**Booleans, Conditionals, & Errors**

*Logical operators:* and, or, not

*New math operators:* % (mod), // (div)

*Short circuit evaluation:* Python only evaluates the second half of a logical operation if it needs to

*Conditional statement:* control structure that allows you to make choices in a program.

**if** booleanExpr**:**

 *ifBody*

**elif** booleanExpr**:**

 *elifBody*

**else:**

 *elseBody*

*Syntax Error:* an error that occurs when Python cannot tokenize or structure code. Examples: SyntaxError, IndentationError, Incomplete Error

*Runtime Error:* an error that occurs when Python encounters a problem while running code. Examples: NameError, TypeError, ZeroDivisionError

*Logical Error:* an error that occurs when code runs properly but does not produce the intended result. Often (but not always) caused by a failed test case with AssertionError

**assert(**funName**(**input**)** **==** output**)**

**Circuits and Gates**

*Circuit:* a hardware component that manipulates bits to compute an algorithmic result. Can also be simulated with an abstract version.

*Gate:* an abstract component of a circuit. Takes some number of bits as input and outputs a bit.

*Gates:* **∧** (and), **∨** (or), **¬** (not), **⊕** (xor); also nand and nor (no special symbols)

*Gates (in circuits):*

and: or: 

not: xor:

nand: nor: 

*Truth table:* a table that lists all possible input bit combinations and the resulting output for a particular gate or circuit

*Half-adder:* a circuit that takes two one-digit binary numbers, adds them, and outputs two digits as the result

*Full adder:* a circuit that takes two one-digit binary numbers and a carried-in digit, adds all three, and outputs two digits as the result

*N-bit adder:* a circuit that takes two n-bit numbers, adds them together by chaining together n full adders, and outputs a n+1-digit result

**While Loops**

*While loop:* a control structure that lets you repeat actions while a given Boolean expression is True

**while** booleanExpr**:**

 *whileBody*

*Infinite loop:* a while loop that never exits due to the state of the program

*Loop control variable:* a variable used to manipulate the number of times a loop iterates. Requires a start value, update action, and continuing condition.

input(msg) - prints msg, lets the user type a response, then returns the response as a string

**For Loops**

*For loop:* a control structure that lets you repeat actions a specific number of times

**for** var **in** range**(**rangeArgs**):**

 *forBody*

*Range:* a function that generates values for the loop control variable in a for loop. Can take 1-3 inputs.

range(end) # [0, end)

range(start, end) # [start, end)

range(start, end, step)

# step provides the increment

**Looping over Strings**

*Index:* access a specific value in a sequence based on its position. Positions start at 0 and end at len(seq)-1. Non-existent indexes result in IndexError.

strExpr**[**index**]**

*Slice*: access a subsequence of a larger sequence based on a given start, end (not inclusive), and step

strExpr**[**start**:**end**:**step**]** # slice

strExpr**[**start**:**end**]** # also slice

# default to 0:len(strExpr):1

*Looping over strings:* use range and indexing to access one character at a time.

**for** i **in** range**(**len**(**strExpr**)):**

 *something with strExpr[i]*

**General Control Structures**

*Control flow chart:* chart that designates how a program steps through commands. Uses branches for conditional checks and arrows leading back to previous commands for loops.

*Nesting:* a control structure can be included in the body of another control structure through use of indentation.

*Nested loop:* a loop with another loop in its body. The inner loop is fully executed for each iteration of the outer loop.