GAME PROGRAMMING IN PYTHON

- A STEM INSPIRED APPROACH -

Student Handout

Your name

GAME PROGRAMMING IN PYTHON

- A STEM INSPIRED APPROACH –

NOVEMBER 11, 2015

Gavriel Feuer, Ph.D. candidate
gavriel.feuer@downstate.edu

Niklas Manz, Ph.D.
Department of Physics
College of Wooster
nmanz@wooster.edu

Copyright ©2014 by the authors.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherw ise, without the written permission of the authors.

Table of Contents

List of Figures 4
List of Tables 5
Getting Started 6
Useful Links and Books 7
Topic 1 – Introduction 8
Topic 2 – Color 12
Topic 3 – Motion 18
Topic 4 – Sounds 22
Topic 5 – Computer hardware 28
Topic 6 – `If` Statement 35
Topic 7 – `While` and `For` Loop 41
Topic 8 – Event Handling 46
Topic 9 – Images 48
Topic 10 – Classes and Sprites 51
Python Commands Cheat Sheet for tasks 53
Game code template 56
Source codes 58
draw_player.py 58
coordinate.py 60
bounce_player.py 62
bounce_player_sound.py 64
play_tone.py 66
if_statements.py 67
while_for_loops.py 68
move_player.py 69
bitmapped_player.py 72
move_player_sprite.py 75
move_player_sprite2.py 78
pong.py 81
pygame2exe.py 84

Copyright © Gavriel Feuer and Niklas Manz, 2014

List of Figures

Figure 1: The additive RGB color system (R=red, G=green, B=blue, Y=yellow, M=magenta, C=cyan, W=white). 12
Figure 2: The subtractive CMYK color system (C=cyan, M=magenta, Y=yellow, R=red, G=green, B=blue, K=key(black)). 13
Figure 3: Sketch of the relationship between bits and bytes. 13
Figure 4: A selection of the 15 simplest RGB colors, their names, and 8-bit channel decimal code. 15
Figure 5: A selection of 11 RGB colors, their names, 8-bit channel decimal code, and 8-bit channel hexadecimal codes. 15
Figure 6: Examples of coordinate systems. Left: mathematical. Right: computer screen. 18
Figure 7: Start and end of the sine animation on wikipedia. 22
Figure 8: Bar diagram of the approximate hearing range of various mammals. 23
Figure 9: Cross-sectional diagram of the human ear. The length of the auditory canal is exaggerated in this image. (This image is licensed under the Creative Commons Attribution 2.5 Generic license.) 23
Figure 10: Mechanics of the amplification-effect of the middle ear. (This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.) 24
Figure 11: The position x of the maximal amplitude of the travelling wave corresponds in a 1-to-1 way to a stimulus frequency. (This file is licensed under the Creative Commons Attribution 2.5 Generic license.) 24
Figure 12: Frequency- and loudness-dependence of human hearing loss. (This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.) 25
Figure 13: Picture of three types of desktop RAM. 31
Figure 14: Flow chart of a simple `if` statement. 37
Figure 15: Flow chart of an `if` statement with optional `else` parameter. 37
Figure 16: Flow chart of an `if` statement with optional `elif` and `else` parameters. 38
Figure 17: Flow chart of a `for` loop (rewritten as a `while` loop) 41
Figure 18: Flow chart of a simple `while` loop. 42

Copyright © Gavriel Feuer and Niklas Manz, 2014

List of Tables

Table 1: Color notation of the pygame code when using the IDLE editor. 8
Table 2: A few useful key combinations. 8
Table 3: How to start a pygame code and how to interpret a log file. 10
Table 4: Student tasks for the introduction. 10
Table 5: The power symbols of the International Electrotechnical Commission (IEC). 14
Table 6: Student tasks for the color topic. 16
Table 7: Student tasks for the motion session. 20
Table 8: Student tasks the sound topic. 27
Table 9: Examples of the main parts of a desktop computer. 30
Table 10: Examples of the differences between laptop and PC hard drives and RAM. 30
Table 11: Prefix definitions. 31
Table 12: Decimal and Binary prefix values. 32
Table 13: Two tables from Leibniz's original publication in 1703. 33
Table 14: Comparison operators. 35
Table 15: Student tasks for an If statement session. 35
Table 16: Syntax of the `if` statement. 36
Table 17: Student tasks for the `if` statement session. 39
Table 18: Syntax of the `for` and `while` loop. 41
Table 19: Syntax comparison of while, for, and if across five different computer languages. 43
Table 20: Comparison of saying "Good Morning" in seven different languages of two language families. 44
Table 21: Student tasks for the loop session. 44
Table 22: First student tasks for the event handling session. 46
Table 23: Second student tasks for the event handling session. 46
Table 23: Image file formats. 48
Table 24: Parameter of a SUNY Downstate logo (300 x 278). 48
Table 25: Student tasks for the image topic. 49
Table 27: Student tasks for the classes and sprites topic. 51

Copyright © Gavriel Feuer and Niklas Manz, 2014

Getting Started

To work with this curriculum, you need to install a few free programs and python libraries. Depending on your operating system, Microsoft Windows or Linux based or the Raspberry Pi, you need to download different files.

1. Install python version 2.7 http://www.python.org/download
2. Install pygame http://www.pygame.org/download.shtml
3. Install numpy https://pypi.python.org/pypi/numpy
4. Install the IDLE editor http://sourceforge.net/projects/pyidlelif/
5. Install the simple graphics program PINTA http://pinta-project.com

In Ubuntu, with internet access, you can type, for example, “sudo apt-get install pygame” into the terminal window to install the missing programs.

On a Raspberry Pi, Python, pygame, and the IDLE editor are pre-installed. But if you face any problems, please go to

http://learn.adafruit.com/adafruit-raspberry-pi-lesson-1-preparing-and-sd-card-for-your-raspberry-pi

Copyright © Gavriel Feuer and Niklas Manz, 2014

Useful Links and Books

The Python web page:	http://www.python.org
A Python web site for kids:	http://nostarch.com/pythonforkids
An excellent Python tutorial:	http://www.petercollingridge.co.uk/book/export/html/6
The pygame tutorial:	http://www.pygame.org/wiki/tutorials
The Invent with Python blog:	http://inventwithpython.com/blog/2010/09/01/the-top-10-pygame-tutorials

A variety of Arcade Games with Python and Pygame:	http://programarcadegames.com/
Think Python: How to Think Like a Computer Scientist:	http://www.greenteapress.com/thinkpython

2D Game Physics with Pygame:

http://physics.gac.edu/~miller/jterm_2013/physics_engine_tutorial.html

Python for beginners with many easy examples and cheat sheets:

http://www.pythonforbeginners.com/

A video course from the Khan Academy on Programming in Python:
https://www.khanacademy.org/science/computer-science-subject

Pygame 1.5.5 Reference Manual: http:// pygame.org/ftp/contrib/pygame_docs.pdf

Python for Kids: A Playful Introduction to Programming by Jason R. Briggs. December 2012, 344 pp. ISBN: 978-1-59327-407-8

Code.org is a non-profit foundation devoted to the vision that every student in every school should have the opportunity to learn how to code.

http://code.org/

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 1 – Introduction

Some short definitions from Wikipedia:

1. **Python** is a widely used general-purpose, high-level programming language.

2. **Pygame** is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.

3. **IDLE** is an integrated development environment for Python. IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.
   According to the included README, its main features are:
   - Multi-window text editor with syntax color highlighting, auto-completion, smart indent and other.
   - Python shell with syntax highlighting.
   - Integrated debugger with stepping, persistent breakpoints, and call stack visibility.

Color	Example	What
black	`position_x = position_x + speed_x`	Code
red	`#----------Main Program Loop ----------`	Comment
green	`pygame.display.set_caption("Retro Screensaver")`	String (plain text)

orange	`while rungame:`
	`  for event in pygame.event.get():`
	`  if event.type == pygame.QUIT:`
		Keyword (classes, loops, and conditional statements)

blue	`rungame=False`	Definition

Table 1: Color notation of the pygame code when using the IDLE editor.

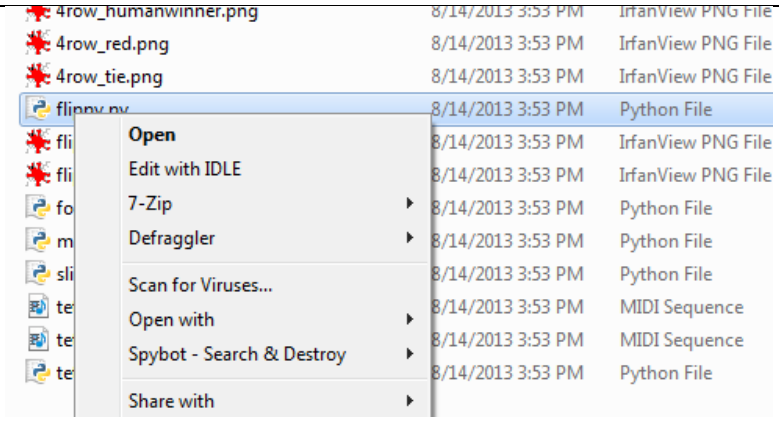
And a few useful key combinations:

Shift *arrow*	Highlight a part of the code.
Ctl c	Copy a highlighted part of the code into the clipboard.
Ctl v	Paste the clipboard into the code.
Ctl x	Cut (copy and delete) a highlighted part of the code.
Ctl z	Undo previous changes in the code.

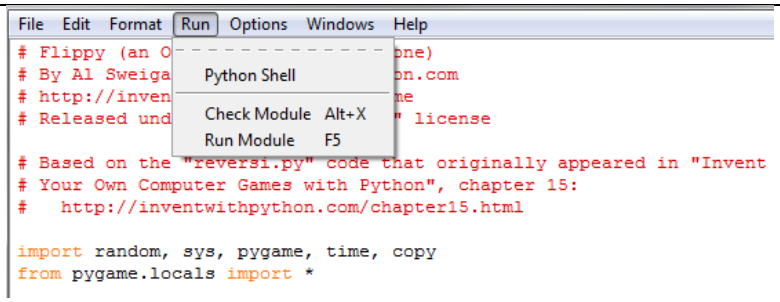
Table 2: A few useful key combinations.

Copyright © Gavriel Feuer and Niklas Manz, 2014

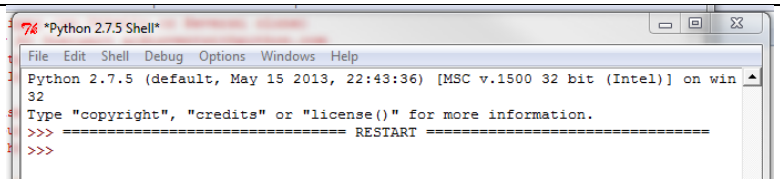
Go to a specific pygame code, right-click on it, and click on 'Edit with IDLE' to open the game code in the IDLE editor.



Go to 'Run' and click on 'Run Module F5'.



You will always get the Python Shell as a log. If the game does not start, you will find helpful error logs.



If you, for example, comment out the definitions for the color white "#WHITE =(255,255,255)", you will get the following log output.

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
>>> ================================ RESTART ================================
>>>

Traceback (most recent call last):
  File "C:\Users\nmanz\Dropbox\nmanz\STEM-Mentoring\GameProgrammingInPython\Fini
shed Code\Intro Games\flippy.py", line 40, in <module>
    TEXTCOLOR = WHITE
NameError: name 'WHITE' is not defined
>>>
```

Table 3: How to start a pygame code and how to interpret a log file.

Introduction Tasks	
1.	Define a new color.
2.	Change the color of one parts of the object.
3.	Change the line width of the circle surrounding the figure.
4.	Change the line width of the legs or arms of the figure.
5.	Reposition the object (or parts of it to create, for example, a headless figure).
6.	Change the window dimensions

Table 4: Student tasks for the introduction.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 1

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 2 – Color

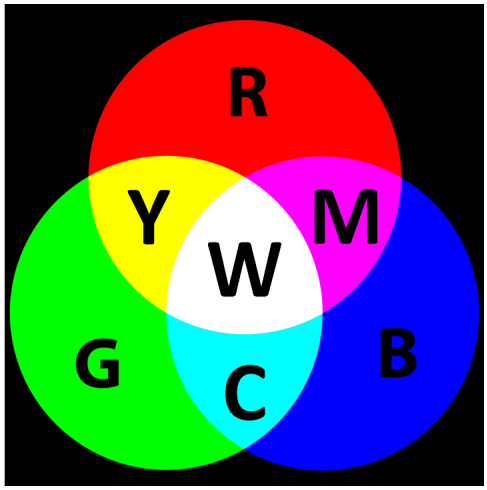


Figure 1: The additive RGB color system (R=red, G=green, B=blue, Y=yellow, M=magenta, C=cyan, W=white).

The RGB color system

- The **RGB color system**, constructs all colors from a combination of red, green, and blue.
- Each **pixel** in the **liquid-crystal display** (LCD) of the monitor contains three **light emitting diodes** (LED), one for each color of the RGB color system. Visible colors on the screen are created by intensity combinations of red, green, and blue.
- When the red pixel is set to 0, the red LED is turned off. When the red pixel is set to 255, the red LED is turned fully on. Any value between them sets the LED to partial light emission.
- Turning two colors off and one color on leaves red, green, or blue.

Copyright © Gavriel Feuer and Niklas Manz, 2014

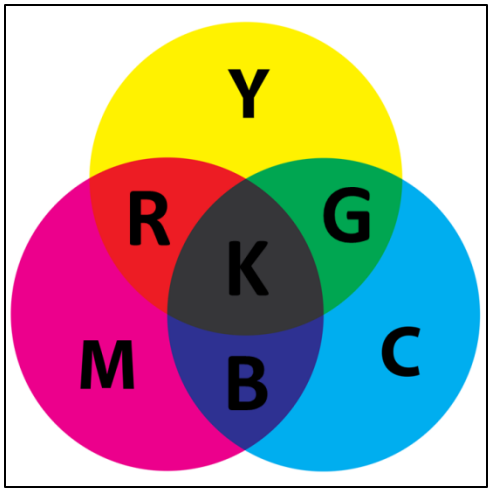


Figure 2: The subtractive CMYK color system (C=cyan, M=magenta, Y=yellow, R=red, G=green, B=blue, K=key(black)).

The bit:

- Smallest unit of information in computing.
- Binary integer value of either "0" or "1", therefore, having two possibilities.
- The term *bit* is a contraction of binary digit.

The byte:

- Smallest addressable unit of memory in many computer architectures.
- One byte most commonly consists of eight bits.

Why is 1 byte = 8 bit = 256 possibilities?

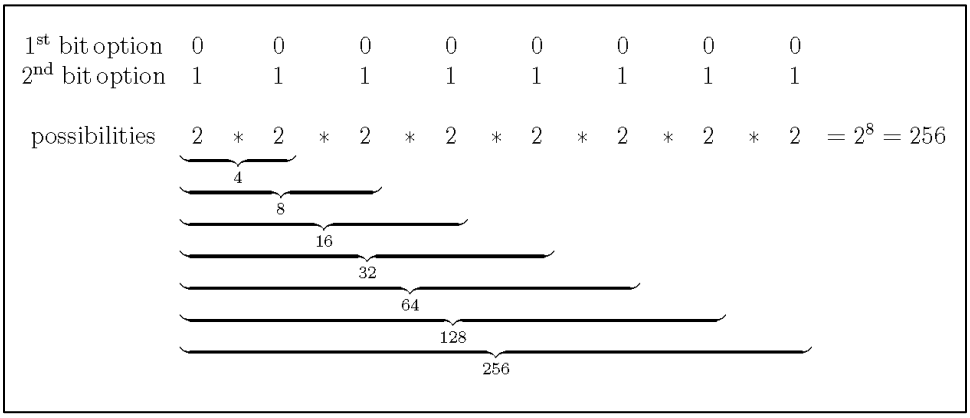


Figure 3: Sketch of the relationship between bits and bytes.

Copyright © Gavriel Feuer and Niklas Manz, 2014

- The red, green and blue use **8 bits** each, which have **integer** values from 0 to 255. This makes 256*256*256 = 16,777,216 possible colors.

symbol	use	description
	The **power on** symbol	Pressing this button or end of a toggle switch sets the equipment into the 'power on' state.
	The **power off** symbol	Pressing this button or end of a toggle switch sets the equipment into the 'power off' state.
	The **power on-off** symbol	Pressing this button switches the equipment between the 'power on' and 'power off' state.
	The **standby** symbol	Pressing this button sets the equipment into a 'low power' or 'sleep mode'. It does not disconnect the equipment from the power supply. This means that there is still a current flowing through the equipment which uses energy and costs money. **Therefore, if you do not need the equipment for a while turn the device off, not just into sleep mode!**

Table 5: The power symbols of the International Electrotechnical Commission (IEC).

Copyright © Gavriel Feuer and Niklas Manz, 2014

Color	Name	RGB triplet (8-bit channel decimal code) [R,G,B]
	Black	(0,0,0)
	White	(255,255,255)
	Red	(255,0,0)
	Green	(0,255,0)
	Blue	(0,0,255)
	Yellow	(255,255,0)
	Cyan	(0,255,255)
	Magenta	(255,0,255)
	Maroon	(128,0,0)
	Dark Green	(0,128,0)
	Navy	(0,0,128)
	Olive	(128,128,0)
	Teal	(0,128,128)
	Purple	(128,0,128)
	Gray	(128,128,128)

Figure 4: A selection of the 15 simplest RGB colors, their names, and 8-bit channel decimal code.

Color	Name	RGB triplet (8-bit channel decimal code) [R,G,B]	RGB triplet (8-bit channel hexadecimal code) [#RRGGBB]
	Black	(0,0,0)	#000000
	White	(255,255,255)	#FFFFFF
	Red	(255,0,0)	#FF0000
	Green	(0,255,0)	#00FF00
	Blue	(0,0,255)	#0000FF
	Yellow	(255,255,0)	#FFFF00
	Cyan	(0,255,255)	#00FFFF
	Magenta	(255,0,255)	#FF00FF
	Maroon	(128,0,0)	#800000
	Dark Green	(0,128,0)	#008000
	Navy	(0,0,128)	#000080

Figure 5: A selection of 11 RGB colors, their names, 8-bit channel decimal code, and 8-bit channel hexadecimal codes.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Color Tasks	
1.	Create (define) a new color variable with a new color.
2.	Use the newly defined color as the color for the circle.
3.	Create a second new color variable.
4.	Use the second new color as the background color.
5.	Define a new color variable setting two of the three numbers of the number triplet to “0” and one less than 255. Use this color somewhere for your figure and explain the difference to the ‘original’ color with two “0”s and one “255”.

Table 6: Student tasks for the color topic.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 2

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 3 – Motion

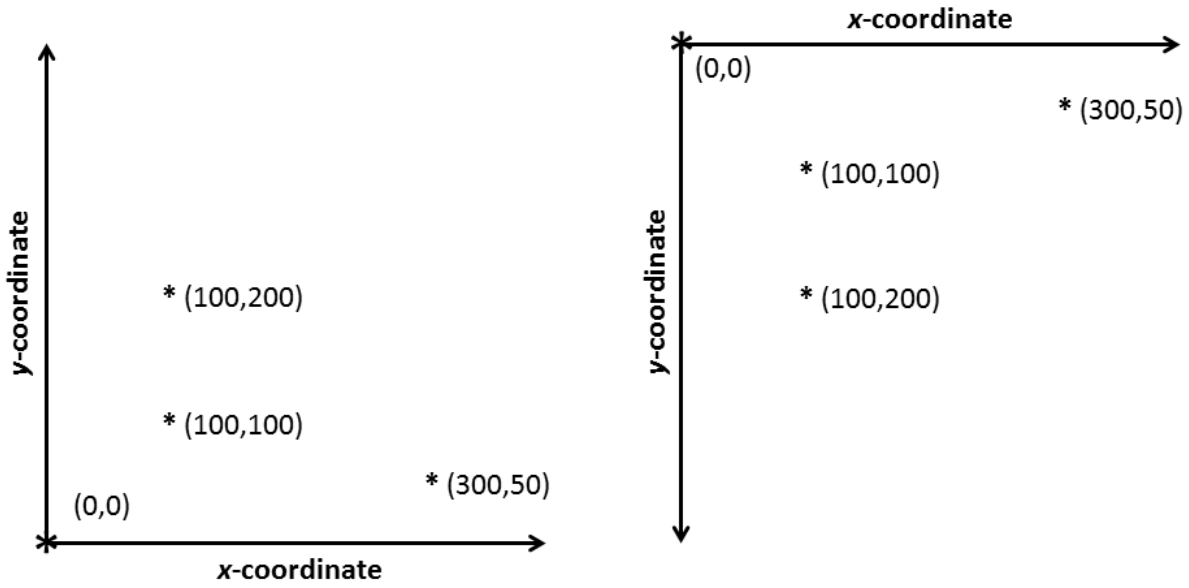


Figure 6: Examples of coordinate systems. Left: mathematical. Right: computer screen.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Motion Tasks		
1.	velocity:	Let the object fly initially horizontally to the right.
2.		Let the object fly initially horizontally to the left.
3.		Let the object fly initially vertically up.
4.		Let the object fly initially vertically down.
5.		Let the object fly initially horizontally to the right and twice as fast as before.
6.		Let the object fly initially diagonally up.
7.	bouncing:	Comment out the segment of code that controls the bouncing effect. Run the code and observe the effect.
8.		Change the bouncing condition that the speed increases with each bouncing.
9.		Change the bouncing condition: object bounces back when center of circle

Copyright © Gavriel Feuer and Niklas Manz, 2014

		hits the wall not the circle itself.

Table 7: Student tasks for the motion session.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 3

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 4 – Sounds

The guitar video on YouTube (http://www.youtube.com/watch?v=TKF6nFzpHBU)

The sine function on Wikipedia (http://en.wikipedia.org/wiki/Sine)

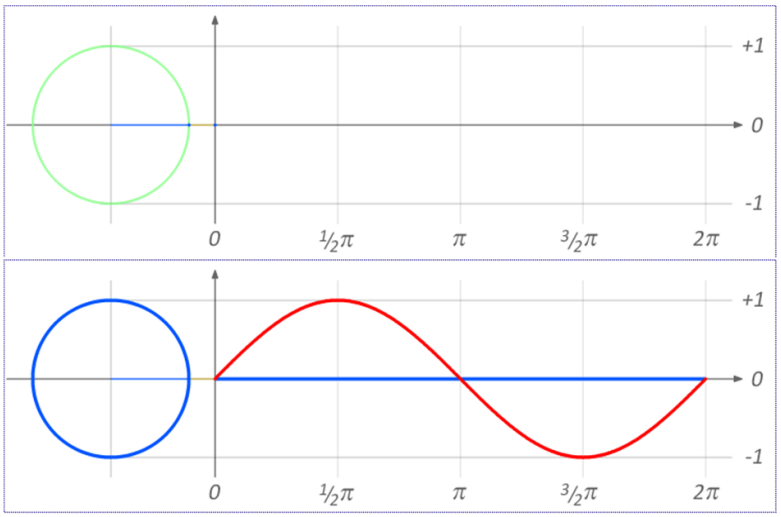


Figure 7: Start and end of the sine animation on wikipedia.

Copyright © Gavriel Feuer and Niklas Manz, 2014

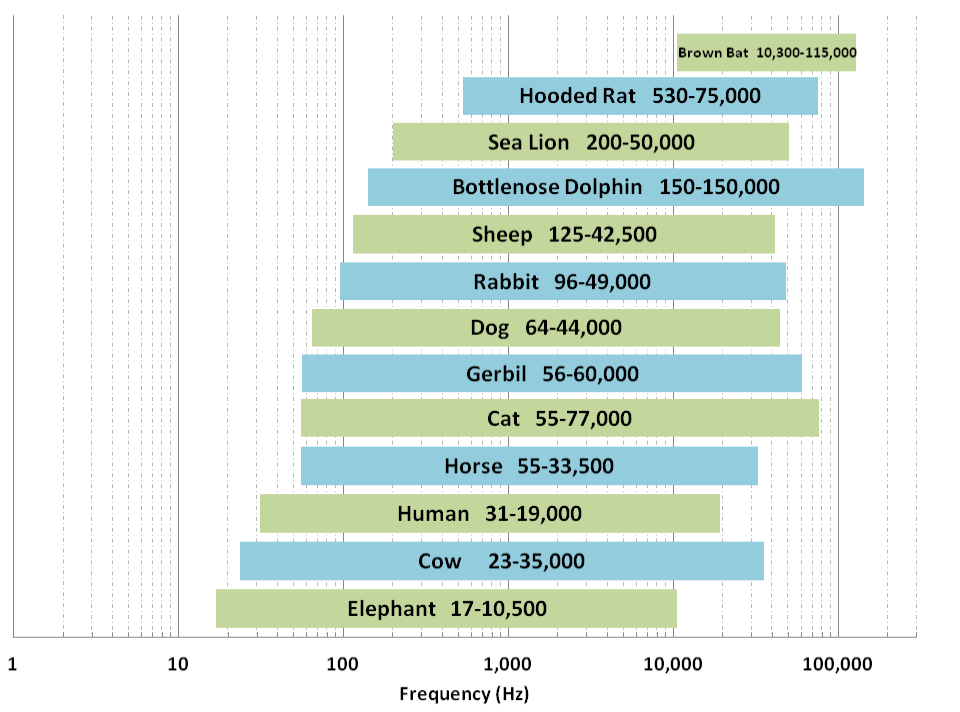


Figure 8: Bar diagram of the approximate hearing range of various mammals.

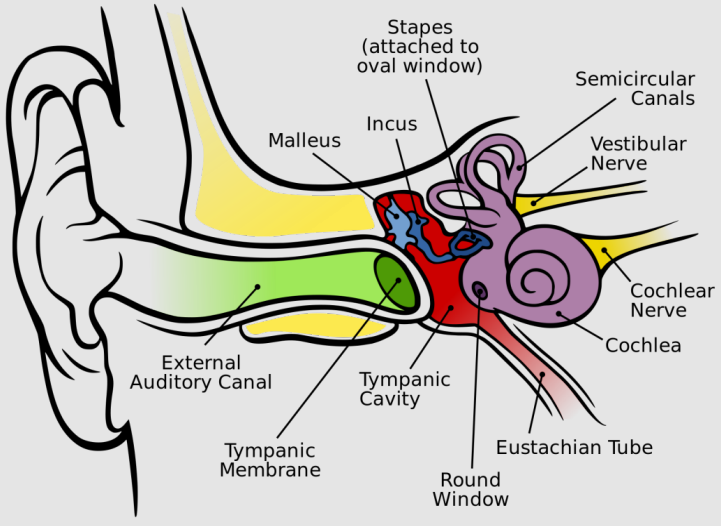


Figure 9: Cross-sectional diagram of the human ear. The length of the auditory canal is exaggerated in this image. (This image is licensed under the Creative Commons Attribution 2.5 Generic license.)

Copyright © Gavriel Feuer and Niklas Manz, 2014

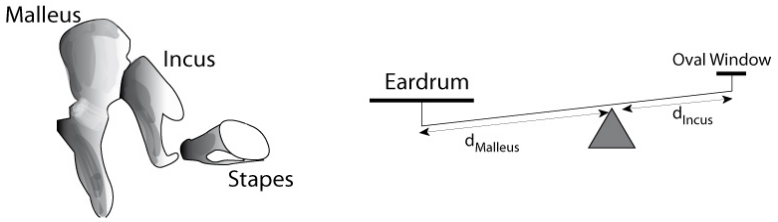


Figure 10: Mechanics of the amplification-effect of the middle ear. (This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.)

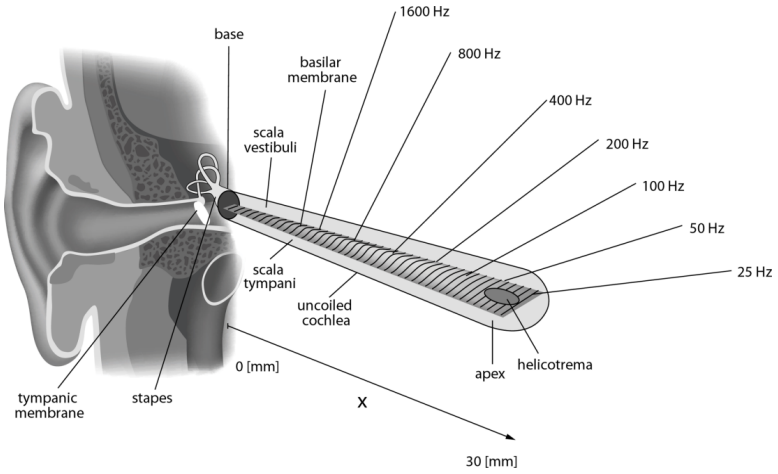


Figure 11: The position x of the maximal amplitude of the travelling wave corresponds in a 1-to-1 way to a stimulus frequency. (This file is licensed under the Creative Commons Attribution 2.5 Generic license.)

Copyright © Gavriel Feuer and Niklas Manz, 2014

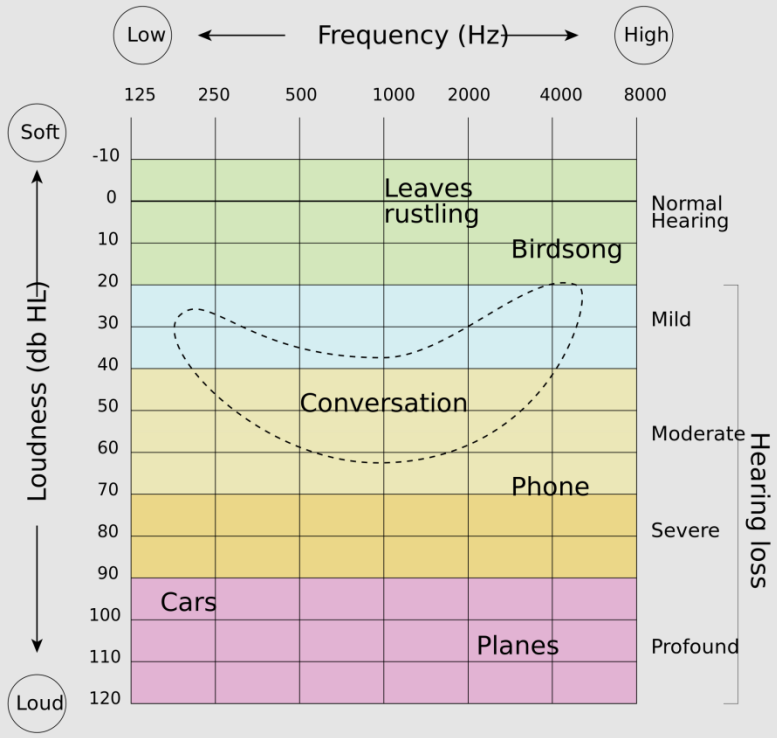


Figure 12: Frequency- and loudness-dependence of human hearing loss. (This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.)

A few resources about sound and hearing:

- The 'Physics classroom' on sound:
  - http://www.physicsclassroom.com/class/sound/
- The ear and hearing range at 'Science and Engineering Encyclopedia:
  - http://www.diracdelta.co.uk/science/source/e/a/ear/source.html
  - http://www.diracdelta.co.uk/science/source/h/e/hearing range in animals/source.html

Copyright © Gavriel Feuer and Niklas Manz, 2014

Sound Tasks	
1.	Open the python code bounce_player_sound.py. There are a few sounds available in the sound folder. Load a **new sound** file for the bouncing effect.
2.	Create a new sound variable with a different tone. Use this sound variable to play a different tone if the figure hits the upper and lower wall.

Copyright © Gavriel Feuer and Niklas Manz, 2014

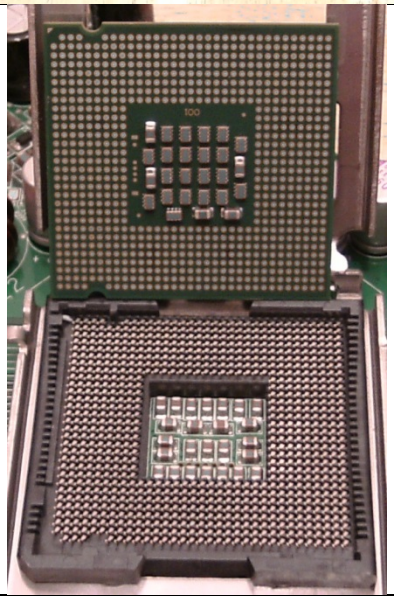
3.	Open the simple python code **play_tone.py**. Change the **frequency** entry for the one-tone sound to obtain a lower tone.
4.	Change the **amplitude** entry for the one-tone sound and observe the change.
5.	Create two tones with frequencies f_1 and f_2. Play the two tones separately in two runs.
6.	Use the two tones with frequencies f_1 and f_2. Create a new tone by **adding the two frequency** values ($f_3=f_1+f_2$). Play the third tone.
7.	Use the two tones with frequencies f_1 and f_2. Create a new tone (sound) by **adding the two tones** into one sound. Play the composite sound.
8.	Repeat tasks 6 and 7 but use multiplication instead of addition.

Table 8: Student tasks the sound topic.

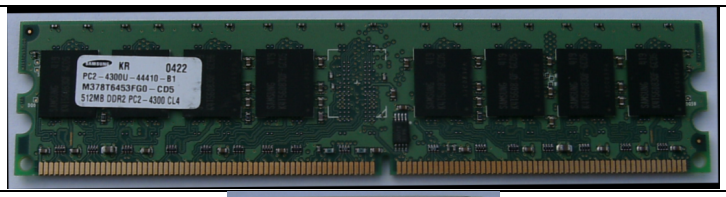
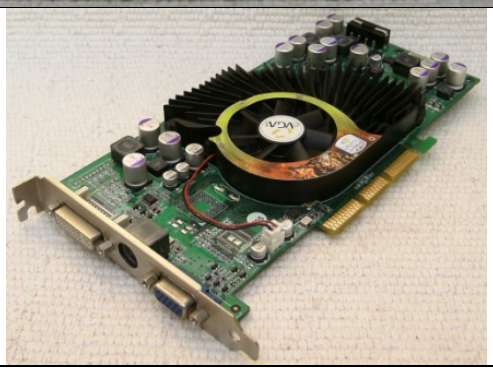
Notes for Topic 4

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 5 – Computer hardware

power supply (with various connectors)	
mother board, main board	
CPU (central processing unit, top) with CPU socket (bottom)	

Copyright © Gavriel Feuer and Niklas Manz, 2014

RAM (random-access memory)	
hard drive	 
CD/DVD drive	
video card, graphics card (with digital DVI, S-Video, and analog VGA outputs)	

Copyright © Gavriel Feuer and Niklas Manz, 2014

ethernet card, network card		
sound card		

Table 9: Examples of the main parts of a desktop computer.

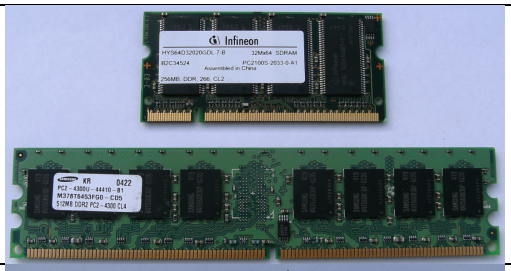
RAM (top: laptop; bottom: PC)		
hard drive (left: laptop; right: PC)		

Table 10: Examples of the differences between laptop and PC hard drives and RAM.

Copyright © Gavriel Feuer and Niklas Manz, 2014

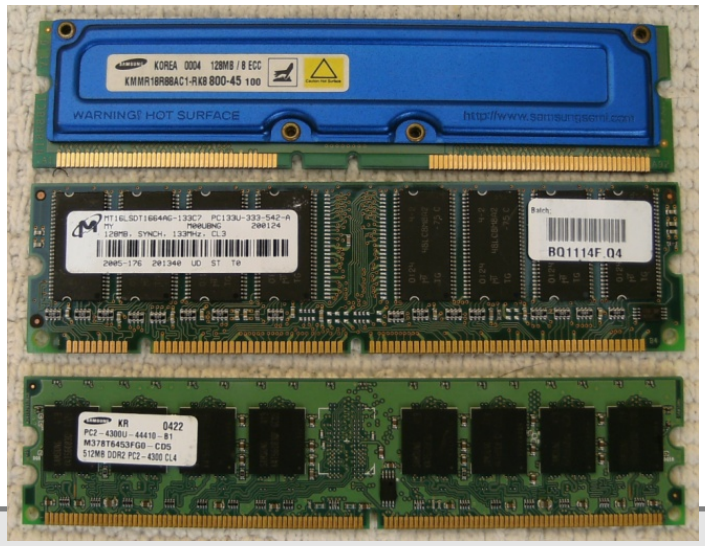


Figure 13: Picture of three types of desktop RAM.

The metric prefixes used in computer parts.

Prefix	Symbol	1000^n	10^n	Decimal	Word
peta	P	1000^5	10^{15}	1,000,000,000,000,000	quadrillion
tera	T	1000^4	10^{12}	1,000,000,000,000	trillion
giga	G	1000^3	10^9	1,000,000,000	billion
mega	M	1000^2	10^6	1,000,000	million
kilo	k	1000^1	10^3	1,000	thousand
		1000^0	10^0	1	one

Table 11: Prefix definitions.

Binary prefixes prepend the units of digital information in computing (bits and bytes), which have a base of 1,024 instead of 1,000. The RAM capacity of, for example, 1 MB equals 1,048,576 bytes, because it is 1024^3. But in most other cases, for example for hard drives, decimal prefixes are used. A 1 MB hard drive has a capacity of 1,000,000 = 1000^3 bytes. Therefore, kibibyte (kiB), mibibyte(MiB), and gibibyte (GiB) were introduced in 1998 to indicate 1,024 bytes (1024^1), 1,048,576 bytes (1024^3), and 1,073,741,824 bytes (1024^6), respectively.

Decimal			Binary		
1000^n	Prefix	Symbol	1024^n	Prefix	Symbol
1000^5	peta	P	1024^5	pebi	Pi
1000^4	tera	T	1024^4	tebi	Ti
1000^3	giga	G	1024^3	gibi	Gi

Copyright © Gavriel Feuer and Niklas Manz, 2014

1000^2	mega	M	1024^2	mebi	Mi
1000^1	kilo	k	1024^1	kibi	Ki

Table 12: Decimal and Binary prefix values.

Leibniz's work on the binary system from 1703: The first talks about the recently decrypted oldest decimal multiplication table, the second reports about the use of the binary system 300 years before Leibniz published his notation, and the third is an English translation of the publication, which was written in French

1. Ancient times table hidden in Chinese bamboo strips. The 2,300-year-old matrix is the world's oldest decimal multiplication table.
   http://www.nature.com/news/ancient-times-table-hidden-in-chinese-bamboo-strips-1.14482

2. Polynesian people used binary numbers 600 years ago. Base-2 system helped to simplify calculations centuries before Europeans rediscovered it.
   http://www.nature.com/news/polynesian-people-used-binary-numbers-600-years-ago-1.14380
   Bender, A. and Beller, S. (2014). Mangarevan invention of binary steps for easier calculation. *Proc Natl Acad Sci U S A*. PMID: 24344278.

3. Gottfried Wilhelm Leibniz's Explanation of binary arithmetic from 1703. An English translation from 2007 of his publication (written in French) can be found at
   http://www.leibniz-translations.com/binary.htm
   The following table contains two images of the original 1703 publication:
   Explication de l'arithmétique binaire, qui se sert des seuls caractères O et I; avec des Remarques sur son utilité, et sur ce qu'elle donne le sens des anciennes figures Chinoises de Fohy [Explication of the binary arithmetic, which uses only the characters O and I; with remarks on its usefulness and on the light it throws on the ancient Chinese figures of Fohy].

Copyright © Gavriel Feuer and Niklas Manz, 2014

Académie Royale des Sciences 1703:85–89.
http://ads.ccsd.cnrs.fr/docs/00/10/47/81/PDF/p85_89_vol3483m.pdf

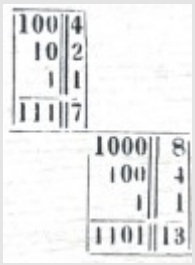


Table explaining the binary numbers for 0 to 32.	Two addition examples.

Table 13: Two tables from Leibniz's original publication in 1703.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 5

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 6 – `If` Statement

Variables and lists are defined using the '=' symbol. Therefore, it can't be used to check the equality of numbers. So, it's a good time to introduce the comparison operators.

Math symbols	Python symbols	Meaning
=	==	equal
≠	!=	not equal
<	<	strictly less than
>	>	strictly greater than
≤	<=	less than or equal
≥	>=	greater than or equal

Table 14: Comparison operators.

At the computers, students are now asked to perform various tasks to understand the syntax of the `range` and `print` commands, and the symbols and syntax of the comparison operators in Python.

If statement session Tasks 1	
1.	Use the range command to display the numbers 5-12 (i.e. [5, 6, 7, 8, 9, 10, 11, 12]).
2.	Define two variables. Compare them using a comparison operator to obtain a True response.
3.	Define a list with items ['a','b','c']. Display the list with the print command.

Table 15: Student tasks for an If statement session.

Copyright © Gavriel Feuer and Niklas Manz, 2014

`if` *statement*	`if` <argument>: <command> `elif` < argument >: <command> `else`: <command>	*if test* *optional elif's* *optional else*

Table 16: Syntax of the `if` statement.

Examples of simple `if` statements:

- ```
  if TRUE:
        print ("Hello World")
  ```
- ```
  A = 5
  B = 10
  if A < B:
        print ("A is less than B")
  ```
- ```
  A = 5
  B = 10
  if A < B:
        print ("A is less than B")
  else:
        print ("A is not less than B")
  ```
- ```
  A = 5
  B = 10
  if A < B:
        print ("A is less than B")
  elif A > B:
        print ("A is larger than B")
  else:
        print ("A is equal to B")
  ```
- ```
  Name_A = raw_input("What is your name?")
  Name_B = raw_input("What is your partner's name")
  Age_A = input("What is your age:")
  Age_B = input("What is your partner's age:")
  if (Age_A < Age_B):
        print (Name_A, "is younger than", Name_B)
  elif (Age_A > Age_B):
        print (Name_A, "is older than", Name_B)
  ```

Copyright © Gavriel Feuer and Niklas Manz, 2014
  ```

```
else:
      print (Name_A, "and", Name_B, "have the same age")
```

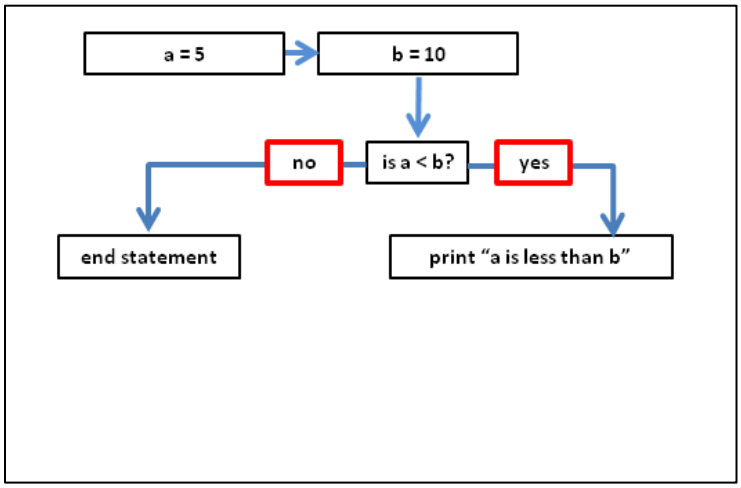


Figure 14: Flow chart of a simple `if` statement.

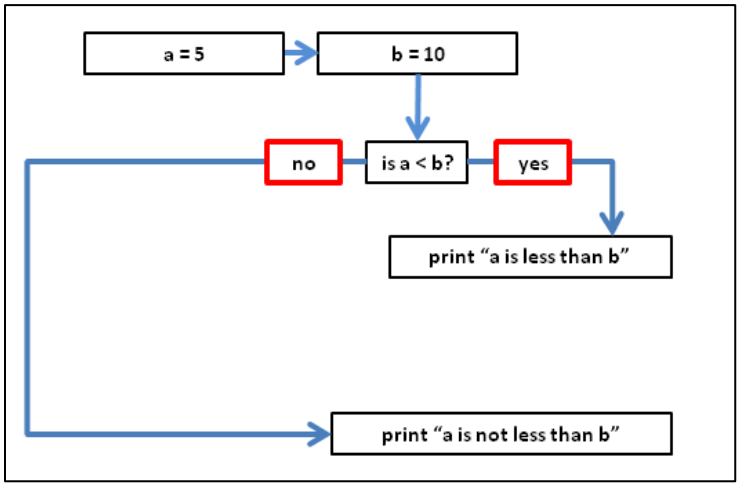


Figure 15: Flow chart of an `if` statement with optional `else` parameter.

Copyright © Gavriel Feuer and Niklas Manz, 2014

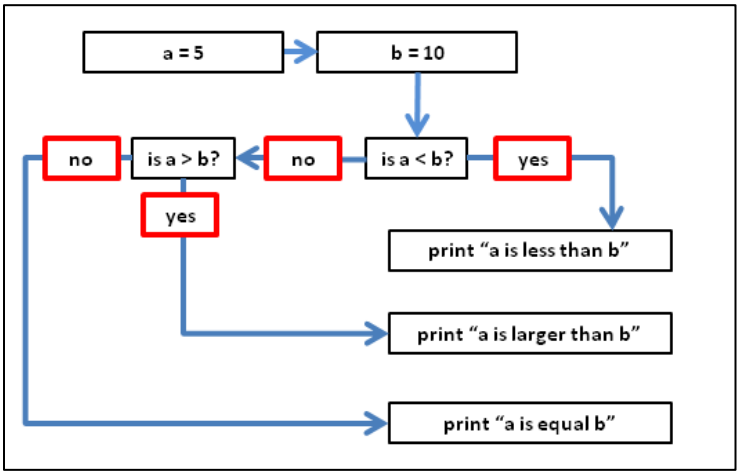


Figure 16: Flow chart of an `if` statement with optional `elif` and `else` parameters.

If statement session Tasks 2	
1.	Define two variables. Test whether the first variable is larger than the second variable. Print the result.
2.	Define two variables. Test whether the first variable is larger than the second variable. Print a different result when it's true than when it's not true.
3.	Define two variables using the `input` command. Test whether the first variable is larger than the second variable. Print the result.
4.	Ask for two strings (your names) with the `raw_input` command. Define two age variables using the `input` command. Test whether the first variable is smaller than the second variable. Print a correct statement if true and another if not true.
5.	Open move_player.py in the IDLE editor. Scroll down to the #"Game Logic"# section. Edit the `if` statement so that the background changes colors when

Copyright © Gavriel Feuer and Niklas Manz, 2014

	the player hits the bottom wall.
6.	Open move_player.py in the IDLE editor. Scroll down to the #"Game Logic"# section. Edit the `if` statement so that when the player hits the right wall, place the figure in the center of the screen.

Table 17: Student tasks for the `if` statement session.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 6

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 7 – `While` and `For` Loop

The basic structure of the while and for loop:

`for` *loop*	`for` <target> `in` <object>: <command>	assign object items to target
`while` *loop*	`while` <argument>: <command>	*loop test*

Table 18: Syntax of the `for` and `while` loop.

Examples of simple `for` loops:

- ```
  for i in range(10):
      print (i)
  ```
- ```
  for i in range(2,10):
      print(i)
  ```
- ```
  for i in range(5,20,2):
      print (i)
  ```
- ```
  list=['a','b','c']
  for item in list:
      print item
  ```

For loops can also be understood as switches when 'rewritten' as while loops. The following is a flow chart of such a loop.

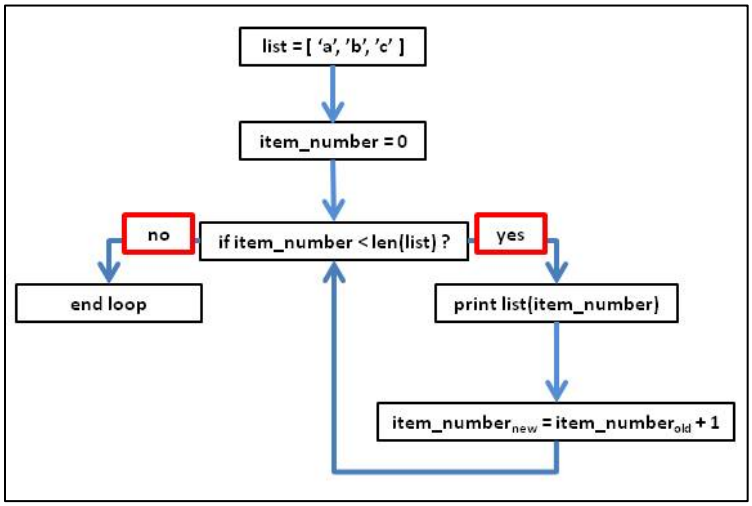


Figure 17: Flow chart of a `for` loop (rewritten as a `while` loop).

Copyright © Gavriel Feuer and Niklas Manz, 2014

Examples of simple `while` loops:

- ```
  while(1):
        print ("Hello World")
  ```
- ```
  while TRUE:
        print ("Hello World")
  ```
- ```
  i=0
  while (i<3):
        print (i)
        i=i+1
  ```
- ```
  i=0
  while (i<3):
        print (i)
        i+=1
  ```

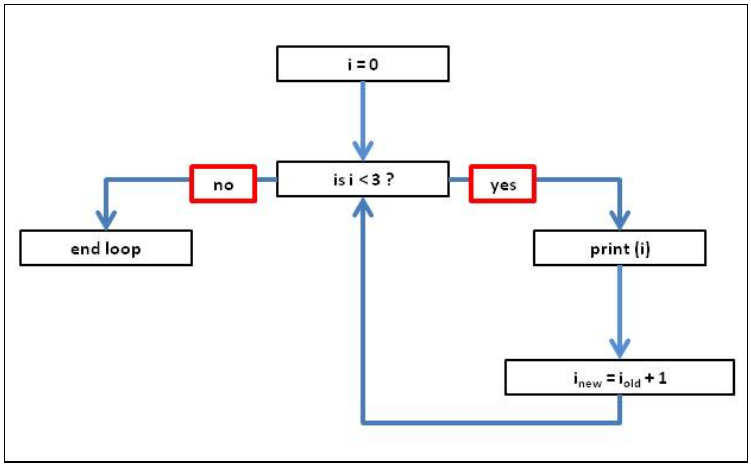


Figure 18: Flow chart of a simple `while` loop.

Different programs are just different computer languages but use the same concepts. The following table is a comparison of the syntax of the learned logical structures in five different computer languages.

- **Python** (Python is a widely used general-purpose, high-level programming language)
- **Unix** (Unix is a multitasking, multi-user computer operating system)

Copyright © Gavriel Feuer and Niklas Manz, 2014

- **Bourne shell** (The Bourne shell (sh) is a shell (computing), or command-line interpreter, for computer operating systems)
- **Ruby** (Ruby is a dynamic, reflective, general-purpose object-oriented programming language)
- **IDL** (IDL, short for Interactive Data Language, is a programming language used for data analysis)

Python	Unix	Bourne shell	Ruby	IDL

while <test>:
 	
<statements>
	while <test>
	do
	<statements>
	done
		while <test>
		do
		<statements>
		done
			while <test>
			<statements>
			end
				while <test> do
				begin
				<statements>
				endwhile

for <target> in <object>: <statements>	foreach <target> in <object> <statement> end	for <target> in <object> do <statement> done	for <target> in <object> <statement> end **OR** <object> do |<target>| <statement> end	for <target> in <object> do begin <statement> endfor

if <test1>:
<statements1>
elif <test2>:
<statements2>
else:
<statements3>
	if <test1> then
	<statements1>
	else if <test2>
	then
	<statements2>
	else
	<statements3>
	endif
		if <test1>
		then
		<statements1>
		elif <test2>
		then
		<statements2>
		else
		<statements3>
		fi
			if <test1>
			<statements1>
			elsif <test2>
			<statements2>
			else
			<statements3>
			end
				if <test1>then
				begin
				<statements1>
				endif else
				begin
				<statements3>
				endelse
				
				OR:
				
				if <test1>then
				begin
				<statements1>
				else then
				<statements2>
				endif

Table 19: Syntax comparison of while, for, and if across five different computer languages.

The three logical concepts and the syntax is like a language. For example, the concept of wishing someone a good morning can be expressed differently:

English	good morning		French	bonjour

Copyright © Gavriel Feuer and Niklas Manz, 2014

German	guten Morgen		Spanish	buenos días
Dutch	goedemorgen		Portuguese	bom dia
Africaans	goeie môre			

Table 20: Comparison of saying "Good Morning" in seven different languages of two language families.

Tasks	
1.	Create a never-ending while loop.
2.	Define a list and print the elements with a “for” loop.
3.	Create a loop with two variables. One variable increasing by 1. One variable increasing by 2.
4.	Multiply 500 x 500 using only addition and any loop (Hint: 5 x 5 = 5+5+5+5+5).

Table 21: Student tasks for the loop session.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 7

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 8 – Event Handling

Tasks	
1.	Play a sound on the computer using events.
2.	Edit the move_player.py program to include events that change the shapes color.

Table 22: First student tasks for the event handling session.

Tasks		
1.	Figure out the controls for the player on the left.	
2.	general	Make the ball twice as big.
3.		Define a new color and use it for the ball.
4.		Make the paddle in red.
5.		Make the paddle a little bit longer.
6.	events	In the Events section, add a new event that changes the screen color to green when you press the “g” Key Down on the keyboard
7.		In the Events section, add new events to move the paddles left or right.

Table 23: Second student tasks for the event handling session.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 8

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 9 – Images

Abbreviation	Name	Compression	Remarks
bmp	Windows bitmap	lossless	Graphics widely accepted in Windows programs, can be large.
gif	Graphics Interchange Format	lossless	8-bit color images (256 colors). One possible transparent color.
jpg (jpeg)	Joint Photographic Expert Group	**lossy**	8-bit grayscale images and 24-bit color images, **quality degradation when repeatedly edited and saved**.
png	Portable Network Graphics	lossless	24-bit color images, **possible transparency of all colors**, good for web browsers
tif (tiff)	Tagged Image File Format	lossy or lossless	24-bit or 48-bit color images, not widely supported by web browsers
svg	Scalable Vector Graphics	-	

Table 24: Image file formats.

Abbreviation	File size (kB)	Color depth (bpp)
bmp	245	24
gif	27	8
jpg (jpeg)	61	24
png	83	24
tif (tiff)	245	24

Table 25: Parameter of a SUNY Downstate logo (300 x 278).

Copyright © Gavriel Feuer and Niklas Manz, 2014

Tasks	
1.	Load a different background image into the game.
2.	Change the transparent color of the figure from white (its background) to black (the figure). What is the effect?
3.	Load the image 'stick.png' into a graphics program. Save the image in the jpg file format. Use 'stick.jpg' as the player image and change the transparent color back to white (as in the beginning). What is the effect?
4.	Comment out the line containing the definition of the transparent color. What is the effect?
5.	Load the image 'stick.jpg' into a graphics program. Save the image in the png file format under a different name. Use 'NewName.png' as the player image and uncomment the line containing the definition of the transparent color. What is the effect?

Table 26: Student tasks for the image topic.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 9

Copyright © Gavriel Feuer and Niklas Manz, 2014

Topic 10 – Classes and Sprites

Tasks	
1.	Change the number of the players
2.	Create a new sprite function that plays a sound when the player crosses the vertical midline of the screen.
3.	Create a new sprite.

Table 27: Student tasks for the classes and sprites topic.

Copyright © Gavriel Feuer and Niklas Manz, 2014

Notes for Topic 10

Copyright © Gavriel Feuer and Niklas Manz, 2014

Python Commands Cheat Sheet for tasks

range(start,stop)

Example 1:

```
>>> my_range = range(1,11)
>>> my_range
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Example 2:

```
>>> start=1
>>> stop=11
>>> my_range = range(start,stop)
>>> my_range
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

print('letters in quotes')

Example 1:

```
>>> print('the first game ever was pong')
the first game ever was pong
```

Example 2:

```
>>> my_list=['halo','pong','portal']
>>> print(my_list)
['halo', 'pong', 'portal']
```

Comparison operators:

Example 1:

```
>>> a=5
>>> b=a
>>> b==5
True
```

Example 2:

```
>>> b=5
>>> c=10
>>> b>c
```

Example 3:

```
>>> b=5
>>> c=10
>>> b<c
True
```

Example 4:

```
>>> b=5
>>> b>'bananas'
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

False	False

Copyright © Gavriel Feuer and Niklas Manz, 2014

if test:
  perform_commands

Example:
```
A=5
B=10
if A<B:
  print ('A is less than B')
else:
  print ('A is not less than B')
```

while (argument):
  perform_commands

Example 1:
```
i=0
while i < 3:
  print(i)
  i=i+1
```

for item in list:
  perform_commands

Example 1:
```
>>> my_list=['halo','pong','portal']
>>> for each_item in my_list:
        print each_item

halo
pong
portal
```

input (for numbers!)

```
Example:
Age_A = input('Please enter your age:')
```

raw_input (for text!)

```
Example:
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
Name_A = input('Please enter your name:')
```

Game code template

```
#Pygame Template
#Gavriel Feuer
#9/29/13
#---------------------Notes-------------------------
# This is a template that can be used to start any pygame.
#---------------------------------------------------

############## Import Libraries ######################
import pygame

############## Define Variables/Initilization ######################
#All variable definitions, functions, and class objects go below this line

#Define color palette
#          r   g   b
black  = (  0,  0,  0)
white  = (255,255,255)
red    = (255,  0,  0)
green  = (  0,255,  0)
blue   = (  0,  0,255)

#set display parameters
size_x=700
size_y=500
size=[size_x,size_y]
display=pygame.display.set_mode(size)
pygame.display.set_caption("Template Game")

#set the clock to manage how fast the screen updates
clock=pygame.time.Clock()

#Setup the loop control
rungame=True

#------------Main Program Loop -----------------
while rungame:

    ########## Events ######################
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False

    ######### Begin Game Logic ####################
    #Start by clearing the old display
    display.fill(blue)

    ######### End Game Logic ######################

    ########## Update screen and clock #############
    #update the screen with the new drawing
    pygame.display.flip()
    #set the clock speed
    clock.tick(20)
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

Source codes

draw_player.py

```
#Drawing of Colorful Shapes
#Gavriel Feuer
#9/17/13
#--------------------Notes-------------------------
import pygame

#Define color palette
#          r   g   b
BLACK = (  0,  0,  0)
WHITE = (255,255,255)
RED   = (255,  0,  0)
BLUE  = (  0,  0,255)
GREEN = (  0,255,  0)

#Initialize the pygame library
pygame.init()

#Set the size of the display canvas: Q-What are the units for the size?
size_x  = 700
size_y  = 500
size    = [size_x, size_y]

screen  = pygame.display.set_mode(size)
pygame.display.set_caption("Retro Screensaver")

#Set the clock to manage how fast the screen updates
clock=pygame.time.Clock()

#Set up the shape parameters
radius      = 50
line_width  = 2

#set the starting position of our shape
position_x  = 100
position_y  = 100

#This is the driving force for the program
rungame=True

#------------Main Program Loop -----------------
while rungame:
   for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False
   #set the background color: Q- what would happen if this line was not included?
   screen.fill(WHITE)

   #Draw the shape: Learn to look at the reference manual. DIY
http://www.pygame.org/docs/ref/draw.html
   pygame.draw.circle(screen, GREEN, [position_x+5,position_y+5], radius, line_width)
   # Outer circle
   pygame.draw.circle(screen, GREEN, [position_x, 10+position_y], radius, line_width )
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    # Head
    pygame.draw.circle(screen, BLACK, [1+position_x, -10+position_y], 10, 0 )

    # Body
    pygame.draw.line(screen,RED,[position_x,25+position_y],[position_x,position_y],10)

    # Arms
    pygame.draw.line(screen,RED,[position_x,8+position_y],[-20+position_x,10+position_y],4)
    pygame.draw.line(screen,RED,[position_x,8+position_y],[ 20+position_x,10+position_y],4)
     # Legs
    pygame.draw.line(screen,BLUE,[position_x-2,25+position_y],[-8+position_x,45+position_y],6)
    pygame.draw.line(screen,BLUE,[position_x+2,25+position_y],[ 8+position_x,45+position_y],6)
    #Set the clock speed [frames per second]
    clock.tick(60)

    #Update the screen with the new drawing
    pygame.display.flip()

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

coordinate.py

```
#Computer screen coordinate exercise
#Gavriel Feuer
#1/29/14
#--------------------Notes-------------------------
#--------------------------------------------------

import pygame

#Define Color Palatte
#   RGB
black = (0,0,0)
white=(255,255,255)
red=(255,0,0)
green=(0,255,0)
blue=(0,0,255)

pygame.init()

radius = 5
line_width=0

size_x=700
size_y=500
size=[size_x,size_y]

screen = pygame.display.set_mode(size)
font = pygame.font.SysFont("Arial", 36)
background = pygame.Surface(screen.get_size())
pygame.display.set_caption("Coordinate System")

#set the clock to manage how fast the screen updates
clock=pygame.time.Clock()
rungame=True
while rungame:

    ########## Events ######################
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False
            ########## Drawing ############################
    #set the background color
    screen.fill(white)
    #pygame.draw.circle(screen, black, [0, 0], radius+10, line_width)
    pygame.draw.circle(screen, green, [100, 100], radius, line_width)
    pygame.draw.circle(screen, red, [100, 200], radius, line_width)
    pygame.draw.circle(screen, blue, [300, 50], radius, line_width)
    text1=font.render("(0,0)",True,black)
    text_position1=[0,0]
    text2=font.render("(100,100)",True,green)
    text_position2=[100,100]
    text3=font.render("(100,200)",True,red)
    text_position3=[100,200]
    text4=font.render("(300,50)",True,blue)
    text_position4=[300,50]

    #screen.blit(text1,text_position1)
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    screen.blit(text2,text_position2)
    screen.blit(text3,text_position3)
    screen.blit(text4,text_position4)
    #Draw our figure
    ########## Update screen and clock #############
    #update the screen with the new drawing
    pygame.display.flip()
    #set the clock speed
    clock.tick(20)

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

bounce_player.py

```
#Drawing and Movement of Colorful Shapes
#Gavriel Feuer
#9/17/13
#--------------------Notes-------------------------
#--------------------------------------------------

import pygame

#Define color palette
#         r   g   b
BLACK = (  0,  0,  0)
WHITE = (255,255,255)
RED   = (255,  0,  0)
GREEN = (  0,255,  0)
BLUE  = (  0,  0,255)

#Initialize the pygame library
pygame.init()

#Set the size of the display canvas: Q-What are the units for the size?
size_x  = 700
size_y  = 500
size    = [size_x, size_y]

screen  = pygame.display.set_mode(size)
pygame.display.set_caption("Retro Screensaver")

#Set the clock to manage how fast the screen updates
clock=pygame.time.Clock()

#Set up the shape parameters
radius      = 50
line_width  = 3

#set the starting position of our shape
position_x  = 100
position_y  = 100

#set the speed and direction of the shape
speed_x     = 3
speed_y     = 3

#This is the driving force for the program
rungame=True

#------------Main Program Loop -----------------
while rungame:
   for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False
   #set the background color: Q- what would happen if this line was not included?
   screen.fill(WHITE)

   #Draw the shape: Learn to look at the reference manual. DIY
http://www.pygame.org/docs/ref/draw.html
   # Outer circle
   pygame.draw.circle(screen, GREEN, [position_x, 10+position_y], radius, line_width )
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    # Head
    pygame.draw.circle(screen, BLACK, [1+position_x, -10+position_y], 10, 0 )

    # Body
    pygame.draw.line(screen,RED,[position_x,25+position_y],[position_x,position_y],10)

    # Arms
    pygame.draw.line(screen,RED,[position_x,8+position_y],[-20+position_x,10+position_y],4)
    pygame.draw.line(screen,RED,[position_x,8+position_y],[ 20+position_x,10+position_y],4)

    # Legs
    pygame.draw.line(screen,BLUE,[position_x-2,25+position_y],[-8+position_x,45+position_y],6)
    pygame.draw.line(screen,BLUE,[position_x+2,25+position_y],[ 8+position_x,45+position_y],6)
    #Move the shape
    position_x = position_x + speed_x
    position_y = position_y + speed_y

    #Bounce the ball if it hits a wall or an obstacle
    if position_x > size_x-radius or position_x < 0+radius:
         speed_x = -1*speed_x
    if position_y > size_y-radius or position_y < 0+radius:
         speed_y = -1*speed_y

    #Set the clock speed [frames per second]
    clock.tick(60)

    #Update the screen with the new drawing
    pygame.display.flip()

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

bounce_player_sound.py

```
#Drawing and Movement with Sound
#Gavriel Feuer
#9/17/13
#--------------------Notes-------------------------
# -make sure the sound file is in the same folder as the bounce_player_sound.py file
#--------------------------------------------------

import pygame

#Define color palette
#         r   g   b
BLACK = (  0,  0,  0)
WHITE = (255,255,255)
RED   = (255,  0,  0)
GREEN = (  0,255,  0)
BLUE  = (  0,  0,255)

#Initialize the pygame library
pygame.init()

#Set the size of the display canvas: Q-What are the units for the size?
size_x  = 700
size_y  = 500
size    = [size_x, size_y]

screen  = pygame.display.set_mode(size)
pygame.display.set_caption("Retro Screensaver")

# Sounds
bounce_sound = pygame.mixer.Sound("beep1.ogg")

#Set the clock to manage how fast the screen updates
clock=pygame.time.Clock()

#Set up the shape parameters
radius      = 50
line_width  = 3

#set the starting position of our shape
position_x  = 100
position_y  = 100

#set the speed and direction of the shape
speed_x     = 3
speed_y     = 3

#This is the driving force for the program
rungame=True

#------------Main Program Loop -----------------
while rungame:
   for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False
   #set the background color: Q- what would happen if this line was not included?
   screen.fill(WHITE)
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    #Draw the shape: Learn to look at the reference manual. DIY
http://www.pygame.org/docs/ref/draw.html
    # Outer circle
    pygame.draw.circle(screen, GREEN, [position_x, 10+position_y], radius, line_width)

    # Head
    pygame.draw.circle(screen, BLACK, [1+position_x, -10+position_y], 10, 0 )

    # Body
    pygame.draw.line(screen,RED,[position_x,25+position_y],[position_x,position_y],10)

    # Arms
    pygame.draw.line(screen,RED,[position_x,8+position_y],[-20+position_x, 10+position_y],4)
    pygame.draw.line(screen,RED,[position_x,8+position_y],[ 20+position_x, 10+position_y],4)

    # Legs
    pygame.draw.line(screen,BLUE,[position_x-2,25+position_y],[-8+position_x,45+position_y],6)
    pygame.draw.line(screen,BLUE,[position_x+2,25+position_y],[ 8+position_x,45+position_y],6)
    #Move the shape
    position_x = position_x + speed_x
    position_y = position_y + speed_y

    #Bounce the ball if it hits a wall or an obstacle
    if position_x > size_x-radius or position_x < 0+radius:
         speed_x = -1*speed_x
         bounce_sound.play()
    if position_y > size_y-radius or position_y < 0+radius:
         speed_y = -1*speed_y
         bounce_sound.play()

    #Set the clock speed [frames per second]
    clock.tick(60)

    #Update the screen with the new drawing
    pygame.display.flip()

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

play_tone.py

```
#Generating sounds in Pygame using numpy
#Gavriel Feuer
#8/25/13
#-------------------------------------------------
##-Before Running this program make sure numpy is installed
(https://pypi.python.org/pypi/numpy)
##-additional information on generating sounds is available here
(http://web.media.mit.edu/~nvawter/otherProduct/keyboard5.py)
#-------------------------------------------------
import numpy
import pygame

Fs=44100
pygame.mixer.init(Fs,-16,1)
pygame.init()
pygame.display.set_mode((800,600))
time=5

length=Fs*time
##freq=600.0
freq1=800.0
##freq2=600.0
##freq3=460.0
#freq1=220.0*(4.0/3.0)
#freq2=17.5
amplitude=500.0
tmp=[]

for t in range(int(length)):
        v1=amplitude*numpy.sin(t*freq1/Fs*2*numpy.pi)
##        v2=amplitude*numpy.sin(t*freq2/Fs*2*numpy.pi)
##        v2=amp*numpy.sin(t*freq3/Fs*2*numpy.pi)
#        v1=1+0.2*numpy.sin(t*freq1/Fs*2*numpy.pi)
#        v2=amplitude*numpy.sin(t*freq2/Fs*2*numpy.pi*v1)
#        v=v1*v2
        v=v1        tmp.append(v1)
sound=pygame.sndarray.make_sound(numpy.array(tmp,numpy.int16))

rungame=True

while rungame:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False
    sound.play()

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

if_statements.py

```
#If examples
#Gavriel Feuer
#11/21/13
#--------------------Notes-------------------------
# Run these code segments in the python shell.
#--------------------------------------------------

if True:
   print(Hello")

################################

#A=5
#B=10
#if A<B:
#    print('A is less than B')

################################

#A=5
#B=10
#if A<B:
#    print('A is less than B')
#else:
#    print('A is not less than B')

################################

#A=input("Please enter a value for A:")
#B=input("Please enter a value for B:")
#if A<B:
#    print("A is less than B")
#else:
#    print("B is less than A")

################################

#Name_A=raw_input('Please enter your name:')
#Name_B=raw_input("Please enter your partner's name:")
#Age_A=input('Please enter your age:')
#Age_B=input("Please enter your partner's age:")

#if Age_A<Age_B:
#    print Name_A, 'is younger than', Name_B
#if Age_A==Age_B:
#    print Name_A, 'and', Name_B, 'are the same age'
#if Age_A>Age_B:
#    print Name_A, 'is older than', Name_B
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

while_for_loops.py

```
#While loop and For loop examples
#Gavriel Feuer
#9/29/13
#--------------------Notes-------------------------
# Run these code segments in the python shell.
#--------------------------------------------------

############## While Examples ####################
##while(1):
##    print("Hello World!")
##
##while True:
##    print("Hello World!")

##i=0
##while i<10:
##    print(i)
##    i=i+1

##i=0
##while i<10:
##    print(i)
##    i+=1

############## For Examples ############################

##for i in range(10):
##    print(i)
##
##for i in range(0,10):
##    print(i)
##
##for i in range(0,20,2):
##    print(i)

##alphabet=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s
','t','u','v','w','x','y','z']
##for letter in alphabet:
##    print(letter)
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

move_player.py

```
#Event Handling and User Input
#Gavriel Feuer
#9/17/13
#--------------------Notes-------------------------
#--------------------------------------------------

import pygame

#Define color palette
#          r   g   b
black  = (  0,  0,  0)
white  = (255,255,255)
red    = (255,  0,  0)
green  = (  0,255,  0)
blue   = (  0,  0,255)

screen_color=white

#Initialize the pygame library
pygame.init()

#Set up the shape parameters
radius      = 50
line_width  = 3

# Sounds
bounce_sound = pygame.mixer.Sound("beep1.ogg")

########## Character Design ###########
#Create a function that defines the stick figure
def draw_figure(screen,x,y):
    #Draw the shape: Learn to look at the reference manual. DIY
http://www.pygame.org/docs/ref/draw.html
    # Outer circle
    pygame.draw.circle(screen, green, [x, 10+y], radius, line_width )

    # Head
    pygame.draw.circle(screen, black, [1+x, -10+y], 10, 0 )

    # Body
    pygame.draw.line(screen,red,[x,25+y],[x,y],10)

    # Arms
    pygame.draw.line(screen,red,[x,8+y],[-20+x,10+y],4)
    pygame.draw.line(screen,red,[x,8+y],[ 20+x,10+y],4)

    # Legs
    pygame.draw.line(screen,blue,[x-2,25+y],[-8+x,45+y],6)
    pygame.draw.line(screen,blue,[x+2,25+y],[8+x,45+y],6)
#######################################

#Initialize pygame
pygame.init()

#set the size of the display canvas
size_x=700
size_y=500
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
size=[size_x,size_y]

screen = pygame.display.set_mode(size)
pygame.display.set_caption("Move the stick figure")

#set the clock to manage how fast the screen updates
clock=pygame.time.Clock()

#Hide the mouse cursor
pygame.mouse.set_visible(0)

#set the starting position
position_x=100
position_y=100

#set the starting speed to zero
speed_x = 0
speed_y = 0

#This is the driving force for the program
rungame=True

#------------Main Program Loop -----------------
while rungame:

    ########## Events ######################
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False

        #Check to see whether the user pressed a key on the keyboard
        #If a key was pressed, check if it was an arrow key
        #adjust the corresponding speed according to the button that was pressed
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                speed_x=-3
            if event.key == pygame.K_RIGHT:
                speed_x=3
            if event.key == pygame.K_UP:
                speed_y=-3
            if event.key == pygame.K_DOWN:
                speed_y=3

        #Check to see whether the user UNpressed a key
        #If a key was lifted, check if it was an arrow key
        #adjust the corresponding speed
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                speed_x=0
            if event.key == pygame.K_RIGHT:
                speed_x=0
            if event.key == pygame.K_UP:
                speed_y=0
            if event.key == pygame.K_DOWN:
                speed_y=0
    ########## Game Logic ########################
    position_x += speed_x
    position_y += speed_y

    #Bounce the ball if it hits a wall or an obstacle
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    if position_x > size_x-radius: #right wall
        speed_x = -1*speed_x
        bounce_sound.play()
        screen_color=blue
    if position_x < 0+radius: # left wall
        speed_x = -1*speed_x
        bounce_sound.play()
    if position_y > size_y-radius: # bottom wall
        position_x=100
        position_y=100

    if position_y < 0+radius: # top wall
        speed_y = -1*speed_y

    ########## Drawing ##########################
    #set the background color
    screen.fill(white)
    #Draw our figure
    draw_figure(screen,position_x, position_y)
    ########## Update screen and clock #############
    #update the screen with the new drawing
    pygame.display.flip()
    #set the clock speed
    clock.tick(20)

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

bitmapped_player.py

```
#Bitmapped Images
#Gavriel Feuer
#2/5/2014
#--------------------Notes-------------------------
#--------------------------------------------------
import pygame

#Define color palette
#          r   g   b
black  = (  0,  0,  0)
white  = (255,255,255)
red    = (255,  0,  0)
green  = (  0,255,  0)
blue   = (  0,  0,255)

#Initialize the pygame library
pygame.init()

#set the size of the display canvas
size_x=700
size_y=500
size=[size_x,size_y]

screen = pygame.display.set_mode(size)
pygame.display.set_caption("Move the stick figure")

background=pygame.Surface(screen.get_size())
background.fill(white)

background_image = pygame.image.load("Hydrangeas.jpg").convert()

#set the clock to manage how fast the screen updates
clock=pygame.time.Clock()

#Set up parameters
position_x=100
position_y=100

########## Character Design ###########
#Create a class that defines the player
class Player(pygame.sprite.Sprite):

    #Initialize the player
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        self.image= pygame.image.load("stick.png").convert()

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.speed_x=0
        self.speed_y=0

    def move(self):
        self.rect.x+=self.speed_x
        self.rect.y+=self.speed_y
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    def bounce(self):
    #Bounce the player if it hits a wall or an obstacle
        if self.rect.x>size_x or self.rect.x < 0:
            self.speed_x=-1*self.speed_x
            bounce_sound.play()
        if self.rect.y>size_y or self.rect.y < 0:
            self.speed_y=-1*self.speed_y
            bounce_sound.play()

############Sprite Initialization#####################

player=Player(position_x,position_y)
sprite_list = pygame.sprite.RenderPlain()
sprite_list.add(player)


#######################################################

#This is the driving force for the program
rungame=True

#------------Main Program Loop -----------------
while rungame:

    ########## Events ######################
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False

        #Check to see whether the user pressed a key on the keyboard
        #If a key was pressed, check if it was an arrow key
        #adjust the corresponding speed according to the button that was pressed
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                player.speed_x=-3
            if event.key == pygame.K_RIGHT:
                player.speed_x=3
            if event.key == pygame.K_UP:
                player.speed_y=-3
            if event.key == pygame.K_DOWN:
                player.speed_y=3

        #Check to see whether the user UNpressed a key
        #If a key was lifted, check if it was an arrow key
        #adjust the corresponding speed
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                player.speed_x=0
            if event.key == pygame.K_RIGHT:
                player.speed_x=0
            if event.key == pygame.K_UP:
                player.speed_y=0
            if event.key == pygame.K_DOWN:
                player.speed_y=0
    ########## Game Logic ########################
    player.move()
    player.bounce()


    ########## Drawing ###########################
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    #set the background color
    #screen.fill(white)
    screen.blit(background_image, [0,0])
    #Draw our figure
    sprite_list.draw(screen)
    ########## Update screen and clock #############
    #update the screen with the new drawing
    pygame.display.flip()
    #set the clock speed
    clock.tick(20)

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

move_player_sprite.py

```
#Sprite Motion 1
#Gavriel Feuer
#9/17/13
#--------------------Notes-------------------------
#--------------------------------------------------
import pygame

#Define color palette
#          r   g   b
black  = (  0,  0,  0)
white  = (255,255,255)
red    = (255,  0,  0)
green  = (  0,255,  0)
blue   = (  0,  0,255)

#Initialize the pygame library
pygame.init()

#set the size of the display canvas
size_x=700
size_y=500
size=[size_x,size_y]

screen = pygame.display.set_mode(size)
pygame.display.set_caption("Move the stick figure")

#set the clock to manage how fast the screen updates
clock=pygame.time.Clock()

#Hide the mouse cursor
pygame.mouse.set_visible(0)

#Set up the shape parameters
radius      = 50
line_width  = 3
position_x=100
position_y=100

# Sounds
bounce_sound = pygame.mixer.Sound("beep1.ogg")

########## Character Design ############
#Create a class that defines the player
class Player(pygame.sprite.Sprite):

    #Initialize the player
    def __init__(self,x,y):
        self.x=x
        self.y=y
        self.speed_x=0
        self.speed_y=0

    def display(self):
        # Outer circle
        pygame.draw.circle(screen, green, [self.x, 10+self.y], radius, line_width )
        # Head
        pygame.draw.circle(screen, black, [1+self.x, -10+self.y], 10, 0 )
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
        # Body
        pygame.draw.line(screen,red,[self.x,25+self.y],[self.x,self.y],10)
         # Arms
        pygame.draw.line(screen,red,[self.x,8+self.y],[-20+self.x,10+self.y],4)
        pygame.draw.line(screen,red,[self.x,8+self.y],[ 20+self.x,10+self.y],4)
        # Legs
        pygame.draw.line(screen,blue,[self.x-2,25+self.y],[-8+self.x,45+self.y],6)
        pygame.draw.line(screen,blue,[self.x+2,25+self.y],[ 8+self.x,45+self.y],6)

    def move(self):
        self.x+=self.speed_x
        self.y+=self.speed_y

    def bounce(self):
        #Bounce the ball if it hits a wall or an obstacle
        if self.x>size_x or self.x < 0:
            self.speed_x=-1*self.speed_x
            bounce_sound.play()
        if self.y>size_y or self.y < 0:
            self.speed_y=-1*self.speed_y
            bounce_sound.play()

############Sprite Initialization#####################

player=Player(position_x,position_y)

#####################################################

#This is the driving force for the program
rungame=True

#------------Main Program Loop -----------------
while rungame:

    ########## Events ######################
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False

        #Check to see whether the user pressed a key on the keyboard
        #If a key was pressed, check if it was an arrow key
        #adjust the corresponding speed according to the button that was pressed
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                player.speed_x=-3
            if event.key == pygame.K_RIGHT:
                player.speed_x=3
            if event.key == pygame.K_UP:
                player.speed_y=-3
            if event.key == pygame.K_DOWN:
                player.speed_y=3

        #Check to see whether the user UNpressed a key
        #If a key was lifted, check if it was an arrow key
        #adjust the corresponding speed
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                player.speed_x=0
            if event.key == pygame.K_RIGHT:
                player.speed_x=0
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
            if event.key == pygame.K_UP:
                player.speed_y=0
            if event.key == pygame.K_DOWN:
                player.speed_y=0
    ########## Game Logic ########################
    player.move()
    player.bounce()


    ########## Drawing ###########################
    #set the background color
    screen.fill(white)
    #Draw our figure
    player.display()
    ########## Update screen and clock #############
    #update the screen with the new drawing
    pygame.display.flip()
    #set the clock speed
    clock.tick(20)

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

move_player_sprite2.py

```
#Sprite Motion 2
#Gavriel Feuer
#9/17/13
#--------------------Notes-------------------------
#--------------------------------------------------

import pygame
import random

#Define color palette
#          r   g   b
black  = (  0,  0,  0)
white  = (255,255,255)
red    = (255,  0,  0)
green  = (  0,255,  0)
blue   = (  0,  0,255)

#Initialize the pygame library
pygame.init()

#set the size of the display canvas
size_x=700
size_y=500
size=[size_x,size_y]

screen = pygame.display.set_mode(size)
pygame.display.set_caption("Move the stick figures")

#set the clock to manage how fast the screen updates
clock=pygame.time.Clock()

#Hide the mouse cursor
pygame.mouse.set_visible(0)

# Number of players
number_of_players=10
my_players=[]

#Set up the shape parameters
radius      = 50
line_width  = 3
position_x=100
position_y=100

# Sounds
bounce_sound = pygame.mixer.Sound("beep1.ogg")

########## Character Design ###########
#Create a class that defines the player
class Player(pygame.sprite.Sprite):

    #Initialize the player
    def __init__(self,x,y):
        self.x=x
        self.y=y
        self.speed_x=0
        self.speed_y=0
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    def display(self):
        # Outer circle
        pygame.draw.circle(screen, green, [self.x, 10+self.y], radius, line_width )
        # Head
        pygame.draw.circle(screen, black, [1+self.x, -10+self.y], 10, 0 )
        # Body
        pygame.draw.line(screen,red,[self.x,25+self.y],[self.x,self.y],10)
         # Arms
        pygame.draw.line(screen,red,[self.x,8+self.y],[-20+self.x,10+self.y],4)
        pygame.draw.line(screen,red,[self.x,8+self.y],[ 20+self.x,10+self.y],4)
        # Legs
        pygame.draw.line(screen,blue,[self.x-2,25+self.y],[-8+self.x,45+self.y],6)
        pygame.draw.line(screen,blue,[self.x+2,25+self.y],[ 8+self.x,45+self.y],6)

    def move(self):
        self.x+=self.speed_x
        self.y+=self.speed_y
    def update(self):
        self.x+=self.speed_x
        self.y+=self.speed_y
    def bounce(self):
        #Bounce the ball if it hits a wall or an obstacle
        if self.x>size_x or self.x < 0:
            self.speed_x=-1*self.speed_x
            bounce_sound.play()
        if self.y>size_y or self.y < 0:
            self.speed_y=-1*self.speed_y
            bounce_sound.play()

############Sprite Initialization#####################
for n in range(number_of_players):
    x=random.randint(0,size_x)
    y=random.randint(0,size_y)
    player=Player(x,y)
    my_players.append(player)

#########################################################

#This is the driving force for the program
rungame=True

#------------Main Program Loop -----------------
while rungame:

    ########## Events ####################
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False

        #Check to see whether the user pressed a key on the keyboard
        #If a key was pressed, check if it was an arrow key
        #adjust the corresponding speed according to the button that was pressed
        for player in my_players:
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    player.speed_x=-3
                if event.key == pygame.K_RIGHT:
                    player.speed_x=3
                if event.key == pygame.K_UP:
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
                    player.speed_y=-3
                if event.key == pygame.K_DOWN:
                    player.speed_y=3

            #Check to see whether the user UNpressed a key
            #If a key was lifted, check if it was an arrow key
            #adjust the corresponding speed
            if event.type == pygame.KEYUP:
                if event.key == pygame.K_LEFT:
                    player.speed_x=0
                if event.key == pygame.K_RIGHT:
                    player.speed_x=0
                if event.key == pygame.K_UP:
                    player.speed_y=0
                if event.key == pygame.K_DOWN:
                    player.speed_y=0
    ########## Game Logic ########################
    #set the background color
    screen.fill(white)

    for player in my_players:
        player.move()
        player.bounce()
        #Draw our figure
        player.display()

    ########## Drawing ###########################
    #for this program the drawing function is within the game logic

    ########## Update screen and clock #############
    #update the screen with the new drawing
    pygame.display.flip()
    #set the clock speed
    clock.tick(20)

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

pong.py

```
#Pong
#Gavriel Feuer
#12/4/13
#--------------------Notes-----------------------------------------------------------
#In computer graphics, a sprite  is a two-dimensional image or animation that is
# integrated into a larger scene.
#--------Import Libraries-------------------------------------------------------------
import pygame

#-------------Color Pallet-------------------------
black  = (  0,  0,  0)
white  = (255,255,255)
red    = (255,  0,  0)
green  = (  0,255,  0)
blue   = (  0,  0,255)

#-------------Initializations-----------------------
pygame.init()
screensize_x=700
screensize_y=500
screensize=[screensize_x,screensize_y]
screen_color=black
screen = pygame.display.set_mode(screensize)
pygame.display.set_caption("Pong")
font = pygame.font.Font(None, 36)
background = pygame.Surface(screen.get_size())
clock=pygame.time.Clock()
paddle_width=20
paddle_height=80

#--------------Player Sprite-------------------
class Player(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        self.width=paddle_width
        self.height=paddle_height
        self.image=pygame.Surface([self.width,self.height])
        self.image.fill(white)
        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.speed_x=0
        self.speed_y=0
    def move(self):
        self.rect.x+=self.speed_x
        self.rect.y+=self.speed_y
    def collide(self):
        if self.rect.y<0:
            self.rect.y=0
        if self.rect.y>screensize_y-self.height:
            self.rect.y=screensize_y-self.height
        if self.rect.x<0:
            self.rect.x=0
        if self.rect.x>screensize_x-self.width:
            self.rect.x=screensize_x-self.width
#--------------Ball Sprite-------------------
class Ball(pygame.sprite.Sprite):
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.width=10
        self.height=10
        self.image=pygame.Surface([self.width,self.height])
        self.image.fill(blue)
        self.rect=self.image.get_rect()
        self.rect.x=screensize_x/2
        self.rect.y=screensize_y/2
        self.speed_x=-3
        self.speed_y=3
    def move(self):
        self.rect.x+=self.speed_x
        self.rect.y+=self.speed_y
    def collide(self):
        if self.rect.x<0 or self.rect.x>screensize_x-self.width:
            self.speed_x=-1*self.speed_x
        if self.rect.y<0 or self.rect.y>screensize_y-self.height:
            self.speed_y=-1*self.speed_y
    def gameover(self):
        if self.rect.x<0 or self.rect.x>screensize_x-paddle_width:
            self.rect.x=screensize_x/2
            return True
        else:
            return False

#------------Sprite initialization----------------
balls = pygame.sprite.Group()
allsprites = pygame.sprite.RenderPlain()
player2=Player(0,0)
player1=Player(screensize_x-paddle_width,0)
ball=Ball()
balls.add(ball)
allsprites.add(player1,player2,ball)

#-----------Game Initialization------------------
rungame=True
gameover=False

#-----------Main Program Loop---------------------

while rungame:
    screen.fill(screen_color)
    #----------Events-----------------------------
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rungame=False

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                player1.speed_y=-4
            if event.key == pygame.K_DOWN:
                player1.speed_y=4
            if event.key == pygame.K_w:
                player2.speed_y=-4
            if event.key == pygame.K_s:
                player2.speed_y=4
            if event.key == pygame.K_SPACE:
                gameover=False
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_UP:
                player1.speed_y=0
            if event.key == pygame.K_DOWN:
                player1.speed_y=0
            if event.key == pygame.K_w:
                player2.speed_y=0
            if event.key == pygame.K_s:
                player2.speed_y=0

    #---------Game Logic-----------------------------
    if not gameover:
        player1.move()
        player2.move()
        gameover=ball.gameover()
        ball.move()
        player1.collide()
        player2.collide
        ball.collide()
    if gameover:
        text=font.render("Game Over: Press Space",True,white)
        text_position=text.get_rect(centerx=background.get_width()/2)
        text_position.top=250
        screen.blit(text,text_position)

    if pygame.sprite.spritecollide(player1,balls,False):
        ball.speed_x=-1*ball.speed_x
    if pygame.sprite.spritecollide(player2,balls,False):
        ball.speed_x=-1*ball.speed_x

   #------------Update Drawings-------------------------
    allsprites.draw(screen)
    pygame.display.flip()

    clock.tick(60)

pygame.quit()
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

pygame2exe.py

```
################################################################################
# 2exe.py
################################################################################
# Script for invoking py2exe to turn a game into an exe
################################################################################

################################################################################
# IMPORTS
################################################################################
from distutils.core import setup
import py2exe
import sys
import os
import glob
import shutil
import pygame
from modulefinder import Module

################################################################################
# GLOBAL DATA
################################################################################

VERSION = 'VERSION'
AUTHOR_NAME = 'AUTHOR_NAME'
AUTHOR_EMAIL = 'AUTHOR_EMAIL'
AUTHOR_URL = "AUTHOR_URL"
PRODUCT_NAME = "PRODUCT_NAME"
SCRIPT_MAIN = 'your_code.py'
VERSIONSTRING = PRODUCT_NAME + " ALPHA " + VERSION

PYGAMEDIR = os.path.split(pygame.base.__file__)[0]

ICONFILE = None

#all .dlls from the pygame directory will be copied to the dist dir
SDL_DLLS = glob.glob(os.path.join(PYGAMEDIR,'*.dll'))

#if true, the build directory will be deleted at the end of the build
REMOVE_BUILD_ON_EXIT = True

#Extra files to be included in the dist directory
                  #directory #files
extra_files = [ #("",         ["README.txt"]),
              ]

#list of modules to be excluded from the .exe
MODULE_EXCLUDES =[
'email',
'AppKit',
'Foundation',
'bdb',
'difflib',
'tcl',
'Tkinter',
'Tkconstants',
'curses',
'distutils',
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
'setuptools',
'urllib',
'urllib2',
'urlparse',
'BaseHTTPServer',
'_LWPCookieJar',
'_MozillaCookieJar',
'ftplib',
'gopherlib',
'_ssl',
'htmllib',
'httplib',
'mimetools',
'mimetypes',
'rfc822',
'tty',
'webbrowser',
'socket',
'base64',
'compiler',
'pydoc']


INCLUDE_STUFF = ['encodings',"encodings.latin_1",]

############################################################################
# OVERRIDE OF BUILDEXE
############################################################################

#override is used to ensure the default font is included
class BuildExe(py2exe.build_exe.py2exe):
   def copy_extensions(self,extensions):
      defaultFont = os.path.join(PYGAMEDIR,pygame.font.get_default_font())

      extensions.append(Module("pygame.font",defaultFont))
      py2exe.build_exe.py2exe.copy_extensions(self,extensions)

############################################################################
# EXECUTION
############################################################################

#append 'py2exe' to the arguments to invoke py2exe
sys.argv.append("py2exe")

#if the dist directory already exists, delete it
if os.path.exists('dist/'): shutil.rmtree('dist/')

#call setup with the correct parameters
setup(
   cmdclass = {'py2exe':BuildExe},
   windows=[
             {'script': SCRIPT_MAIN,
               'other_resources': [(u"VERSIONTAG",1,VERSIONSTRING)]}],
               #'icon_resources': [(1,"pygame.ico")]}],
   options = {"py2exe": {
                         "optimize": 2,
                         "includes": INCLUDE_STUFF,
                         "compressed": 1,
                         "ascii": 1,
                         "bundle_files": 2,
```

Copyright © Gavriel Feuer and Niklas Manz, 2014

```
                          "ignores": ['tcl','AppKit','Numeric','Foundation'],
                         "excludes": MODULE_EXCLUDES} },
    name = PRODUCT_NAME,
    version = VERSION,
    data_files = extra_files,
    zipfile = None,
    author = AUTHOR_NAME,
    author_email = AUTHOR_EMAIL,
    url = AUTHOR_URL)


#clean up
if os.path.exists('dist/tcl'): shutil.rmtree('dist/tcl')

# Remove the build tree
if REMOVE_BUILD_ON_EXIT:
      shutil.rmtree('build/')

if os.path.exists('dist/tcl84.dll'): os.unlink('dist/tcl84.dll')
if os.path.exists('dist/tk84.dll'): os.unlink('dist/tk84.dll')

for f in SDL_DLLS:
    fname = os.path.basename(f)
    try:
        shutil.copyfile(f,os.path.join('dist',fname))
    except: pass
```

Copyright © Gavriel Feuer and Niklas Manz, 2014