Python

Reserved Words:

* false: Is a Boolean, is assigned to a variable.
* none
* true: Is a Boolean, is assigned to a variable.
* and
* as
* assert
* break: skips the rest of the loop and exits it.
* class
* if
* def
* del
* elif
* else
* except
* return: Simplifies the use of print in functions. Everything after this word inside a defining function is going to be ignored.
* for
* from
* global
* try
* import
* in: works as a logical operator, when used outside an if or for, give a Boolean.
* is
* lambda
* while
* not
* or
* pass
* raise
* finally
* continue: skips the rest of the loop and goes back to the top
* nonlocal
* with
* yield

Operators: In order of priority of operation

* Parentheses are always respected
* \*\* power
* \*, / and % (Remainder)
* + or -
* Left to right.

Comparison operators:

* <
* <=
* == Equal to
* >=
* >
* != Not equal
* = is for assignment

Try/Except:

Used as if/then statement but to avoid python from stopping in the middle of the code and showing an error.

Store and reuse (Defining functions):

def thing()[in this space there could be more parameters]:

here goes what the thing() does

Libraries: First goes the string and then .library.

.Lstrip(), .rstrip(), .strip(): Remove whitespaces on the left, right and both sides (respectively).

.find() returns the position of the input string. If not found, returns -1.

.startswith() returns Boolean.

Opening a file:

open() is the function to get a file from the computer and returns a “file handle” used to manipulate the file.

Ex: handle=open(filename,mode), there, filename is a string and mode is optional and should be ‘r’ if we are going to read the file and ‘w’ if we are going to write to the file.

New line:

Is a special character that indicates the end of a line.

It is \n and is still treated as one.

When a file handle is opened for read, it can be treated as a sequence of strings

So the for will be used to read each one of the lines or each sequence.

.read() function will put everything into a var including the \n. Although when you print it, the \n will disappear and show the lines one below the other.

.rstrip() eliminates the \n at the end of each line because in the original is considered as a whitespace.

Algorithm: Set of steps or rules used to solve a problem.

Data Structures: A particular way of organizing data in a computer.

List, dictionaries and tuples are data structures.

Lists are made by square brackets and elements separated by commas.

A list element can be any python object, even another list.

Lists can be empty.

Range() returns a list of numbers that range from zero to one less than the parameter.

Lists can be concatenated.

Lists can be created typing x = list()

Dir() shows all the libraries that can be applied to the object being studied.

Split() turns a string into a list. Inside the parentheses one can specify the delimeter.

Guardian: Is a piece of code before a risky instruction or compounded into the risky instruction that works as a debugging device.

Dictionaries: A bag of values, each with its own label.

Dict() creates a dictionary and items are added as follows:

It can also be created like this: ex={ } o ex = {“elmt1”: 1, “elmt2”: 2}

Purse=dict()

Purse[“money”]=2

Money is the key and 2 the value for that key.

Calling a key that is not in a dictionary gives traceback.

Method .get() allows us to check for a key in a dictionary and give a default value if it is not there. No traceback.

X=count.get(name, 0)

If name is not found in count, x becomes 0.

Tuples:

They are in a way the same as a list with the difference that tuples are immutable.

They are created with () instead of [].

Tuples are more efficient that lists because python does not have to allocate memory to create those since they are immutable.

Tuples are better as temporary variables.

The method .items() returns a list of (key, values) tuples.

Python expects to have tuples on both sides of a commad.

Short expression to sort tuples by value:

List Comprehension:

Print( sort ( [ (v,k) for k,v in dictionary.items () ] ) )

That is a list comprehension that creates a dynamic list. In this case, we make a list of reversed tuples and then sort it.

Another example is:

print(sum([int(i) for i in re.findall('[0-9]+'**,** fh)]))

That basically means; apply the int() function to the iterative variable “i” in the for loop of re.findall that looks for one or more numbers in the fh file.

For each loop it sums the values and then prints the result.

Sorted(item, reverse=true). The last part sorts from high to low.

Regular Expressions: These things look goooooood

To be used, they must be imported “import re”.

Re.search() shows if a string matches a regular expression. Similar to use find() method.

Re.findall() extracts portions of a string that matches the reg expr. Similar to combine find() and slicing [:].



For example:

^X.\*: (the semicolon is part of the expression) looks for lines that start with capital X, followed by any number of characters.

Depending on how “clean” the data is and the purpose of the app, the search can be narrowed:

^X-\S+: (the semicolon is part of the expression) looks for lines that start with capital X, followed by a dash, and matches any non-whitespace character then one or more times.

Networked Technology:

TCP Connections / Sockets

In computer networking, an internet or network socket is an endpoint of a bidirectional inter-process communication flow across an internet protocol-based computer network.

Port numbers: Is an app-specific or process-specific software comms endpoint.

Allows multiple networked apps to coexist on the same server.

There is a list of well-known TCP port numbers.

Port 80 is for the web server and the 443 is for the HTTPS protocol.

How to tell Python to connect:

Import socket

Mysock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

Mysock.connect((‘host name’, port))

This above only dials the phone but does not send any information.

The following is an example of a line to read through data in a website.

While True:

Data = mysock.recv(512)

If (len(data)<1):

Break

Print (data.decode())

The library urllib does this all much simpler.

Import urllib.request, urllib.parse, urllib.error.

Fhand = urllib.request.urlopen(‘url to retrieve’)

For line in fhand:

Print(line.decode().strip())

Unicode Characters:

Each character is represented by a number between 0 and 256 stored in 8 bits of memory.

The ord() function tells us the numeric value of a simple ASCII character.

Web Scraping:

Is when a program or script pretends to be a browser and retrieves web pages, looks at them and extract info and then does the same with other pages. The popular web crawling.

Why?:

* Pull data (social data).
* Get your own data out of some system that has no “export capability”.
* Monitor a site for new information.
* Spider the web to make a database for a search engine.

Data on the Web:

Data is shared through the internet in the XML and JSON formats.

XML is the most robust method and JSON the most flexible.

Comms between programing languages is done through these two methods.

XML:

It basically has start tag, end tag, text content, attribute (same as html this one goes inside the <> of the tag) and self-closing tag. The last closes in the same line.

White spaces do not matter, only matter when they are in the middle of a text area.

Besides of that, the indentation is used to make it easier to read.

XML Schema:

Description of the legal format of an XML doc.

Often used to specify what one of the systems accepts as an XML doc that will conform to a particular Schema.

How to read the XML in Python:

By writing ‘’’ I am able to write a string in multiple lines and still be in the same var.

Import xml.etree.ElementTree and assign it an alias.

Import xml.etree.ElementTree as ET.

When reading XML this helps form a tree in which I can easily reach for the data of interest with methods that are part of the function.

Findall as used before, will make a list of the tags, using the form (‘users/user’).

The find method in item.find(‘name’) only makes python to look out for the tag ‘name’ in the list element, the .text must be added to let python know that we want the text.

ET.fromstring(‘data processed from the web’) to arm the tree.

What I learned from the process of doing the assignment: “Extracting Data From XML”:

1. The process to get the URL and read it is the same as with HTML. You have to import the library, request and open the link and use the .read() method.

This will store the data in a variable as bytes which then the ET library will read and transform into a tree.

Although if you print that variable after read, it will appear as a string, it is not a string. Use the .decode() method to turn it into a string.

1. .findall() method will make a list with all the elements of the tree according to the specified route that is in the parentheses.

So tree.findall(‘.//comments’) gives only one element, which is the parent tag of comments and all of the children that are below comments.

On the other hand using the methods .find() gives me the path to the first element on each branch of the tree.

If I wanted to access a certain branch manually, I would have to use the .findall() method and then use .find() to go deeper in the branch but treating the variable that resulted from the .findall() as a list.

Tree.findall(‘.//comment’) will make a list of all the comments children, for which each one of them contains the name and count tags.

So, if I want to make a list of all the names, I use tree.findall(‘.//name’) and then to print them I would use a for i in (variable that resulted from findall) and inside the indent i.text.

That way, for example, list = tree.findall(‘.//name’) gives me the list of all the name tags and the i.text will give me the value of each of the tags.

JSON:

First import json.

This is sort of a simplified option that it’s easier to use.

One of the differences is that it is presented as a dictionary:

{

'name': 'Chuck',

'phone': {

'type': 'intl',

'number': '+1 734 303 4456'

},

'email': {

'hide': 'yes'

}

}

The .loads() parses the data so we can handle it easily. Converts it into a structured data.

If for example the data is presented in square brackets [], it most probably means that is a list and there are two little dictionaries. So the length of the data will be two once it is parsed.

The difference with XML is that the JSON will be transformed into a dictionary, so the command item[‘element’] is the same we use in a dictionary to call the info from JSON.

The method .getheaders() gives us the headers that urllib truncates.

Object Oriented Definitions:

Class: A template. Look at it like a blueprint that has the characteristics of an object or thing. Describes its nature, what fields or properties it has and its capabilities, what it can do.

Method or message: A defined capability of a class. What it can do or can be done to an object. Like .find(), .sort(), .append().

Field or attribute: A bit of data in a class.

Object or instance: A particular instance of a class. Like the breed of a dog in the dog class. The set of values of the attributes of a particular object is called state.

*# This will be the blueprint for the class.*

**class** PartyAnimal:

*# In this case, it contains a variable.*

    x=0

*# And also contains a function.*

**def** party(self):

        self.x = self.x + 1

        print("So far", self.x)

*# By writing this, it's like using the list() function.*

*# This will store the parameters, attributes and methods*

*# in a variable that we assign.*

an = PartyAnimal()

*# So here, an is an object that contains a variable x*

*# defined to zero and the method .party().*

*# Here we are applying the method that is a*

*# function defined within*

*# the class to the object an.*

an.party()

an.party()

an.party()

Constructor: In a class, is a special block of statements called when an object is created.

Many instances:

In the template, we can create lots of objects and store each distinct one in its own variable.

Inheritance: Is to make a new class but inheriting the capabilities of an existing one. Then, we add our own little bit to make our new class.

So, the new class has everything the old one has and some more.

*# Inheritance*

*# Simply create the class and then when defining*

*# the name of the class, add the parentheses and*

*# put the class you want to inherit in there.*

**class** newclass(oldclass)

Relational Databases:

There model data by storing rows and columns as tables. They can efficiently retrieve data from those tables.

Terminology:

* Database: Contains many tables.
* Relation (or table): Contains tuples or attributes.
* Tuple (or row): Set of fields that generally represents an object.
* Attribute (also column or field): one of possibly many elements of data corresponding to the object represented by the row.

SQL is the language to issue commands to the database.

Create, insert, read or retrieve, delete and update.

Large Projects roles:

* App developer: Builds the logic, look, and feel of the app – monitors it for problems.
* Database admin: Monitors and adjusts the DB as program runs in prod.

SQL Summary:

CREATE TABLE Users(

name VARCHAR(128)

email VARCHAR(128)

)

CREATE TABLE, does what it says and creates a table named users.

name is the field and VARCHAR is a command that is part of the contract and 128 is obligatory to create it.

128 is the length of the field, how much characters can be put in there.

Same for email.

INSERT INTO Users (name, email) VALUES (‘Kristin’, ‘kf@umich.edu’)

It literally inserts in the table Users the values Kristin and [kf@umich.edu](mailto:kf@umich.edu) into the name and email fields.

DELETE FROM Users WHERE email=’ted@umich.edu’

It deletes from the Users table the row/rows that contains the email described.

UPDATE Users SET name=’Charles’ WHERE email=’csev@umich.edu’

This updates the Users table by setting the field name to Charles if the row contains the email csev in the email field.

SELECT \* FROM Users

This selects a list of columns (the \* is for columns) from the table Users.

Condition can be added with the WHERE function.

ORDER BY is a function that after using the SELECT, will sort the list taken either by email or name or whatever the field that you put here is.

Building Data Models:

* Draw a picture of the data objects for the app and figure out how to represent the objects and their relationships.
* Rule: Do no put the same string data twice, use a relationship instead.
* If a thing is in the “real world”, there should be a copy in the database.

Focus on the main purpose of the app. That way you can start somewhere and successfully build the model.

Primary key: Used as the way point to a row in other tables.

Foreign key: Starting point of the arrow that points to the Primary key.

Logical key: It’s the key that is most probably going to be used in a WHERE clause.

That is because we will want to search for a thing in the database that specifically has that key. For example, title in a track database.

Some code:

CREATE TABLE table\_name (

field\_name INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

field2\_name INTEGER, imagine this one is a foreign key

field3\_name TEXT

)

The command on the right to the fields can also be set in the SQLite interface.

It is okay to have replication if it is with numbers.

JOIN:

This operation links across several tables as part of a select operation.

I must tell the JOIN how to use the keys that make the connections between the tables using an ON clause.

Example of the syntax:

Select Album.title, Artis.name from Album join Artist on Album.artits\_id=Artist.id

What we want to see.

The tables that hold the data.

How the tables are linked.

The main table is always the first one, the rest are the ones that are going to be joined.

When joining tables that are related many-to-many, the table that contains the relations should be in the middle.

Data Mining Technologies:

<https://hadoop.apache.org>

<http://spark.apache.org>

<https://aws.amazon.com/redshift>

<http://community.pentaho.com>