

Reserved keywords

Input() – assignment value is given an input call that is given to the interpreter input then gets updated a value/data we put in.

Name = input(“givename”) print(name)

Input returns a string so change it to int if needed. Eg Floor = input(“enter data”) Usfloor = int(floor) + 1. Print(Usfloor) o/p – integer value

“=” is assignment operator and == is the “equal to”. The comparison operators are usually used to solve and asking questions and thereby getting those result in boolean ie True or False.

Also, it is usually used in if else while and other loop operations.

Try / except Structure – surround a dangerous section of your code with try and except. Also, if code in ‘try’ works then except is skipped. If code in try fails it jumps to except section.

2 types of fns – build in (like input(), type(), int()…) and other is the function that we define on our own. “Def” is a function block, it actually “ remembers” the code and it names it anything. So when “thing()” is called/invoked it goes up and runs your code.

Big() – biggest letter

Functions are basically remembering&store and THEN reuse WITH initiation/fn call. Eg max(). In backhand someone already wrote the fn for finding the character with many for loops and other stuff and then reads through other list and etc.

When we return a value in a function, then the returned value it should be assigned a variable so that that data can be used globally in the whole program.

Arguments can be passed inside a function. Eg. Greet(Lang) here lang is a place holder. When we do a fn call ie greet(‘en’) then lang is now a alias to ‘en’ then it is used inside the function. One more thing, basically, what we defined outside the function, that we initialized ie x and name should be used as arguments to be able to play with it.

Python you can explicitly iterate over a set of elements in a list, string or tuple but in other languages you need to provide indexing. Ie for i in [3,4,5] print(i)

Range [1,6] – generates a sequence from [1,2,3,4,5] but if we give print it outputs 1,2,3,4,5 downwards unless \n specified. Which is the same output as for I in [1,2,3,4,5]

Srtirp excess space – greet.strip() takes out space from left and right side of the given word but not inbetween the word.

File -> handle = open(“qle.txt”, ‘r’) ie variable that you store file handle = open file inside file name and opening mode read, write or append.

Variables – have only 1 value in them. So we now introduce, list – is a collection(can put lots of things to organize them like suitcases)

Range – returns a list of rumbers.eg; a= b,c,d. print(range(a)) – [0,1,2].   
String and list link -> a split fn can be used to change a string to list. Eg. Abc = “ ch ma ga” stuff = Abc.split() print(stuff) o/p – list of ch, ma, ga.

List – lienar collection of values that stay in order.(basically organinzed collection Dictionary – a “bag” of values, each with its own label (bunch of flower…..in that flower with tag is ours soo basically its like key and a value) or think like we have to come up with labels and get the thing back up with labels.

List use indices to access element and dictionary uses key value pair. Tuples – are like list but with ( ) while list [ ]. Just that tuples are immutable(cant be altered or changed) ie x = (1,2,3) x(1) = 4 then error.

re.search() – eg if re.search(‘from: ‘, line) : print(line) – so here, within the library regular expression go find the search function and search for the string “from” from the string line.

SOCKET -Import socket.

Socket is like a file handle that doesn’t have any data associated with it yet. eg: sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) then we connect that to a socket to destination across the internet (with domain name and port name). then send http command ie GET,,,,eg cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode() here .encode() is for converting some of the string in python to utfa format. Then after we made the connection we send it. Ie sock.send(cmd).

decode basically does the opposite of encode above ie unicode to python readable string.

Urllib – does all the socket work for us and make web page look like a file. (it’s a library) So…Basically url basically makes the connection, encodes the GET req and then it actually retrieves at moment headers and returns object that looks like a file handle like hand = input()

Eg. fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')

So basically, urllib makes url functins inside python very much like files.

Web scraping – pgm pretends to be a browser and retrieves webpages, extract info and looks at more info.

Soup – from the given html study it. Then given it to an object soup. Eg. soup = BeautifulSoup(html, 'html.parser'). then we call a fn to get all anchor tags ie. tags = soup('a') what gives list of tags. Then if we loop in href inside those tags we get or pull out the text of the href attribute. Ie for tag in tags: print(tag.get('href', None))…

XML - define and store data in a shareable manner. Has a Start tag,end tag, text content, attribute, self closing tag.

We have a built in XML parser in python called element tree. Ie.. import xml.etree.ElementTree as ET. (here as ET is just a shortcut handle to call the element tree)

Tree = ET.fromstring(data) this will read this data from string and give us an object(tree).

We can find using tags so using uaing object.find we can find the tag name. ie.. print('Name:', tree.find('name').text). to get the contents of an attribute we use .get ie.. print('Attr:', tree.find('email').get('hide'))

So, to get attribute we use get(fn) to get the info like name age that we provide which is not an attribute we use find(fn). Findall – to get all the users and find to just get a single user.

JSON – represent data as nested “lists” and “dictionaries”.

Here we take the string from our given input(ie data) then pass it into json library loads which reads it, parses it and looks at all the white spaces and returns us a dictionary which we assign an object like xml we did before ie info. Eg print('Name:', info["name"]). Gives chuck as output.

API – Application program interface. They set the rules, set the urls. Will say if its xml or json.

Objects – are bits of code and data. Object hides details, they allow the rest of the program to ignore the details about “us”.

Class – a template - eg. Dog or a cookie cuter

Method or message – A defined capability of a class – eg. bark()

Field or attribute – a bit of data in a class – eg. length

Object or instance – a particular instance of a class - eg. Bulldog or snoman cookie, apple cookie(BASICALLY MEAN THE SHAPE HERE)

Code we write is in the class and object are the instance. Methods are fns are lives inside the class. Fields are the variables that are defined inside the class.

Dir() – is used to find the capabilities of our newly created class. Object creation – constructor and destruction – destructor.

When an object is created it runs all the variables and then runs the constructor ie…………….. def \_\_init\_\_(self): self.x = 0

Eg. an = PartyAnimal() I want to involved when this object is created – this executes till I am constructed message. Then when created object loses its pointer somehow. Ie by maybe assigning it a value ie. an = 20. Then destructor gets activated (I want to be involved when the object is destroyed) ie def \_\_del\_\_(self): print('I am destructed', self.x)

Constructors can have additional parameters that can be used to set up instance variables for the particular instance of the class.

Inheritance – when we make a new class we reuse the existing class and inherit all the capabilities of an existing class and then add our own little bit to make our new class. Basically called as “subclassing”, were attributes are inherited from their parent class and introduced their own.

So basically inheritance will take the constructor ok parent class and also execute the constructor of current class. Same goes for destructor.