Introduction to Programming Using Python

Class Guide

[Introduction 3](#_Toc110763467)

[1. Student Goals 3](#_Toc110763468)

[2. Course Goals 3](#_Toc110763469)

[3. The Instructor 3](#_Toc110763470)

[4. The Class – Using Zoom – Even in Person 3](#_Toc110763471)

[Session #1 Introduction - Touching on the topic 4](#_Toc110763472)

[1. Direction 4](#_Toc110763473)

[5. IDLE - Not just IDLE Curiosity 4](#_Toc110763474)

[6. Samples: 4](#_Toc110763475)

[7. Python vs English – a lot in common 7](#_Toc110763476)

[8. Course Project - Guessing Game – Twenty Questions 8](#_Toc110763477)

[9. Homework: 8](#_Toc110763478)

[Session #2 Ideas, Tools, Functions 9](#_Toc110763479)

[1. Startup 9](#_Toc110763480)

[10. programming ideas 9](#_Toc110763481)

[11. Functions 10](#_Toc110763482)

[Session 3: Functions Why and How 13](#_Toc110763483)

[1. Setup 13](#_Toc110763484)

[12. Functions to develop ground work 13](#_Toc110763485)

[13. Help / info: 14](#_Toc110763486)

[14. Strings 14](#_Toc110763487)

[15. Bouncing Ball – A Closer Look 15](#_Toc110763488)

[Session 4: Dictionary 16](#_Toc110763489)

[1. Setup 16](#_Toc110763490)

[*16.* *Preview Sample – fireworks.py* 16](#_Toc110763491)

[*17.* *Dictionary Atributes* 16](#_Toc110763492)

[*18.* *do dictionary exercise(s)* 16](#_Toc110763493)

[*19.* *do motion.py example* 16](#_Toc110763494)

[20. Sample Review 17](#_Toc110763495)

[Session 5: Classes – Objects – Object Oriented 19](#_Toc110763496)

[1. Setup 19](#_Toc110763497)

[2. Preview Sample 19](#_Toc110763498)

[3. Python Style 19](#_Toc110763499)

[4. World is full of objects 19](#_Toc110763500)

[5. Classes – Usage Details 20](#_Toc110763501)

[6. 2D Billiard Ball Example 21](#_Toc110763502)

[7. Optional – Class Inheritance 22](#_Toc110763503)

[Session 6: Files 23](#_Toc110763504)

[Session 7: Graphics 25](#_Toc110763505)

[Programming / Python Tools 26](#_Toc110763506)

# Introduction

## Student Goals

### Have Fun

### Discover Programming

### Understand Programming

### Decide if/how much programming is for you,

### Learn by DOING

## Course Goals

### Programming for non-programmers

### Demonstrate Operations / Methods

### Enable / Energize new programmers

## The Instructor

### Ray Smith – A Longtime Programmer

## The Class – Using Zoom – Even in Person

### Emphasize Student Response

### Use Hand Raise for completion/understanding

# Session #1 Introduction - Touching on the topic

## Direction

### A program is a story.

### We’re telling the computer what to do.

### Python, like other programming languages is what we use.

### Don’t need to know everything about Python.

### Clarity is IMPORTANT.

### Ambiguity can be a problem.

* Learn by DOING

### Mistakes are OK.

## IDLE - Not just IDLE Curiosity

### Arithmetic

* + 60\*60\*24\*365
  + 99999999999999999999+1

### Test out language

### Use if a == 5 NOT if a = 5

### Find documentation

### turtle

### Python tutorial

## Samples:

Learn by Doing

### We’re going to start with some **simple** but **useful** examples.

### **Files -** A place to hold your story (program)

### With each program file, **YOU** and I are going to the following:

1. Find it
   1. **Open it**
   2. **Run it**
2. **Use it**
   1. **Save a copy**
   2. **Change the copy**
   3. **Save the copy with a NEW name**
   4. **Run changed program**

### **Hello World!**

* Traditional first program
* Biggest step is to get anything going
* Comments
* Print
* run, IDLE Shell
* Save As, change, Run

### **Don't be square.**

Turtle - a simple but powerful graphics (picture drawing) facility

### **square.py – first introduction to turtle**

* **from** *module\_name* **import** *name*
* **color(**string**), forward(***distance***), right(***angle***),**
* Save As, change, run

### **square\_loop.py – loops**

* Run, more than one way to …
* **for** *variable* **in** **range**(*number***):**
* **#** Comment at end of line

### Be a star... spokes.py

* **if** *test***:**

*do this*

*stuff*

* *elif test:*

*do this*

*stuff*

* else:

*do this*

*stuff*

* *Save As, change, run*

### Some input... some\_input.py

* Input – program can respond to user
* **inp**(*optional prompt to user*)
* **variable – named place to put stuff**
* ***print(****stuff to print, more stuff,…****)***
* **Ask address: Save As, change, run**

### Some Time…

* Lots of available tools for your use

sometime.py – demonstrates access and use of time and date

* import datetime, import time
* **time.sleep**(*number of seconds* to pause)

### A little flare... starry\_night.py

* ””” Multi-line

Commentary

”””

* List of things
  + primes = [2,3,5,7,11]
* **len**(list) – number of things in list
* tprime = primes[2] # start at zero

# Get third

* wrapped = primes[i%len(list)]

# % remainder / modulo

# 5%2 == 1

## Python vs English – a lot in common

How would you learn a s new spoken language?

To what purpose?

What’s in common?

What’s not?

### Same or Similar

### New or Different

Computer languages often give precise additional meanings to words or punctuation that have general use in the generally spoken or written language. This is similar to traffic signals giving special meanings to color, e.g., "red" – stop, "green" - go

### Examples

exercises/if\_statement/

exercises/while\_statement/

exercises/for\_statement/

## Course Project - Guessing Game – Twenty Questions

(with at least .5 hour to go in lecture session of class)

Little steps – Iterations

Just a loop

## Homework:

For Fun !

presentation/Introduction.../homework

# Session #2 Ideas, Tools, Functions

## Startup

Homework Poll

Questions

Programming, Python, Homework

## programming ideas

Like, Not Like

Programming Tools editors, IDE, common sense

Concepts e.g., Car Example

Similar Activities:

* Abstract/Indirect
* Physical

Build Once 🡺 use many times

lists append, range

samples/

lists1.py

list\_range.py

spokes\_list.py

saving\_interest.py

What’s Easy?

What’s Not?

Python Rules and Mechanics

More Python Tools

More on Lists

## Functions

### What and Why

#### Divide and Conquer

#### Benefits

### How functions work/used

Like a train with cars

* Calling code
  + chooses train by name
  + places values in the train cars
* Function code
  + Gets the values from the cars
  + Does calculation / operation
  + Returns result, if any

### How Functions are written

* def, comments, parenthesis, colon, body, return

### Function parameters

* Positional
  + In order
  + required
* keyword
  + keyword = value
  + optional
  + defaults

### Samples

samples/

spokes\_function.py

compare with spokes\_list.py

add

spoke(spoke\_len\*.5, colr, 20\*.5)

spoke(spoke\_len\*1.5, colr, 20\*1.5)

spokes\_function\_defaults.py

square\_positional.py

square\_keyword.py

square\_keywd\_none.py

Use None for more than simple default e.g., previous

print\_args.py – demonstrate print()’s keywords sep, end

Program Iterations Slide 48

Help from outside the lecture/class

From IDLE Help Docs:

The Python Tutorial - for more - needn't read all

IDLE: Help->Python Docs->The Python Tutorial

1. Whetting Your Appetite

Skip: 2. Using the Python Interpreter

3. An Informal Introduction to Python

4. More Control Flow Tools

5. Data Structures

lists, tuples,...

The Python Language Reference

- dry but the "final answer"

From Internet:

"Google" python...what you want to know

https://www.w3schools.com/python/

Summary

presentation/shapes\_arround.py

# Session 3: Functions Why and How

## Setup

poll How was the homework?

(Class 2 Homework - for Twenty Questions Project

Iteration 3:Say if guess is greater, less or equal)

review iteration 2, then work to iteration 3

function summary

samples/

shapes.py - function - points - to lines - to text

## Functions to develop ground work

Friends (names) – provide list with tools

What tools?

Storage - what's available?

Files available in IntroductionTo…\exercises\functions\friends\_family\simple\

Friends

Help students by FileExplorer copy friends\_family to my\_work/

* + 1. Name storage – just a list
    2. Simple access –adding, listing
    3. A simple listing function
    4. A place/file/module to hold the friends function(s)
    5. Testing the module
    6. Expand module
       1. Friend\_mod.py: adding set\_friends, add\_friend
       2. Friends\_mod\_use.py: Testing the module – with a shortened name

Thinking of Iterations - Show progress: Session 2 slide 49

## Help / info:

### Python Tutorial

IDLE->Help->Python Docs(F1)

Python Tutorial:

4.6. Defining Functions

4.7. More on Defining Functions

4.7.2. Keyword Arguments

### The Python Language Reference

### Online

https://www.w3schools.com/python

## Strings

functions/figures/my\_polygon\_keyw.py

functions/figures/my\_polygon\_keyw\_2.py

## Bouncing Ball – A Closer Look

# Session 4: Dictionary

## Setup

### go over previous session's homework:

iteration\_4.py - random target

exercises/functions/playing\_cards/cards\_mod.py

## Preview Sample – fireworks.py

## *Dictionary Atributes*

Dictionary vs list

## *do dictionary exercise(s)*

*trades\_d,...*

*Using file\_explorer,*

*copy exercises/dictionaries/ to my\_work/class\_4/*

*exercises/dictionaries/dictionary\_1.py*

*exercises/dictionaries/dictionary\_2.py*

## *do motion.py example*

* *Students should program along*
* *show initial board*
* *Students should play the game - move around*
* *Short look at motion.py*
  + *# top line comment!*
  + *Multiline “DOC” comment*
  + *cmds Note that the important thing here is the keys*
  + *Building board*
  + *Game loop*
    - *print\_board()*
    - *prompt for, accept command*
    - *check for bounds*
      * *report error*
      * *update board state*
* *What could be added / improved ?*

*change empty to something their own*

*e.g. +, " "*

*change board\_width e.g. 9*

## Sample Review

### File name: fireworks.py

### Overview

Graphic with expanding circles/balloons/fire works

Loops forever

* Circles are created randomly
* Circles die when hitting the edge

Looping is accomplished via turtle ontimer() function calls

### Major data structures:

**firework\_list**:

* list – list of firework dictionaries

firework – firework dictionary keys are firework attribues

* ‘x’ – location x-coordinate
* ‘y’ – location y-coordinate
* ‘size’ – size in pixels
* ‘size\_inc’ – size increment per time
* ‘color’ – color

Major sequence:

* Call fireworks\_start()
  + Fireworks\_start:
    - Call ontimer(fireworks\_update,…)
      * fireworks\_update:
        + Call firework\_update(fw)
        + Call firework\_display(fw)
  + As called…
    - firework\_update:
      * update this firework
      * check if at display edge
        + call firework\_create() to create new firework, replacing current firework
    - firework\_display:
      * display firework

# Session 5: Classes – Objects – Object Oriented

## Setup

go over previous session's homework:

iteration\_5.py - preamble with range

presentation/.../homework/solutions/move\_to\_goal.py

## Preview Sample

* square\_lattice.py
* square\_lattice\_varied.py

## Python Style

PEP 20 -- The Zen of Python

PEP 8 -- Style Guide for Python Code

## World is full of objects

### Objects can be:

### Held

### Used

### Moved

### Software Objects 🡸🡺 World Objects

* Record
* Model
* Control

Classes - Programmer Defined Objects / Data Types

Reasons / Usage

group data, group Activity

### Class / Object Example using turtle

/exercises/turtle/obj\_multiple.py - several turtles

turtle/obj\_multiple\_2.py - showing object independence

### Classes – Examples World Data/Action

### Modules – Sharing Code

From others e.g., turtle

Bring in turtle

whole module, parts

exercises/tuttle/obj\_multiple.py

exercises/turtle/obj\_multiple\_2.py

## Classes – Usage Details

Creation / Use

Definition

Predefined functions:

Like colors (red,green) in traffic lights

Have specific use

User defined functions:

Like arrow has it own purpose

Examples

class Person

Simplest:

### Class Example PersonClass

Person\_classes/

person\_1.py

person\_2.py

person.py

definition

code

classes/person\_classes/person.py

class PersonGroup

### Class Examples Ball2d, Ball2dTable

## 2D Billiard Ball Example

### Why and What Issues

### What Objects?

### Our Classes / Objects

### A bit about each Class

### A look in detail

## Optional – Class Inheritance

samples/rectangle\_simp.py, square\_simp.py

samples/rectangle.py, square.py

class Ball2d

class Ball2dTable

# Session 6: Files

go over previous session's homework:

iteration\_6.py - multiple games

These sessions 6 and 7 are purposely light

Please open questions about what you are confused, unsure, interested in.

Files - Came into use:

1. because computers could not hold all the data

2. data needed to stay after program completed

evolved from physical file cabinets

card decks, punched tape -> magnetic tape, disk -> internet cloud

Early computer work required detailed knowledge of structure

Now - mostly the idea

accessible by name

often created or used by other programs

stays around after program is done

usually linear often only read or write

often for BIG data

Your python programs are examples

created by you with a text editor

used by you via python program compiler

stays around for future access

review

file\_search\_graphical.py - top level search

graphical\_text.py - object for graphics

file\_search\_graphical\_non\_obj.py - old style

# Session 7: Graphics

go over previous session's homework:

1. iteration\_7.py - handle typos - illegal input

Please open questions about what you are confused, unsure, interested in.

2. presentation/homework/solutions/file\_search.py

a. web search for python find string in string

b. IDLE doc string module -> str

go over simple graphics

Introduce debugging

# Programming / Python Tools

comments

arithmetic

numbers - int, float

strings

variables constants

flow control

decisions

looping

grouping

program files

iteration

functions (subroutines, procedures)

lists (arrays)

dictionaries (stuctures)

classes (stuctures)

## Putting together the files for class distribution

### Goals

Provide a reasonable list of examples for lecture and independent student use.

Facilitate quick use for lectures

Limit need for many files

### File organization

To facilitate delivery and setup of the class files the files will be compressed into a single file **prog\_intro.zip**

**prog\_intro**: file folder to be placed on DeskTop Folder. The following subfolders are contained:

**exercises**: file folder to hold many programming exercise. Some topics are themselves organized into further subfolders.

**presentation**: subfolder, the contents being used to drive the course presentation. Contains a sub folder for each class session. Each class session folder contains the folding:

***Lecture Slides*:** A Microsoft PowerPoint file used to guide the lecture

**samples:** folder containingPython program files to introduce the session’s topic

**homework:** folder supporting homework to demonstrate concepts presented during this session (Aways OPTIONAL)

***Homework\_description:*** *PowerPoint explanation file.*

**solutions**: folder to contain our solutions, if not present in other locations, e.g., **exercises**. folder

**my\_work**: empty file folder to be filled up by the student, first with unchanged copies of lecture files, then by their own modifications and additional work

**Cheat\_Sheets**: holds examples of quick guides to Python