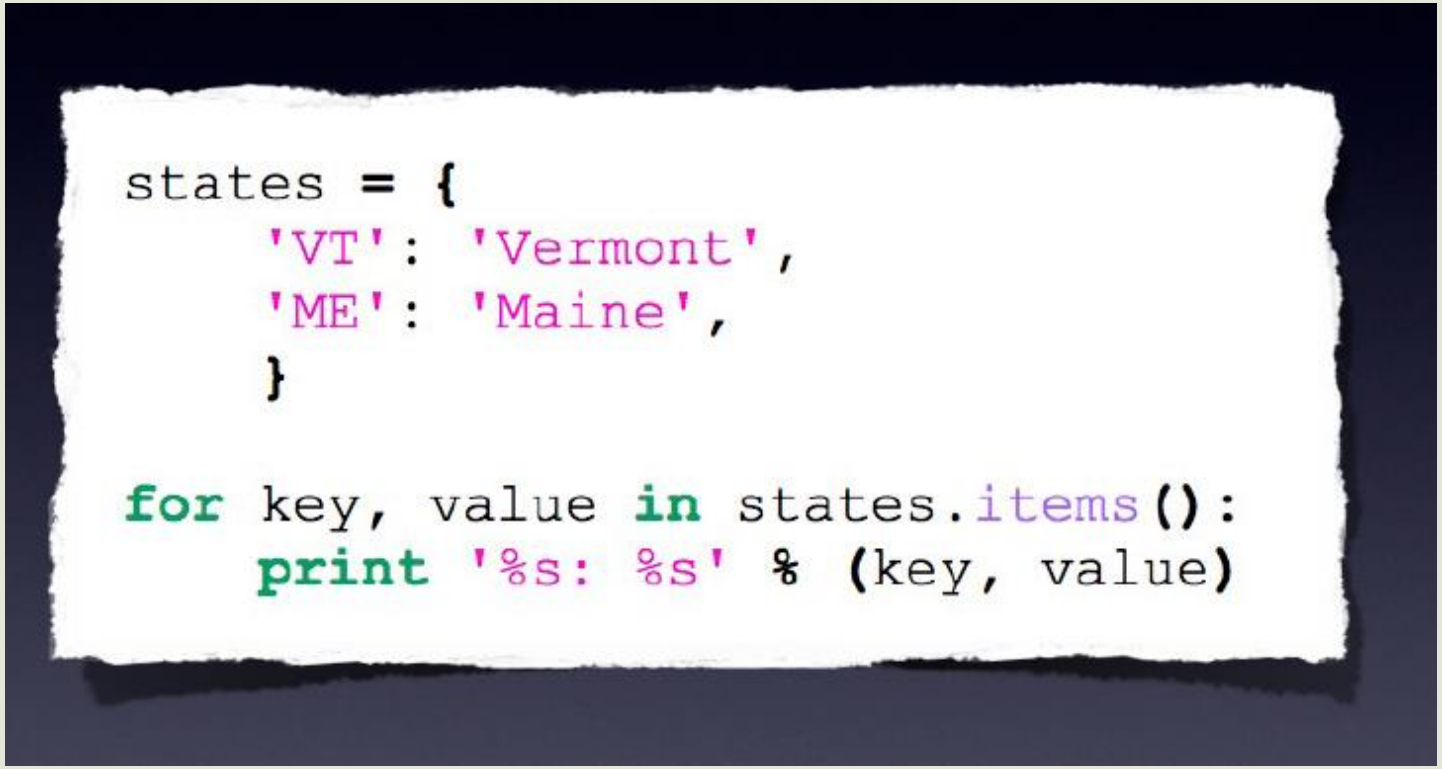
**Note:** print requires parentheses in Python 3: print(some_var)

# For Loop




**Note:** print requires parentheses in Python 3: print(some_var)

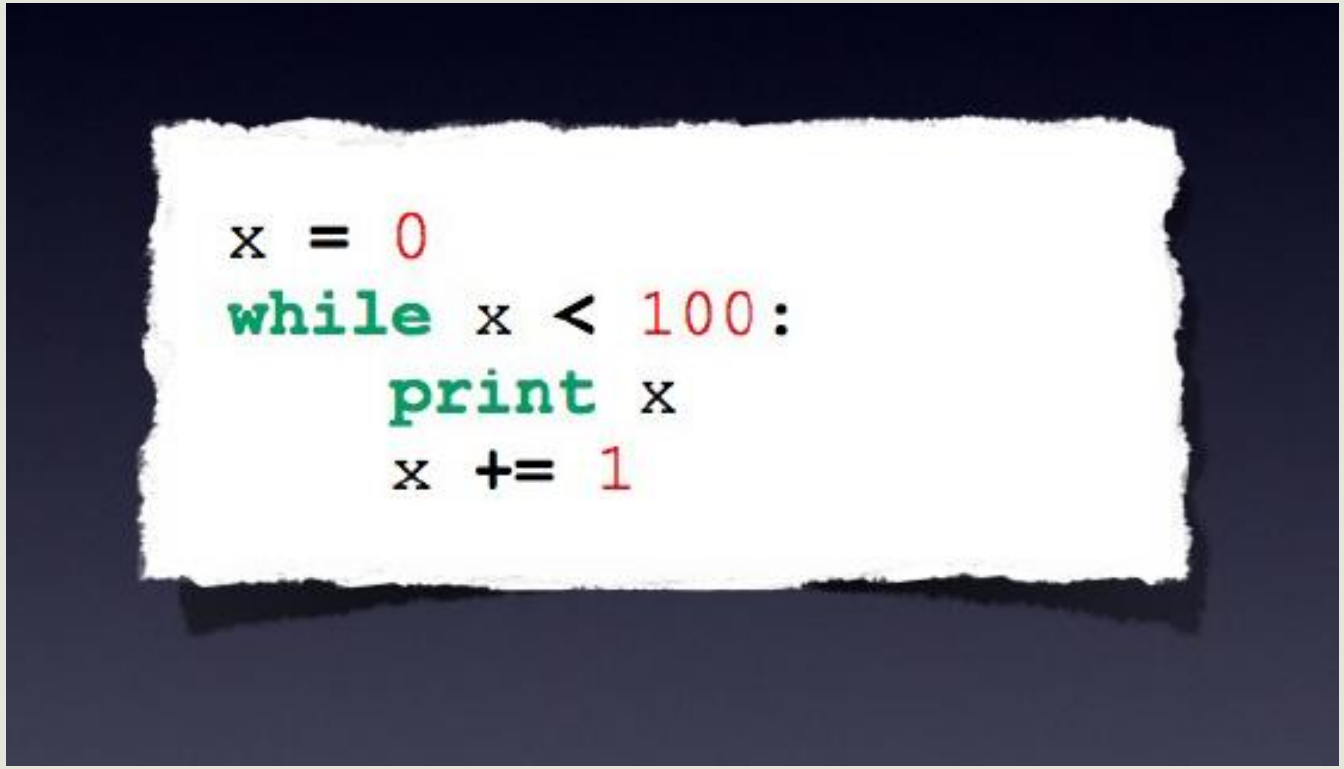
# Expanded For Loop



```
states = {
    'VT': 'Vermont',
    'ME': 'Maine',
    }

for key, value in states.items():
    print '%s: %s' % (key, value)
```

**Note:** print requires parentheses in Python 3: print(some_var)

# While Loop



```
x = 0
while x < 100:
    print x
    x += 1
```

**Note:** print requires parentheses in Python 3: print(some_var)

# List Comprehensions

Simple for loops that are used to build lists can be replaced with list comprehension notations

For example, this:

```
odds = []
for x in range(50):
    if x % 2:
        odds.append(x)
```

Can be replaced with:

```
odds = [ x for x in range(50) if x % 2 ]
```

# Basic Function

```
def my_function():
    """Function Documentation"""
    print "Hello World"
```

**Note:** print requires parentheses in Python 3: print(some_var)

# Function with arguments

```
# Positional
def add(x, y):
    return x + y

# Keyword
def shout(phrase='Yipee!'):
    print phrase

# Positional + Keyword
def echo(text, prefix=''):
    print '%s%s' % (prefix, text)
```

**Note:** print requires parentheses in Python 3: print(some_var)

# Class


**Constructor**

**Data Attributes**

**Methods**

```
class Car:
    def __init__(self, speed=0):
        self.speed = speed
        self.odometer = 0
        self.time = 0

    def say_state(self):
        print("I'm going {} kph!".format(self.speed))

    def accelerate(self):
        self.speed += 5

    def brake(self):
        if self.speed < 5:
            self.speed = 0
        else:
            self.speed -= 5

    def step(self):
        self.odometer += self.speed
        self.time += 1

    def average_speed(self):
        if self.time != 0:
            return self.odometer / self.time
        else:
            pass
```

# Instantiating class and using resultant object


**Instantiation** →

**Calling methods** →

```
my_car = Car()
print("I'm a car!")
while True:
        action = input("What should I do? [A]ccelerate, [B]rake, " "show [O]dometer, or show average [S]peed?").upper()
        if action not in "ABOS" or len(action) != 1:
                print("I don't know how to do that")
                continue
        if action == 'A':
                my_car.accelerate()
        elif action == 'B':
                my_car.brake()
        elif action == 'O':
                print("The car has driven {} kilometers".format(my_car.odometer))
        elif action == 'S':
                print("The car's average speed was {} kph".format(my_car.average_speed()))
        my_car.step()
        my_car.say_state()
```

# Video

- This weeks video is available at https://www.youtube.com/watch?v=dGeUH_bqNpA

# Helpful Resources


- https://www.w3schools.com/python/default.asp
- https://docs.python.org/3/library/index.html
- https://www.pythonforbeginners.com/cheatsheet/python-cheat-sheets