CARLETON COMPUTER
SCIENCE SOCIETY

Comp1405 Final Review Session

Variables

Valid Variable Names

- only letters, numbers & underscores
- camelCase and snake_case

Variable types

- Integer
- Floating point
- Boolean
- String

Algebra Operators

Operator	Function	Example (If x = 10, and y = 8)
+	Addition	x+y = 18
-	Subtraction	x-y = 2
/	Float Division	x/y = 1.25
//	Integer Division	x//y = 1
%	Modulus	x%y = 2
*	Multiplication	x*y = 80
**	Exponent	x**y=100,000,000

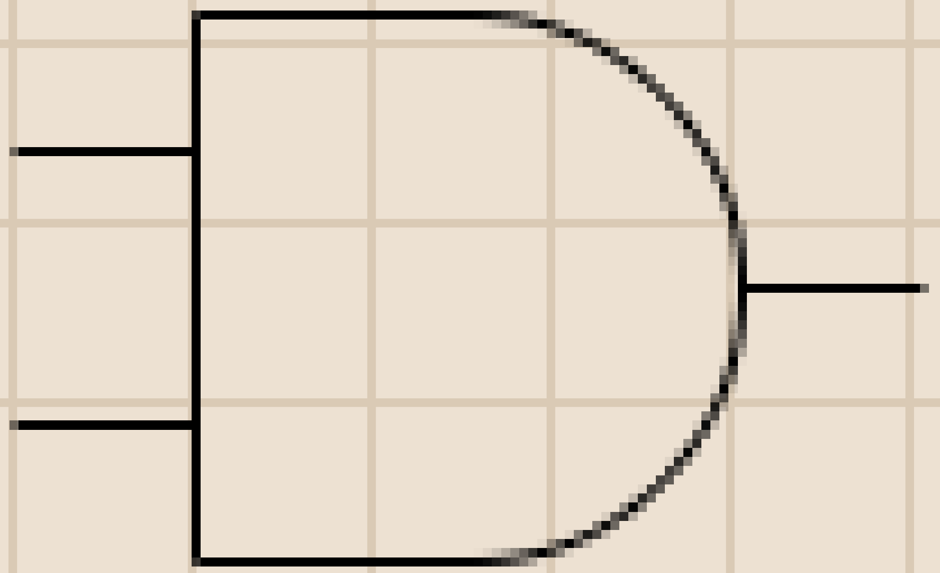
Conditional Operators

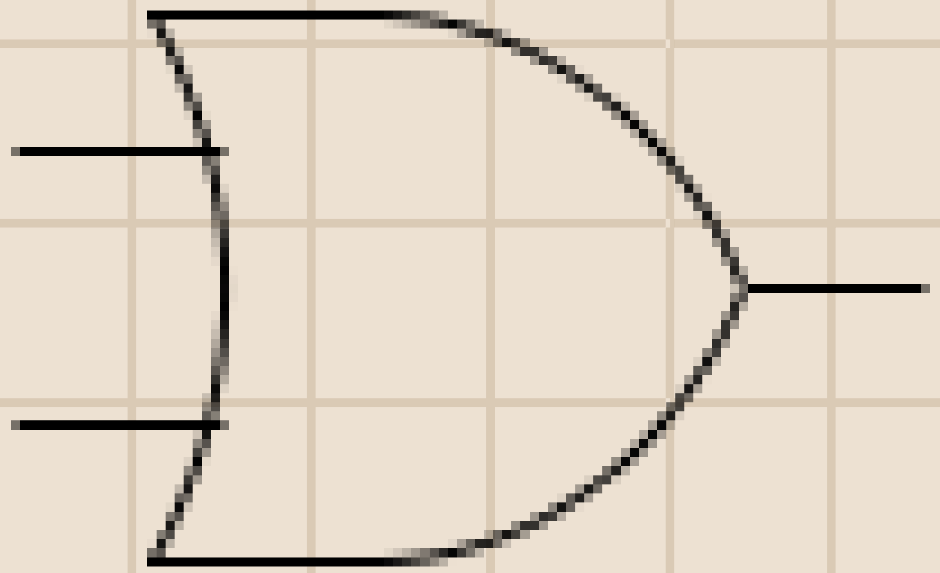
Operator	Function	Example (If x = 6, and y = 7)
==	Equal	x == y returns false (6 does not equal 7)
!=	Not Equal	x != y returns true (6 does not equal 7)
>	Greater Than	x>y returns false
<	Lesser Than	x<y returns true
>=	Greater or Equal To	x>=y returns false
<=	Lesser or Equal To	x<=y returns true

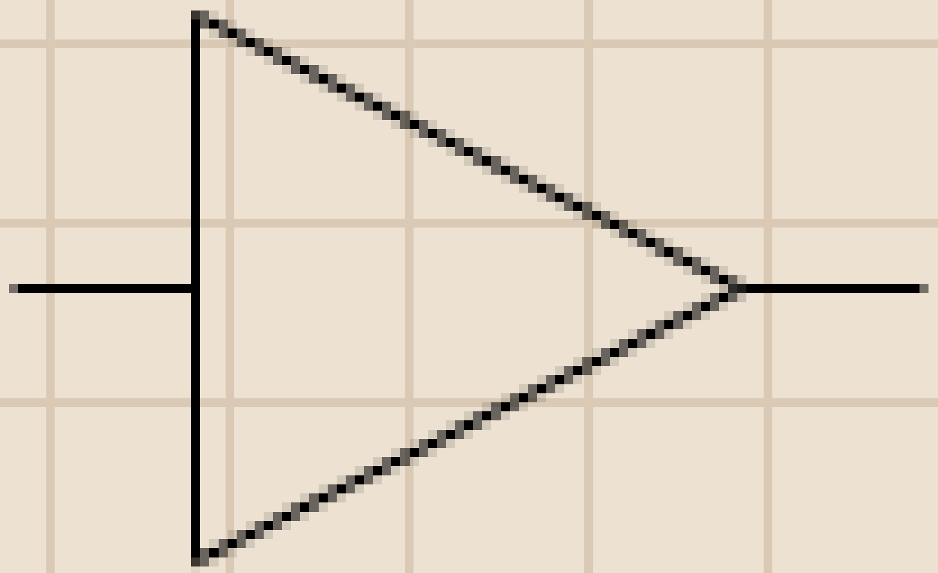
Logic Operators

Operator	Function	Example (If x = true, and y = false)
and (conjunction)	Only returns ‘true’ if both conditions are ‘true’	x and y -> false
or (disjunction)	Returns ‘true’ when one of the conditions is ‘true’	x or y -> true
not (negation)	Returns the opposite of the current value	not x -> false not y -> true

Logic Gates


AND


OR


NOT

Lists

- Collection - that may contain the same data type

my_list = [1, ‘a’, ‘b’, True, 2, 3, 5, 8]

- Accessing elements - each value has it’s unique index, starting at 0

my_list[0] -> would return 1

List Methods

- my_list.append(x) - adds the element to the back of the list
- my_list.insert(i, x) - adds the elements at position i
- my_list.remove(x) - removes the given element
- my_list.pop(i) - removes the item located at position i and returns it
- my_list.pop() - removes the item at the end of the list
- my_list.clear() - removes all of the elements from the list

Strings

Collection Data Type

- Strings are character sequences that must be written in quotation marks, that can be treated as linear (iterable) collections of characters.
- Strings are lists!
- Example: “Hello, World!”

String Methods

- len(str) returns the length of the string.
- Similar to lists, characters can be accessed using their index value and spliced using the splicing operator.
- str.split(sep) splits a string into a list at the specified separator (whitespace by default)
- str.lower()/str.upper() convert the string to lowercase and uppercase characters respectively

List & String Operators

Concatenation

```
greeting = "Hello" + " " + "World!"
print(greeting) # Output: "Hello World!"
```

Repetition

```
line = "ha" * 3
print(line) # Output: "hahaha"
```

Membership

```
print("cat" in "concatenation") # Output: True
```

Dictionaries

A dictionary is an associative collection of items, i.e. it is a collection of keys with values associated with them.

<u>Components of a dictionary:</u>

- Key - A unique, immutable identifier with a value associated with it.
- Value - What is stored in association with a key.

Dictionary Methods

- **my_dictionary.keys()** - returns a list of the dictionary’s keys
- **my_dictionary.values()** - returns a list of the dictionary’s values
- **my_dictionary.items()** - returns a list containing a tuple for each key-value pair
- **my_dictionary.get(key, default_value (optional))** - returns the value associated with the key, or the default_value if the key does not exist (useful to avoid a KeyError)
- **my_dictionary.pop(key)** - removes the element with the specified key from the dictionary and returns its value

Dictionary Operators

Adding an element - my_dictionary[key] = value (if the key already exists, the value is rewritten.

Accessing a value - value = my_dictionary[key] (if the key does not exist, a KeyError is raised)

Deleting an element- del my_dictionary[key] (if the key does not exist, a KeyError is raised)

Check if a key is present - key in my_dictionary, returns True if the key is present in the dictionary

Loops

Loops are used when algorithms need to repeat a certain block of code.

- There must always be a condition that is tested at every iteration of the loop, and one terminating control flow path (so that the loop ends eventually)

Loops (cont.)

Components of a loop

- Condition - source of the boolean value which determines if loop will continue
- Body - block of code that is to be repeatedly executed
- Initialization - phase where initial values are assigned to crucial variables for the loop
- Termination - the loop terminates after the final execution of the body

Functions

Self-contained algorithms which execute a specific task.

Parts of a function:

- ‘def’ statement - used to define a function. Followed by the function name, a set of round brackets ‘()’, and (if applicable), a comma-separated list of parameters.
- Parameters - variables used as input values.
- Return value - The value that a function call evaluates to.
    - Set using the ‘return’ keyword (program also moves out of the function upon a return statement).
    - If no return value is specified, function returns None (None is a type representing the absence of a value).

Functions - default values

Given the following line:
def addThreeNumbers(a, b, c = 10)

- **a** and **b** are <u>mandatory</u> - no default value
- **c** is optional, as a a default value has been specified
    - If it is not provided, the function will use 10
- Order of parameters matters – mandatory parameters <u>must</u> come before the optional ones

File I/O

Python interacting with files stored on your computer.

The 3 most common modes:

- “r”: Opens the file for reading, from the top.
- “w”: Creates the file if it does not exist,
  - If it already exists, it will be overwritten.
- “a”: Creates the file if it does not exist.
  - If it already exists, it adds new data to the bottom of the file.

File I/O (cont.)

- **f = open(filename, mode)** - Opening a file
- **f.readlines()** - Reads entire file and returns a list, each element being a line (string)
- **f.readline()** - Reads one line (string), including the newline (“\n”)
  - Can loop, calling at every interation, avoiding the use of a list
    - **while line != “”** or for line in file
  - Returns an empty string when it reaches the end of the file.

Recursion

A recursive function is a function that calls itself. This can be done to improve the program’s runtime.

<u>Steps needed to solve a recursive problem:</u>

- Simplify the given argument
- Make the recursive call
- Do the operations required at each step to reach a solution

Make sure that your recursive function - has a way to end, otherwise.. StackOverflow error

Sorting

- **Selection sort** - Finding smallest elements, and putting them at their respective positions
- **Bubble sort** - Pass through list, comparing the elements that are next to each other and swapping them if they are out of order, repeating until there are no longer any swaps made
- **Merge sort** - Separate the list until it’s in units of two, then sort those units, combining and sorting them with another unit of the same size, repeating the process until the list is sorted

Searching

<u>Must be done on a sorted list!</u>

- **Linear Search** - Searching all of the elements in order, until you find the element that you are looking for
- **Binary Search -** Start at the middle - determine which half must contain the value that you are looking for, continue until you find the target value

Object Oriented Programming (OOP)

<u>Object-Oriented Design</u>

- The practice of procedural programming is concerned with the development of procedures
- The practice of object-oriented programming is concerned with the development of objects
- An object is an entity (i.e., some 'thing') that includes both data and functionality components
- Example: A car -- Color, brand, wheels, etc.

Object Oriented Programming (OOP)

<u>Classes and Objects</u>

- Classes
  - Define a data type, design the structure of objects, defines attributes and methods
- Objects
  - Instances of a class, hold specific values
  - Separate scope per object