CS 8 Notes: Intro to Python

Operator	Action	Operator	Action
/	division, always returns float	set( )	empty set
//	division, drops remainder	set | set2	combines 2 sets
%	remainder	set & set2	(intersection) returns elements in both sets
**	power	set - set2	removes elements in set 2 in set 1 (elements not in set 1 won't be removed)
a in b	returns True if a is in b (i.e., substring, element)	set ^ set2	(symmetric difference) unique elements in both sets
{ }	dictionary (empty)	set < set2	returns True/False based on whether set is a subset of set2

Function Name	Parameters	Action
type( )	1	Returns data type of parameter
print( )		Prints parameters
min( )		returns minimum of inputted parameters (min value if 1 list inputted, or parameters are all values)
max( )		returns maximum of inputted parameters (max value if 1 list inputted, or parameters are all values)
len( )	1	returns the length of the parameter
int( )	1	converts into integer type
float( )	1	converts into float type
sum( )	1 - list	returns the sum
reverse( )	1 - list	returns the list in reverse index
lyst.count( )	1	returns the count # of occurrences in list

lyst.index( )	1	returns the index of first occurrence in list
lyst.remove( )	1	removes the index of first occurrence in list
lyst.sort( )	0	sorts alphabetically (Strings) or increasing values (int/float); cannot mix
input( )	0	allows keyboard user input in a String
lyst.append( )	1	adds parameter to the end of the list
lyst.pop( )	0 or 1 - int	remove the element of parameter in the list; (end of list if no parameter)
namedtuple._replace ( )	1	allows changing of attributes of name tuples by returning a new namedtuple object
range( )	1,2, or 3	creates a list of the specified bounds (exclusive upper bound if 1 parameter); third argument is the step counter
String.split( )	0 or more	splits a string into a list of strings separated by white space (space or new line); parameters could define separation instead
String.strip( )	0 or 1 (String)	removes the whitespace (or whatever is inside the string sequence) starting at the beginning and end of Strings (stops if the index is not symmetric backward and forward
String.find( )	1 (String)	returns the index of the first character that matches the exact String parameter; -1 if parameter is not in the String
String.startswith( )	1 (String)	returns True if the String starts with the parameter; else, false
String.endswith( )	1 (String)	returns True if the String ends with the parameter; else, false
String.count( )	1 (String)	returns the # of times that the parameter is within the String
String.replace( )	2 (Strings)	returns a string where all substrings that match the first parameter with the second
String.lower( )	0	returns the String changed to lower case
String.upper(	0	returns the String changed to upper case

random.randrange( )	1, 2, or 3 (int)	1 parameter - exclusive int upper bound; 2 parameters - inclusive int lower bound; 3 parameters - step
read( )	0 or 1	0 parameters - reads the entire file into a string (good for small data); 1 parameter - reads n characters from the file (better for larger data; connections keep track of n, it doesn't reset)
readline( )		read everything until the next "\n" character or the end of the file in a String
readlines( )		reads all lines in the file and puts every line into a list (newline characters are kept)
open( )	1 or 2	1st parameter - file name; 2nd - "r" = read only, "w" = writing; "a" = append
Dictionary.pop( )	1	returns the value associated with the key passed in the parameter
Dictionary.update( )	1	updates the first dictionary with the dictionary passed in the parameter; if a new key exists, it gets appended
Dictionary.get( )	1	returns the value associated with the key; otherwise, none
Dictionary.keys( )	0	returns a dictionary view which acts like a list (can be turned into a list of keys to be accessed)

Traditional Computer System:

- can do arithmetic, copy, store, and manipulate data
- processor: does computations
- memory/storage: contains info to act on/info to execute
- peripherals: attaches to computing unit (i.e., keyboard, mouse)
- software: instruction processor follows
- computer program: a set of instructions a computer can understand and follow
- algorithm: step by step procedure to do something
- source code/program: text containing algorithms a computer can understand

IDLE:

- Interactive Shell: uses REPL (Read Evaluate Print Loop); cannot save
- Use ".py" files to save work

Variable Name:

- must start w/ letter or underscore
- consist only of letters, numbers, or underscores
- case sensitive

Multiline Comments: ''' ''' or """ """

String: collection of characters

String Formatting:

- when there are { }, the parameters in the .format( ) function will be inputted into them in chronological order
    - if less { } than parameters in .format( ) the last n parameters are ignored; if more { } than parameters, error
    - {n: } specifies the nth index in .format( ) that will be substituted in the { }
    - { :n} defines that the field will occupy n spaces
        - if the field needs to occupy more than n spaces, it will overflow
        - numerical types (and booleans, which are evaluated to numerical values) are right justified
        - String type is left justified
        - {:<n} to left justify; {:>n} to right justify
    - {:.nf} displays numerical values with n precision after the decimal
    - {:.n) displays float values with n significant figures
- Format specifiers: specify a type value to put in the { }; otherwise, error is thrown
    - s = String; d = int; f = float

Escape Characters:

- \n : new line (implicit by returning on keyboard); len("\n") = 1

Lists:

- collection of values; can have duplicate values, different types
- mutable type; when passed into a function a copy ISN'T MADE, so it will change within a function and even after the function returns

Syntax: grammar (true, but can't work)

Semantics: meaning (can work, but false)

Tuples:

- immutable lists
- defined with parenthesis instead of brackets
- Namedtuples: way to package heterogeneous things into a multi-attribute item
    - can represent more complex data into a single type
    - can access attributes by names
    - can be updated by redefining

Print( ):

- In python3, print( ) is a function
- Comma in between parameters add space between concatenation
- default separation = " "; can change using sep="" to remove spaces, for example
- print("\"hi\""); \ is the escape character to print the quote as a character in the String
- default end="\n" for for loops; can change end to remove new lines, for example

Object-Oriented Programming:

- create own types with own attributes
- can modify attributes over time

Pytest:

- assert : creates a test that causes an error if False and breaks the code (unit testing)
- test functions must be defined with test_

Misc:

- Python doesn't check for data types; values dictate type
- executes line by line
- evalues right and assigns left
- negative indexes go backward
- variable[a:b]; a and b are the bounds of the index, where a is inclusive and b is exclusive
- False = 0; True = 1
- can only concatenate String with Strings
- there are no constants/final; ALL CAPS is stylistic constants
- \ backslash extends statement into next line
    - can use triple quotes to function as a multi-lined String
- when a function doesn't return a value, it returns none
- Python doesn't restrict parameter type when passed through
- Stub: a value that is a placeholder that is incorrectly
- help( ) is a help manual; returns the comment within a function, if any
- sentinel value: show a value that shouldn't occur

Conditionals:

- customization of flow of execution

For Loops:

- for VARIABLE in COLLECTION:
        STATEMENTS
- Collections: strings, lists, etc
- if the collection has multiple variables, the for loop must have that number of variables to correspond as such

While Loops:

- continue: jumps to the next iteration of the loop without finishing the execution of the previous iteration
- break: breaks out of the loop
- pass: doesn't do anything; used for legal syntax purposes

Nested Control Structures:

- controls statement execution by having different blocks of code defined by indentation

User Defined Modules:

- importing code not in the standard library
- if __name__=="__main__": ;
- importing modules easier and not include code that may not want to be executed
- imports functions but not test functions nor print statements
- when file is not run directly, the name isn't the main file and anything nested in that statement is false
- can import a specific thing using "from ____ import _____" ; changes the reference
    - when only importing must use filename.function
    - otherwise, no file name needed

Accumulator Pattern

- update a variable in a loop by some sort of counting event

Table

- two dimensions, row and column

File I/O (Input / Output)

- Files: documents
    - gives persistence (allows data to be saved between program execution)
    - read, write, and stored in many different formats
- Plain Text Files
    - characters represented in ASCII (American Standard Code for Information Interchange) -- each character is 8 bits (1 byte) of data
    - 1 bit = one 0 or one 1 (n bits give 2^n possible combinations)
- UTF-8 recognizes all characters in many languages
- Directory: folder
    - Working directory: current directory
- File System: collection of all files and folders on the computer
- File I/O
    - 1. open the file to create a connection between the program and the file and choose if the connection will read, write, or append the file
    - 2. read / write the data
    - 3. close the file / connection (get rid of excess memory)
    - for a_line in file: (a_line represents a line in the file)
    - Write
        - has no implicit new line character
        - deletes all previous text in the file

Dictionaries

- table with a key which maps to a value
    - gives more precise indexing (i.e., dict[“key”]) where key is an index of immutable type hashed to the value
    - increases performance search; provides direct access
    - {key:value, key: value, etc}
    - can access keys by accessing individual elements (i.e., using for loops)
    - keys must be unique; otherwise, last key definition is kept

Views

- windows to content of dictionary
- self-updating and shows what current state is

Sets

- duplicates are ignored
- order and position doesn't matter; can access directly because it returns the index based on a hash; then, check if it exists
- dictionary without associated values (no colon and corresponding value, only keys)
- proper subset is a subset smaller than the initial set

Performance Benchmarking

- time import has perf_counter (gives timestamp when called)
    - call before and after to determine time elapsed
- dictionaries are much faster than searching through lists