Lecture 1 notes:

CPU

RAM/Main memory(short term memory)

Secondary memory (long term)

I/O

Python isn’t a compiled language

Compilable language produces .exe file, source code -> compiler, produces an executable file (.exe), source code isn’t given to end user. [java, c++]

Python is interpretable language, programmer writes source code -> source code given to end user, they can modify it -> install python interpreter to run source code

Source code is referred to as script.

**CHAPTER 2:**

Functions aren’t methods

Function are essentially static

Comment - $

Nameslots in ram that store temporary information (variable)

Input from keyboard is a string, cant do math on string

No curly braces to define scope, indentation defines scope

String is a alphanumeric value

Don’t have a do while loop.

Range function returns a list eg: range(5) returns a list of 0,1,2,3,4

Hitting a tab or 5 times space isn’t the same thing, your scope wont be the same thing

Sentinel

Input validation loops

Modularized program

Hierarchy chart

pass by value: unidirectional commuication between fucntions

**Positional arguments must appear first**

**Standard library**

**Randint(1,100), randrange(same arg as range), random()-no arguments, random float between 1 and 10, uniform(random float in range user specifies)**

**Random.seed() – default is system time**

**IPO chart- tool to design and document fuctions – offers enough info not to use flowcharts**

**Math module**

**Modularization: group related function in module**

**Module name- no keyword, ends with .py**

**Sequential access-beg to end, cant skip ahead; direct access: jump to any piece of code**

**File object referred by a variable**

**Open modes: string specifying how the files will be opened**

**Can specify alternative path and file name in the open function argument**

* + - **Prefix the path string literal with the letter r**

**Readline method also returns \n**

**Append mode: data written to the end of the file**

**You can only write string to the file (type conversions must) file performs with strings**

**For loop to read lines: for line in file object:**

**Record: set of data that makes item and FIELD**

**Exception; traceback: error message-line info, info about type of exception,**

* **Some exceptions cannot be avoided by careful coding: oen a file to read that doesn’t exist, convert non-numeric string to integer**

**Exception handler: code that responds when exception occurs: try excep**

**Try suite**

**Handler: statements inside the except block**

**No exception-handlers are skipped**

* **Exception object: object created in memory when an exception is thrown**

**Usually contains default error message**

* + - **Example: except ValueError as err:**

**Try except may have optional else clause (appears after all except): executed after try only when no exception thrown**

**Finally: perform cleanup operations, occurs after all except**

**Recursion: have way to limit number of time calls itself, has if else: if true call itself else then return value**

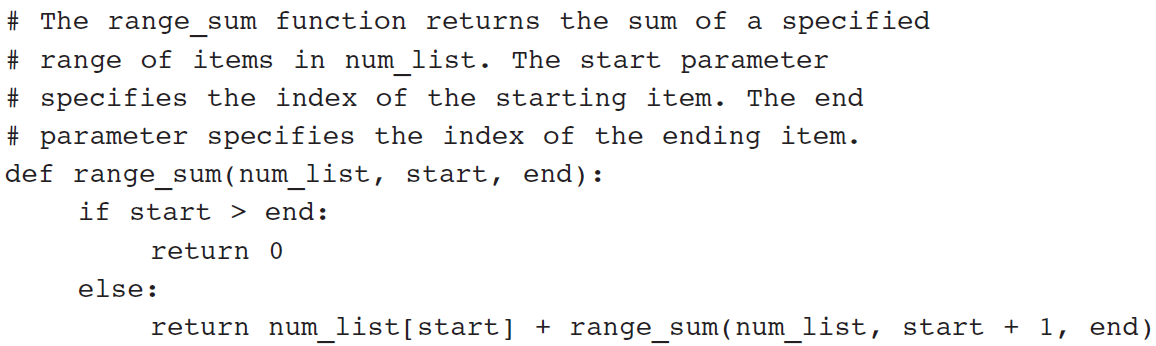
**Depth of recursion:**

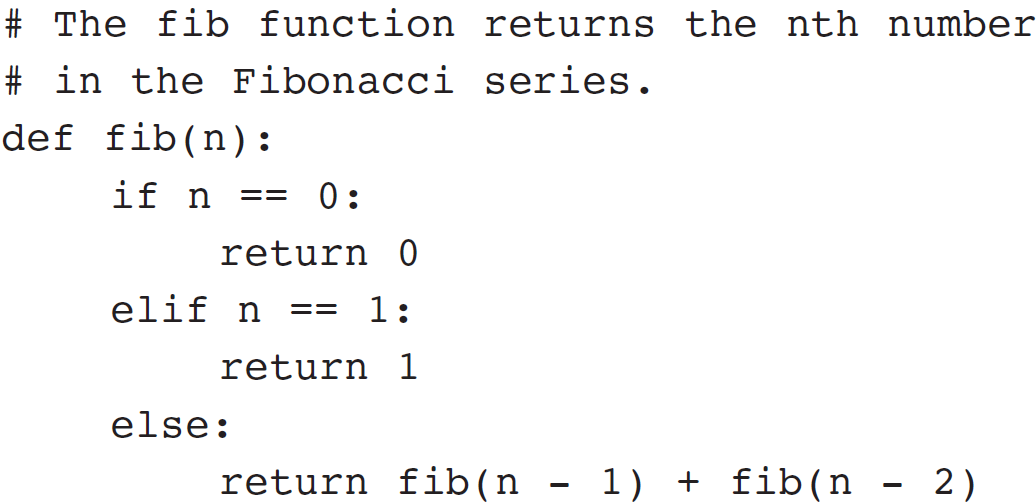
**Recursive problems can be solved using loops; less efficient more overhead**

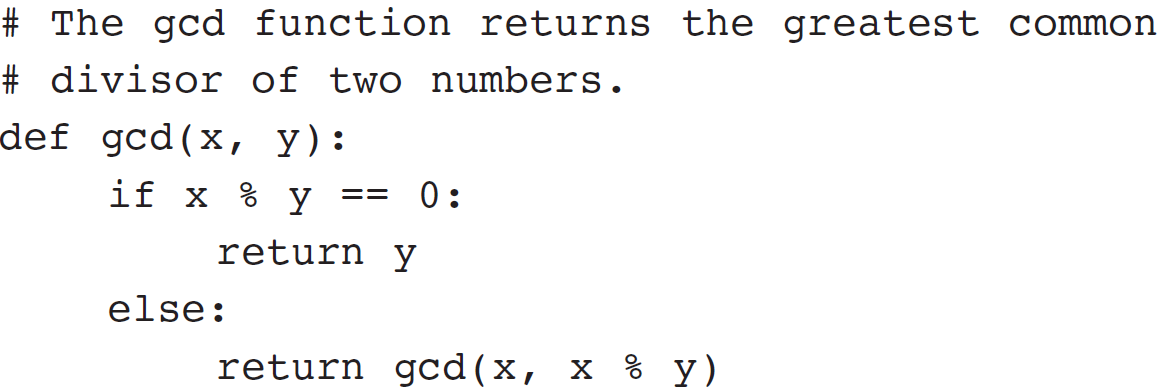
**Base case(solve and return) and recursive case(calls function again)**

* + - **factorial(*n*) = *n* x factorial(*n*-1)**

**types of recursion: direct and indirect**

****

****

****

**Lists and tuples:**

**Sequence- object that contains multiple data items**

**List is mutable and tuple isn’t: they can hold elements of different types**

**List function converts objects to list**

**Repetition operator: \***

**Index- access individual element of list**

**IndexError exception-invalid index is used, so use**

**List can be concatenated with a list only, using + or += operators**

**List slicing: list[start:end]; slicing expressions can have step values, negative index as well**

**Methods: append(item), index(item) = throws ValueError exception**

**Insert(index, item), sort(), remove(item), reverse()**

**Mix(list), max(list) del list[index]**

**Copying list: concatenate old to new, use for loop to insert element by element**

* **To read data from a file use the file object’s readlines method**

**Two dimensional list: nested list, 2 for loops**

**Tuples don’t support: reverse, sort, remove, append, insert and del**

**Processing tuples are faster and tuples are safe**

**Strings are sequences as well, for character in string:**

**Len(str) and IndexError exception- out of range index**

**String concatenation - + and +=, OPERAND ON LEFT SIDE OF += MUST BE AN EXISITNG VARIABLE OTHERWISE AN EXCEPTION IS RAISED**

**Strings are immutable**

**With strings we cant use str[i]= new char [raises an exception]**

* + - **Concatenation doesn’t actually change the existing string, but rather creates a new string and assigns the new string to the previously used variable**

**String slicing also known as substring**

**In operator to check if substring 1 is in string 2, string 1 and 2 can be string literal or variable**

**Tetsing Methods: isalnum(), isalplha(),isdigit(),islower(),isupper(),isspace()**

**String comparisons are case sensitive; lower and upper methods -case insensitive comparisons**

**Modification methods: lower, upper, lstrip, rstrip, strip**

**Endswith(substring), startswith(substring), find(substring) – returns -1 if not found or return lowest index of substring, replace(substring, newstring)**

**Split()- returns list containing words in string, by default space as separator**

**Dictionary: each element is key value pair(mapping of key to values)**

**Keys- immutable objects**

**ELEMENTS IN DICTIONARY ARE UNSORTED**

**Retrieve value: dic[key] raised KeyError exception**

**Test whether key is in dictionary or not using in operator to avoid KeyError exception**

**Dictionaries are immutable objects**

**Add new key value pair: dict[key] = value, if key exists value is replace**

**Delete key value pair: del dict[key] => can raise KeyError exception**

**Keys are immutable objects but can be of different data types, values can be of diff data types**

**Len(dict)**

**Empty dictionary = {} or dict()- built In function**

**For loop to iterate over dictionary=> for key In dict:**

**Delete all elements of dictionary: dit.clear()**

**Dict.get(key, default) =>alternative to [] and avoids keyerror exception**

**Dict.items() => returns all key and value associated with it, returns a dictionary view, each element of dictionary view is tuple**

* + - **Use a for loop to iterate over the tuples in the sequence**
      * **Can use a variable which receives a tuple, or can use two variables which receive key and value**

**Dict.keys() – returns all keys as sequence**

**Dict.**

**Dict.pop(key, default) – returns value associated with key and removes key value pair**

**Dict.popitem()- returns random key value pair and remove the pair**

**Sets:**

**Elements in set are unique and unordered and of different data tpyes**

**Empty set=> set()**

**Non empty set=> set(argument)**

**Argument is object that contains iterable elements can be string, lsit, tuple**

**If argument is string, each char becomes an element**

**if duplicates are there, only one duplicate appears**

**for set of strings, pass a list of string to set function**

**set are mutable objects**

**len(set); add method-add element to set; update- adds group of elements; clear() -removes all elements of set**

**remove and discard method() – pass element to remove, same behavior. When element is not found, remove raises exception KeyError and discard doesn’t**

**for loop and in operator**

* + **union-Both techniques return a new set which contains the union of both sets**

**| = union; & = intersection; difference = -; symmetric\_difference = ^ (contains elements uncommon to both); seta.is subset(setB) or setA <=setB;**

**Picking = serializing an object**

* + **Import the pickle module**
  + **Open a file for binary writing**
  + **Call the pickle.dump function**
    - **Format: pickle.dump(*object*, *file*)**
  + **Close the file**
* **You can pickle multiple objects to one file prior to closing the file**
* **To unpickle an object:**
  + **Import the pickle module**
  + **Open a file for binary writing**
  + **Call the pickle.load function**
    - **Format: pickle.load(*file*)**
  + **Close the file**
* **You can unpickle multiple objects from the file**

**Chapter 10:**

**Procedural programming: program that have functions to perform specific tasks, data items passed from one procedure to another**

**Object: entity that contains data (data attributes) and procedures(methods)**

**Encapsulation- combining data and code into single object**

**Data hiding: object’s data attributes are hidden from code outside the object. Access restricted to the object’s methods**

**Object reusability**

**State of object defined by data attributes**

**Public method-allows external code to modify objct and private methods are for object’s internal working**

**Class- code that specifies data attribute and methods of object (blueprint of house)**

**Instance-object of class(can be many instance of one class)**

**Class definition-set of statements that defines class’s method and attributes**

**Format: class Classname: starts with uppercase**

**Self parameter-every method has this parameter**

**Initializer method():executed when instance of class is created, initialized data attributes and assigns self parameter to object.**

**Format: def \_\_init\_\_(self) [first method in class definition]**

**Create instance of class=> myclass =classname(); instance.method()**

**Reference to self is passed automatically**

**Object data attributes must be private.**

**Classes can be stored in modules., import modules**

**\_\_str\_\_() : displays object’s state [automatically called when object passed to print or str function]**

**Each instance has its own set of data attributes**

**Accessor: safe way for**

**Mutator: change/ store value of data attribute**

**UML-Unified modeling language(depicts oo systems graphically)**

**Inheritance = is a relationship**

* **In UML diagram, show inheritance by drawing a line with an open arrowhead from subclass to superclass**
* **Polymorphism: an object’s ability to take different forms**
  + **Ability to define a method in a superclass and override it in a subclass**
  + **Ability to call the correct version of overridden method depending on the type of object that called for it**

**AttributeError: when method receives object which isn’t instance of right class**

**Isinstance(object, class)**

**UI: parts of comp with which user interacts**

**Command line interface: displays a prompt and user types a command**

**users to interact with a program through graphical elements on the screen**

**dialog box: small window that displays information and allow users to perform action**

**widget-graphical element with which user can interact with**

**button, checkbutton, canvas, entry, frame, listbox, scale(select value by moving slider) scrollbar, message, radiobutton, toplevel, menu (list of menu choices displayed on screen)**

**object oriented approach-build GUI in init method**

**label(1st rarguement refers root widget and 2nd-text that should be displayed)**

**pack method-called once for each widget, position widget; side = top, left, right; positioning depends on order.**

**Button- text to appear on the face of the button; callback function (event handler)**

**Dialog box- tkinter.messagebox.showinfo(title, message) :title bar and main window resp.**

**Quit button-closes the program, root widget’s destroy method as callback function**

**Entry widget is followed by button-callback function extracts data from entry**

**Get method of entry widget-returns string**

**Labels can be used as output fields, blank label widget, displays data in label when button clicked; replaces dialog box**

**StringCar class:can be used along with label, label widget is associated with object of class**

**Radiobuttons are mututally exclusive, only one selected at a time, selected and deselected(not filled).**

**IntVar class used along with radiobuttons; assign unique integer to each radiobutton widget and associate them with same intVar object.**

**Some action to be performed once you click on radiobutton, then use argument command = self.my\_method, no need to click buttons**

**Checkbuttons aren’t mutually exclusive , use checkbutton class; each**

**Libraries-has packages and modules**

**Packages-collection of modules**

**PIP=package management system (installs and manages python packages)**

**PIP comes with python 3.4+**

**Install pip => easy\_install pip**

**Pip install package**

**Pip –help**

**Import module or from package import function or import package.function**

**Request library- web scraping, interafce to interact with HTML**

**Beautifulsoup – process web scraped contents, xml and html parsing library**

**nltk – natural language toolkit; analyze text for data processing**

**re library- regular expression and pattern matching**

**no need to install it, just import it**

**Questions:**

In a print statement, you can set the \_\_\_\_\_\_\_\_ argument to a space or empty string to stop the output from advancing to a new line.

print('The path is D:\\sample\\test.')

True/False: Functions can be called from statements in the body of a loop, and loops can be called from the body of a function.

**Second quiz:**

What is the first step that needs to be taken in order to apply a recursive approach?

Python comes with \_\_\_\_\_\_\_\_ functions that have been already prewritten for the programmer.

**Quiz 3:**

True/False: When a class inherits another class, it is required to use all the data attributes and methods of the superclass.

What attributes belong to a specific instance of the class?

What type of programming contains class definitions?

True/False: An object is a stand-alone program but is used by programs that need its service.

2 out of 2 points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | | | | |
|  | True/False: A mutator method has no control over the way that a class's data attributes are modified. | |  |  |  |
|  |  |  |  |  |

Base classes are required to implement \_\_\_\_\_\_\_\_\_\_\_\_\_

What concept involves a superclass and a subclass?

Quiz 4:

Quiz 5:

In the following code, how to you access the items in the PANDAS series?

s = pd.Series([7, 'Heisenberg', 3.14, -1789710578, 'Happy Eating!'])

What does this code do?

import requests  
from bs4 import BeautifulSoup  
  
def main():  
    html=requests.get("http://www.bigrigg.net").text  
    soup = BeautifulSoup(html, 'html.parser')  
    print(soup.prettify())

r = requests.get('https://api.github.com/events')