Good news! You've completed an equivalent course, so you don't need to complete this course to receive credit in the Fundamentals of Computing Specialization.

View

Concepts and Examples

This reading contains a short written summary of the concepts covered in this class. It also includes links to all of the examples include in video lectures as well as more examples produced by several students. Feel free to review this material if you feel that it is beneficial.

Week Zero - Comments, strings, numbers, arithmetic operators, arithmetic expressions, variables

Comments — CodeSkulptor

- Non-computational parts of the program that textually describe the behavior of the program.
- Comments begin with #, everything to right of the hash is ignored by Python.
- Comments should be frequent so you and others can understand the code.
- Lecture examples - CodeSkulptor
- More examples - Comments, Strings, and Print

Strings — CodeSkulptor

- Sequence of characters enclosed by a pair of single or double quotes
- Examples are "`cats hate dogs`" and '`Strings are fun!`'.
- Strings are one kind of data in Python. Their data type is denoted `str`.
- Lecture examples - Hello World
- More examples - Comments, Strings, and Print

Numbers — Arithmetic Expressions

- There are two kinds of numerical data in Python: integers and decimal numbers.

Help Center

- Integers correspond to the data type `int`. Decimal numbers are represented by floating-point numbers corresponding to the data type `float`.
- Floating-point numbers have around 15 decimal digits of accuracy.
- In CodeSkulptor, all numbers (even integers) are represented internally as floating-point numbers.
- Lecture examples - Arithmetic Expressions
- More examples - Floats and Ints

Arithmetic Operators — Arithmetic Expressions

- Five basic arithmetic operators; addition (+), subtraction (-), multiplication (*), division (/) and exponentiation (**)
- CodeSkulptor implements a subset of Python 2. In Python 2, the division operator (/) returns a float approximation to the exact answer if either of the operands is a float. If both operands are integers, division returns the exact answer round down to the nearest integer.
- The integer division operator // returns the quotient of two numbers..
- Lecture examples - Arithmetic Expressions
- More examples - Arithmetic Operations, Division

Arithmetic Expressions — Arithmetic Expressions

- An arithmetic expression is either a number or an operator applied to two arithmetic expressions.
- Arithmetic expressions are entered as a sequence of numbers and arithmetic operators.
- Expressions are formed by grouping operators and operands via precedence: "Please excuse my dear Aunt Sallie"; parentheses, exponentiation, multiplication, division, addition, subtraction.
- Lecture examples - Arithmetic Expressions
- More examples - Order of Operations for Arithmetic Expressions, Possible Errors for Arithmetic Expressions

Variables — Variables

- Variable names consist of a sequence of letters, number and underscores (_).
- Variable names start with a letter or underscore and are case sensitive.
- Single equals (=) is used for assignment to variables. Double equals (==) is used for testing equality.
- Lecture examples - Variables

- More examples - Variable Naming, Variable Assignment, Variable Operations, Formulas

Week One - Functions, indentation, remainders, modules, Boolean expressions, relational operators, conditional statements

Functions — Functions

- Functions are reusable pieces of programs that take an input and produce an output.
- A function definition is a compound statement consisting of a header and a body.
- The header includes the keyword `def`, a sequence of parameters enclosed by parentheses, followed by a colon `:`.
- The body consists of a sequence of statements, all indented by 4 spaces.
- Functions may return a value using the keyword `return` or have a side effect (e.g. `print`).
- To evaluate a function call, replace the function's parameters in the body of the function by their associated values in the call and execute the body of the function.
- Lecture examples - Functions
- More examples - Structure of Functions, Uses of Functions, Scope of Variables, Examples of Functions

Indentation — Functions

- Indentation consists of whitespace formed by blanks, tabs, and newlines.
- Leading white space indicates indentation level (4 spaces per level) and specifies logical grouping of statements in Python.
- Incorrect indentation can lead to errors.
- Lecture examples - Functions
- More examples - Function Errors

Remainders and modular arithmetic — More Operations

- Standard long division yields a quotient and a remainder. The integer division operator `//` computes the quotient. The operator `%` computes the remainder.
- For any integers `a` and `b`, `a == b * (a // b) + (a % b)`.
- In Python, `a % b` always returns an answer that is between `0` and `b` (even if `a` and/or `b` is negative).
- Remainders and modular arithmetic are very useful in games for the purpose of "wrapping" the

canvas, i.e. causing objects that pass off of one side of the canvas to reappear on the opposite side of the canvas.

- Lecture examples - More operations
- More examples - Modulus, Math Module,

Modules — More Operations

- Modules are libraries of Python code that implement useful operations not included in basic Python.
- Modules can be accessed via the `import` statement.
- CodeSkulptor implements parts of the standard Python modules `math` and `random`.
- Lecture examples - More operations
- More examples - Math Module, Numbers and Strings, Random Module, Module Errors

Boolean Expressions — Logic and Comparisons

- The constants `True` and `False` of the type `bool`.
- These constants can be combined to form Boolean expressions via the logical operators `and`, `or`, and `not`.
- The `and` of two Boolean expressions is `True` if both of the expressions are `True`.
- The `or` of two Boolean expressions is `True` if at least one of the expressions is `True`.
- Lecture examples - None
- More examples - Booleans, Boolean Logic

Relational Operators — Logic and Comparisons

- The values of two arithmetic expressions can be compared using the operators ==, !=, <, >, <=, >=.
- These comparisons return either `True` or `False` .
- Lecture examples - None
- More examples - Comparison, Boolean Expressions

Conditional Statements — Conditionals

- Conditional statements are compound statements consisting one or more clauses headed by the keywords `if`, `elif`, and `else`.
- Each `if` or `elif` clause is followed by a Boolean expression and a colon :.

- If the Boolean expression for a clause is `True`, the body of the clause is executed.
- Lecture examples - Conditionals
- More examples - if-elif-else, Examples of Conditionals

Programming Tips — Week 1

Week Two - Event handlers, local variables, global variables, SimpleGui module, buttons, input fields

Event handlers — Event-driven programming

- Event handlers (also called callbacks) are functions registered to an event such as a button click, keyboard press or mouse click.
- Event handlers react to the event by changing the state (collection of information) encoded in the program.
- Lecture examples - Events
- More examples - None

Local variables — Local vs. global variables

- Assignment to a variable inside a Python function creates a local variable.
- The scope of variable (portion of the program where the value of the variable can be accessed) is the body of function.
- Lecture examples - Local vs Global
- More examples - Example

Global variables — Local vs. global variables

- Variables defined outside functions are global variables. Their values may be accessed inside functions without declaration.
- To modify to a global variable inside a function, the variable must be declared inside the function using the keyword `global`.
- Global variables are a convenient (but dangerous) way for event handlers to share information in event-driven programming.
- Lecture examples - Local vs Global
- More examples - Example

SimpleGUI module — SimpleGUI

- Special module for CodeSkulptor that supports 2D interactive applications. The Docs button links to documentation for SimpleGUI.
- SimpleGUI allows creations of frames and timers as well as loading sounds and images.
- Frames include a control panel (with buttons and input fields), a status area (for monitoring keyboard and mouse events) and a canvas (with simple 2D drawing operations).
- Lecture examples - SimpleGUI, Template
- More examples - Layout, Frame, Errors

Buttons — Buttons

- Buttons may be created (and their event handlers registered) via `add_button`.
- Buttons are positioned linearly (top/down) in the control panel in their order of creation.
- Lecture examples - Calculator, Buttons
- More examples - Structure, Canvas Color, Prize Boxes, True-False Quiz

Input fields — Input fields

- Input fields may be created (and their event handlers registered) via `add_input`.
- Input fields are positioned linearly (top/down) in the control panel in their order of creation.
- The event handlers for the input field take a single parameter that is the text string entered.
- Lecture examples - Calculator, Input Fields
- More examples - Structure, Functions, Factoring, Silly Words

Programming Tips — Week 2

Week Three - Canvas, event-driven drawing, string operations, drawing operations, timers

Canvas — Canvas and drawing

- The canvas is the area associated with an application where information contain in the application may be drawn.
- In SimpleGUI applications, the width and height of the canvas are specified (in pixels) in `create_frame`. (A pixel is the smallest unit of area that your computer can draw in.)
- The origin for the canvas is the upper left-hand corner.

- Positions in the canvas are specified as pairs `[x,y]` where `x` is the horizontal coordinate and `y` is the vertical coordinate.
- Lecture examples - Canvas and Drawing
- More examples - Structure

Event-driven drawing — Canvas and drawing

- Computers refresh their screens around 60 times per sec.
- For each application, the computer calls a special event handler, the draw handler, that is registered to the application.
- In SimpleGUI, the draw handler is registered via `set_draw_handler`.
- The draw handler can modify the canvas via simple draw operations defined in SimpleGUI
- Lecture examples - Canvas and Drawing
- More examples - Structure, Echo

String operations — String processing

- The function `str` converts other types of data into a string.
- The concatenation operator + joins two strings to form a single string,
- The `i`th element of a string `my_string` can be access via `my_string[i]`. Note that strings are immutable (cannot be changed).
- Substrings of `my_string` can be accessed via *slicing* .
- Lecture examples - Strings, Dollars and Cents One, Dollars and Cents Two
- More examples - Operations, Input Checking, Initials

Drawing text — Interactive drawing

- The method `draw_text` draws a string when given a position (lower left corner), a font size and a color.
- The method `draw_circle` (see "More examples" below) draws a circle at a given point with a given radius in pixels. To draw a point, draw a circle of radius one.
- The method `draw_line` (see "More examples" below) draws a line between two points.
- The method `draw_polygon` (see "More examples" below) draws a sequence of points (enclosed in square brackets and separated by commas) as a closed polygon.
- Colors for drawing methods can be specified as HTML color strings;
`"White"`, `"Red"`, `"Green"`, ...

- Lecture examples - Drawing, Interactive Drawing
- More examples - Shapes, Pictures, Hidden Picture

Timers — Timers

- Timers are another component of SimpleGUI that generate regularly timed events.
- Users create a timer using `create_timer` with a specified interval and event handler.
- The timer calls its associated event handler regularly at the given interval.
- A timer `t` is started with `t.start()` and is stopped by `t.stop()`.
- Lecture examples - Timers
- More examples - Blinking Text, Reaction Time

Programming Tips — Week 3 (pt. 1), Week 3 (pt. 2)

Week Four - Lists, distance computations, reflections, keyboard events, positional control, velocity control, mutable vs. immutable data

Lists — Lists

- Lists can be constructed as a sequence of objects inside square brackets; e.g. `[2,4,6]`. The `list` function also converts other types of sequence data such as strings into lists.
- Sequence operations such as indexing `l[i]`, concatenation `+`, length `len(l)` and slicing apply to lists.
- As opposed to strings, an element of a list can be *mutated* via assignment `l[i] = x`.
- Lecture examples - None
- More examples - Structure, Tuples, True-False Quiz, Silly Words, Rainbow Canvas, Digital Numbers

Points and vectors — Motion

- A point in 2D is represented by a pair of Cartesian coordinates.
- A vector in 2D is represented by a pair of numbers (its horizontal and vertical components).
- Taking the difference of two points (componentwise) yields a vector.
- Vectors can be added and scaled (componentwise). Points should not be added or scaled.
- Adding a point and a vector (componentwise) yields a new point. This operation can be used to animate moving points.
- Lecture examples - Timer Control, Formula Control

- More examples - Drawing Vectors

Distance computations — Collisions and reflections

- Mathematically, the distance between two points (x_0, y_0) and $(x_1, y_1$ is $\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$.
- The distance between a point and a circle and the distance between two circles follows from this formula.
- Given the three numbers a, b, c, the set of points (x, y) that satisfy the equation $ax + by + c = 0$ is a line. The distance from this line to a fixed point (x_0, y_0) is $\frac{1}{\sqrt{a^2+b^2}}(ax_0 + by_0 + c)$.
- The distance from a circle to a line is the distance from the center of the circle to the line minus the radius.
- Lecture examples - Collisions and Reflections
- More examples - Drawing Vectors

Reflections — Collisions and reflections

- Mathematically, the direction of reflection for a ball bouncing off of a wall depends on the incoming velocity vector v and the normal vector to the wall at the point of contact.
- The incoming velocity vector can be decomposed into $v = v_p + v_n$ where v_n is the component of v orthogonal to wall and v_p is the component parallel to the wall.
- In this model, the reflected vector is $v = v_p - v_n$.
- This model simplifies for horizontal and vertical walls. In particular, reflection simply negates one component of the velocity vector v.
- Lecture examples - Collisions and Reflections
- More examples - None

Keyboard events — Keyboard input

- SimpleGUI supports two event handlers for keyboard events.
- The key down event handler is registered via `set_keydown_handler`.
- The key up event handler is registered via `set_keyup_handler`.
- The variable passed to each of these handlers is a number that can be compared at the constants in `KEY_MAP` to determine which key has been pressed.
- Lecture examples - Echo

- More examples - Shape Selection

Positional control — Keyboard input

- In Python, we can control the position of a point `p` directly using key board events.
- The horizontal and vertical components of a point `p`'s position can be increment/decrement via `p[i] += c` in response to key presses.
- Lecture examples - Ball Position
- More examples - None

Velocity control — Velocity control

- Mathematically, basic physics relates the position of a point p and its velocity p via the equation $p+ = v * dt$ where dt is a small time step.
- In Python, we can control the motion of the point `p` via keyup/keydown events that increment/decrement the two components of the velocity vector `v` via `v[i] += c`.
- Lecture examples - Velocity Control
- More examples - Ball Track

Mutable vs. immutable data — Programming tips #4

- Numbers, Booleans and strings are *immutable* and can not be modified. Only new copies of these kinds of data can be made.
- On the other hand, parts of a list can be *mutated* via assignment to individual elements of the list.
- Assignment of an entire list to a variable generates a *reference* that refers to the list. Subsequent assignment may generate multiple references to the same list.
- Mutating a list with multiple references modifies all references to the list. This capability is very useful, but can generate subtle errors.
- Lecture examples - Global, Tuples
- More examples - List Structure