Python

Office hours Thursday 3:30 - 5, room 6161 1 park place

D:\Programs\Anaconda\python D:\Coding\Python\FunPython.py

New Python Algebra

// returns int

/ returns float

\*\* is to the power

Veriable names can have letters, numbers, \_, and unicode chars

If starts with \_ means private

Cannot start with a number

Wednesday 1/16

Eval() horrible for security

Docstring is first line in function, string statement is called when help(function) is called

' and '' are same but ''' is used for multi line strings

A variable and function cannot be called before they are defined, they do not exist

Wednesday 1/23

List1 + list2 appends to list1

Can make a 1 item list with [item]

Can make a 1 item tuple with (item,), tuples are like lists except are immutable, elements cannot be assigned after initiation

Can make a set with {list}, sets are like sets in math, are mutable but cannot be indexed

For strings, /' or /'' adds the char to the string instead of ending it

Monday 1/28

format() function

open() returns a file-like object

Need to call close() on any opened file

Read(), read(n), readline(), readlines(), write(s), seek() functions

Raising an exception, raise keyword

Wednesday 1/30

With key word, used for resources that require cleanup

* Is exception proof, file or object is closed automatically

Dictionaries

* Initiated with {} and lists pairs, one key and one data
* Any type for data but only hashable, immutable variable types for key
* Data is accessed with d[key]

2/4

ASCII is stored in a byte, every char is a byte, very limited chars

* Ord('char')
* Char('char')

Python moved to UNICODE, which includes almost every char

* Unicode char number initiated with \u ex '\u0409'
* Unicode transformation format (UTF)

Bite String ex: b'hello/nthere'

* \xNN is python representation of ASCII code < 32 or > 127
* Decode('UTF8') function translates bite string to unicode

Random class funtcions (psuedo random?)

* randrange(int1, int2) returns random int in range
* uniform(int1, int2) returns random float in range
* Choice(list) returns random list member
* Shuffle() function does not work for tuple

CH7

Every function has a namespace, which is the scope of a variable

* Variables made in for loops can be used outside loop, anywhere in the function
* If var is defined in global scope, to refer to same variable in a function and not make new variable, use global keyword, ex global var

Try and Except

* Try block executes lines, stops if exception is raised, then except block executes
* Except (specificError) only catches a specific error

Modules(libraries) are namespaces when imported

* Dir(module) prints all module attributes
* Module functions can be called (after mod is imported) from modName.function()

2/6

Top level module

* Program is top level module when read from script or called in command line
* \_\_name\_\_ is a name that is special to the python language

Method calls

* 'this' in java is usually 'self' in python, refers to the class
* For class function parameters, 'self' must always be first parameter, python philosophy
* All variables in python is public
* Object.\_variableName is soft private notation, interpreter reads the same way
* Object.\_\_variableName makes access more difficult but not hard, language enforced security
* Implicit constructer naturally created but explicit constructors made from \_\_init\_\_ ex: def \_\_init\_\_(self, x = 0, y = 0)
* \_\_varNameOrMethodName\_\_ is interpreted with special significance with special features

2/11 CH8

* Operator '+' is an overload operator, is translated to object1.\_\_add\_\_(object2)
* All operators, comparisons, repr(), str(), len(), <type>() are all overload operators
* Print(object) calls \_\_str\_\_(), if not present it calls \_\_repr\_\_()

2/13

* To overload '==' operator, make class function def \_\_eq\_\_(self, other):
* No abstract classes, inheritance only used for adding functionality
  + To create a subclass, define a new class as class subClass(superClass):
  + Can be multiple super classes for a subclass, for each method call not overridden in subClass, python uses algorithm to search super classes and their super classes to use the first instance of the method
  + super(superClass, self) and super() run the super class method
* User defined exceptions initiated with class myError(Exception):
  + If custom exception is detected, raise myError('error statement')
* Decorators are defined by @decoratorName on line before function name
  + Kind of like static, returns a performs the function with extra steps
  + @dataclasses.dataclass assigns constructor parameters to type annotations stated in class, only works in simple, trivial classes
    - Ex class Point:

X: float

Y : float

Z = Point(4.0, 1.5)

* Type annotations new in python 3.5
  + Ex: def func(x: int, y: int) -> int:
  + Python interpreter still does not type check

2/25

* Rosalind Bioinformatics Armory, BioPython
* Project info
* **Tkinter** and pyside2 libraries for GUIs
  + Pack() positions a widget, default is on top boundary of window
  + grid() makes grid for widgets
  + Label() creates a text label
  + Button() creates a button with an event handler
  + Entry() creates a text box
    - Get(), insert(index, text), delete(from, to) functions

import tkinter as tk

From tkinter Import tk, Label, PhotoImage

Entry = entry()

def insertFunction():

Global Entry

Entry.insert(0, "text")

Root = tk.Tk()

Hello = Label(master = root, text = 'hi')

Hello.pack()

Button = Button(root, text = 'text', command = insertFunction)

Button.grid(row = 0, column = 0)

Photo = PhotoImage(file = 'file path')

Root.mainloop() #creates empty window

* Event-driven programming is based upon an event loop started with mainloop()
  + While true:

Check for environment changes

Run the associated event handler

Redraws components

2/27

* More Tkinter

From tikinter import Tk, Text, BOTH

Def record(event):

"event handling function for key press events"

print('char = {}'.format(event.keysym)) #prints key symbol

Root = Tk()

Text = Text(root, width = 20, height = 5)

Text.bind(<keypress>, record) # connects a keypress with a function

Tkinter Canvas widget

From ikinter import TK, Canvas

Def begin(event):

Global x, y

X, y = event.x, event.y

Def draw(event)

Global x, y

Newx, newy = event.x, event.y

Other stuff…

Frame widget is a frame within the window, can be the "master" of other widgets

3/4

Pyside 2

* Tkinter is not as sophisticated as Pyside 2
* Binding Qt framework (includes other programming languages exposed to

python)

* Qt is cross-platform QUI+ framework written in C++
* Pip is python package manager
  + Pip install Pyside 2
* Pyside 2 example

Import sys

Import PySide2.QtWidgets as qw

Import PySide.QtCore as qc

# app is the event loop

App = qw.Qapplication(sys.argv)

B.qw.QPushButton("Hi")

B.show()

Sys.exit(app.run())

* A Pyside GUI layout can be made into a class
  + QT UI files can create layout for a QT GUI

Class Layout(qw.QWidget):

Def \_\_init\_\_(self):

Self.button.clicked.connect(lambda: print(self.text.text()))

* Signals and Slots

3-6

Good libraries

* Numpy - scientific computing
* cython and numba - fast processing with C injection
* Pyside2 - GUI
* Pillow - image processing
* Pyosundfile - sound processing
* Pandas - big data processing, like R
* Tensorflow - machine learning
* Keras - nuero network

Recursion

* Print(text, end = ' ') does not go to next line for print
* If recursive step calls recursion more than once, running time is exponential
* Generators can act like range function
  + No memory requirements
  + Factorial examples:

Def gfact(cutoff):

N = 1

For I in range(cutoff):

N = I \* N

**Yield** N

For fact in gfact(num):

Print(fact)

Def iter\_words(frame):

With open(frame, 'r') as infile:

For line in infile:

For word in line.split():

Yield word

For word in iter\_words("infile.txt")

Print(word)

3-11

Neural Networks are universal function approximators

* Graph of input and output nodes, defined by the weights
  + Changing the weights is essential to machine learning and getting an algorithm to give the desired outcome
* Forward pass for classification
  + Is activation of network for output
  + Hidden layer state = tanh(sum(input \* weight))
    - Preactivation is sum
    - Activation is tanh(sum), the return of a layer or calculation
    - Tanh() function is matrix multiplication of input and weights
* Backward pass for training of functions
  + Finds change of error at a certain weight in weight vs error graph
  + A Loss function l finds the derivative of a weight given a certain input
    - Uses chain rule, which tensorflow simplifies
    - Mean squared error is common loss function, cross entropy is another but is not protected from certain security attack
  + Small loss is close to accurate, big loss is inaccurate

Numpy

* Easy matrix multiplication, fast processing because it translates into C
* Broadcasting - applying a function to all elements in a matrix (ex. Tanh([,1,2,3]))

3-27

Machine Learning

* Dataset of inputs and labels (desired outputs) used for training
* Training/Optimization: inputs -> weight adjustment -> desired outputs
* Execution: unknown input -> trained network -> desired output

Tensorflow

* Google project, bad documentation and error messages, confusing API, GPU only works with CUDA
* Graphs are run in sessions

Import tensorflow as tf

A = Tf.constant(5.0)

B = Tf.constant(3.0)

C = A \* B

With tf.session as sesh

Print(run(C))

* Tensors represent data
* Placeholders are used for values to be defined at runtime
  + Are manually initialized
* Variables are initially assigned in session with tf.glbal\_variables\_initializer()
* Constants are constant
* Tf does the backpass for us

4-3

Python and Databases

* Sqlite3 is python standard library that provides an API for accessing database files
* Mapreduce is a way to inverse index
  + Ex. Given a list of text files with different words, returns a list of words with files that the word is in

4-8

Python Multiprocessing

* Global Interpreter Lock(GIL) - forces python to process one thing at a time, overcome by pools
* From multiprocessing import Pool

pool = Pool(2)

Res = pool.map(function, list)

* Processing ID for every processor getpid()
* Process consists of storage of application (memory, files, pid, and threads)
* Threads run code

Webscrapping

* User-Agent Spoof tricks host page to retrieve page contents
* From urllib.request import urlopen

Response = Urlopen('url')

Text = response.read().decode()

* HTMLParser() python class
  + Has beginTag, endTag, and data handlers

4-17

* From re import findall #for using regular expressions
* Cython is a super language that all of python is included in
  + Is compiled into C, run from command line or called in other programs
  + Pip install cython and setuptools, need python sources

4-22

* Numba library uses llvm/clang compiler automatically to speed up code
  + Converts what python it can to C
  + Used with decorator @numba.jit(nopython = True)
  + Requires specific format to automatically convert code
  + Numba, cython, and numpy (for matrix math) all speed up code
  + Cython is precompiled manually, but numba is automatically compiled at first running of program