Python Programming Notes

**Part1**

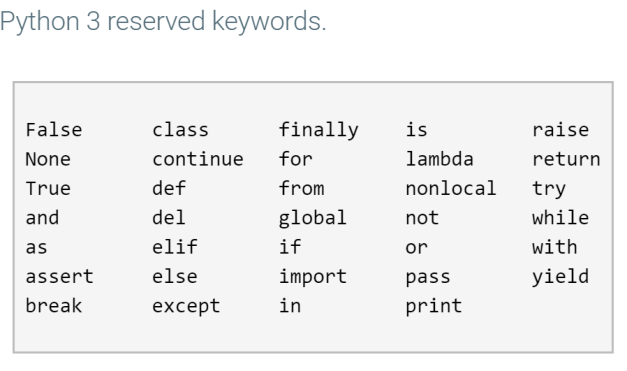
Print('Hello', end=' ') # inline output

int(input('Enter hourly wage: '))

#"Mul|Add|Sub|Div 97, #9, 98" # \*\*the data in memory location 97 by the number 9, storing the result into memory location 98.

Print(, sep=' ') no space in between

print('the area of the white square is: %.2f' %(asmall))

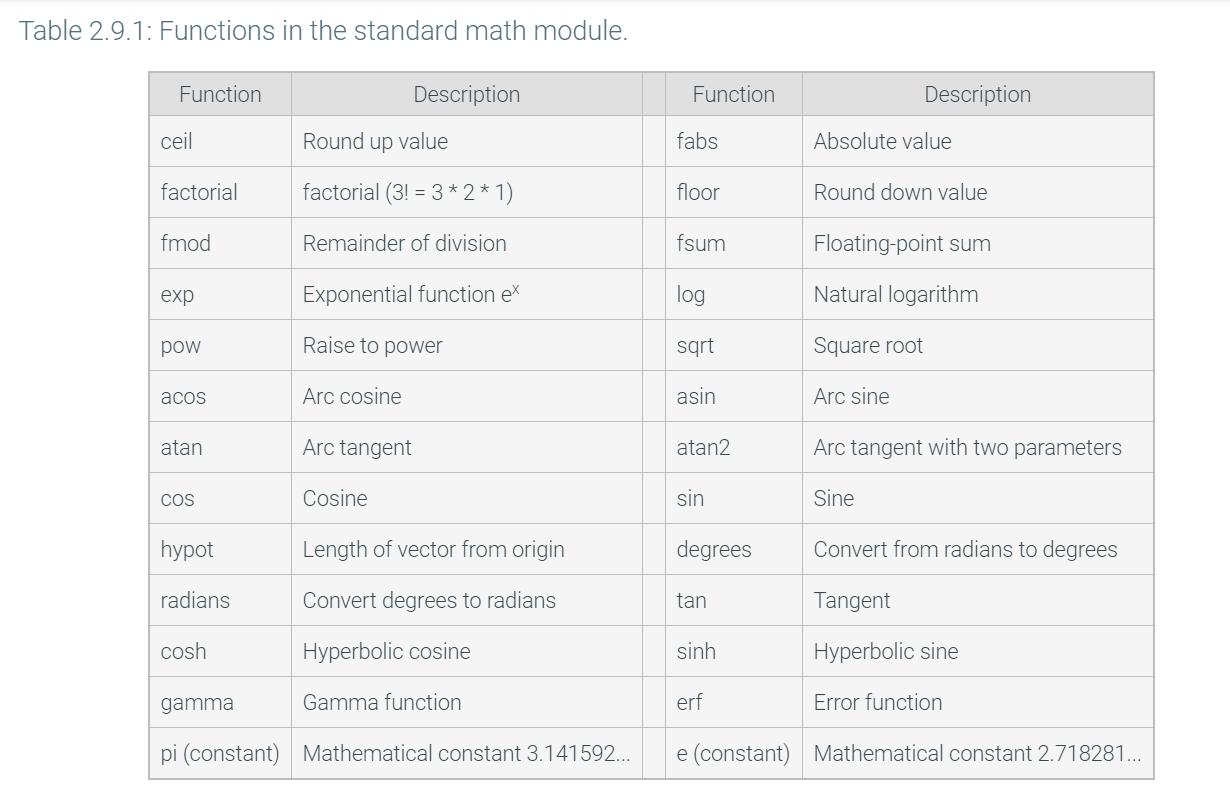


**print**(id(x)) # Print the identity (memory address) of the x object

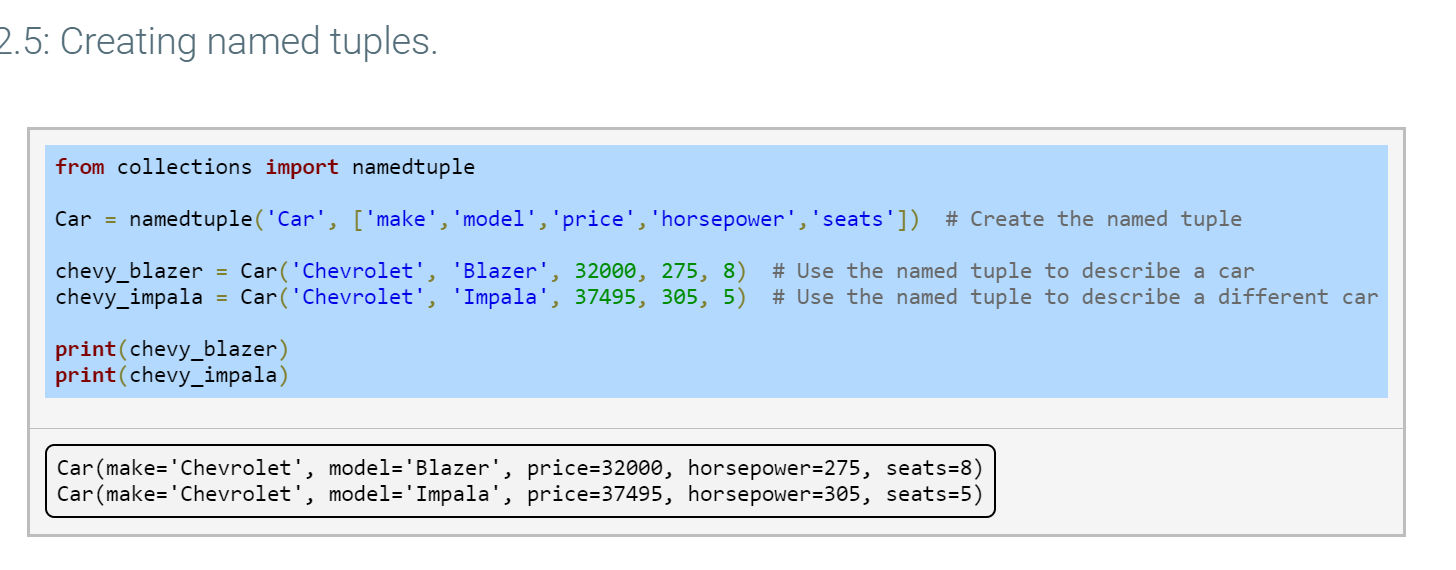
Get prefix

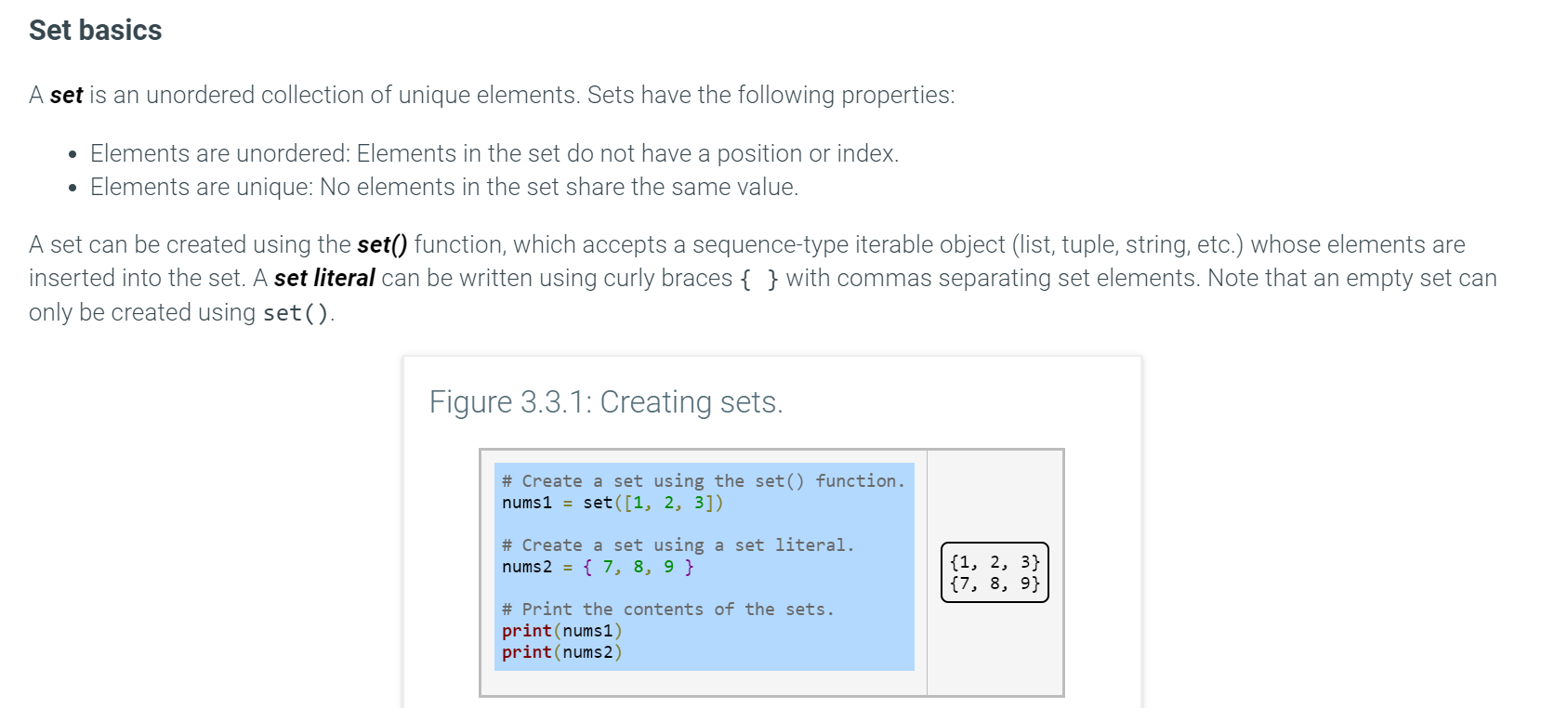
tmp\_val = phone\_num // 10000 # // 10000 shifts right by 4, so 936555. prefix\_num = tmp\_val % 1000 # % 1000 gets the right 3 digits, so 555.

'\_\_main\_\_' # the file was executed as a script



Chapter3



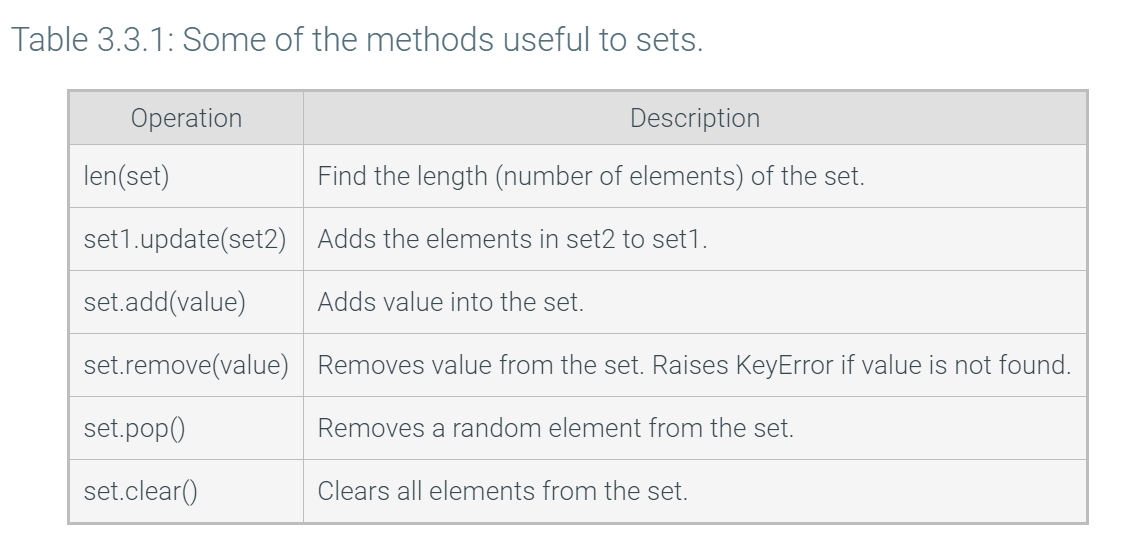


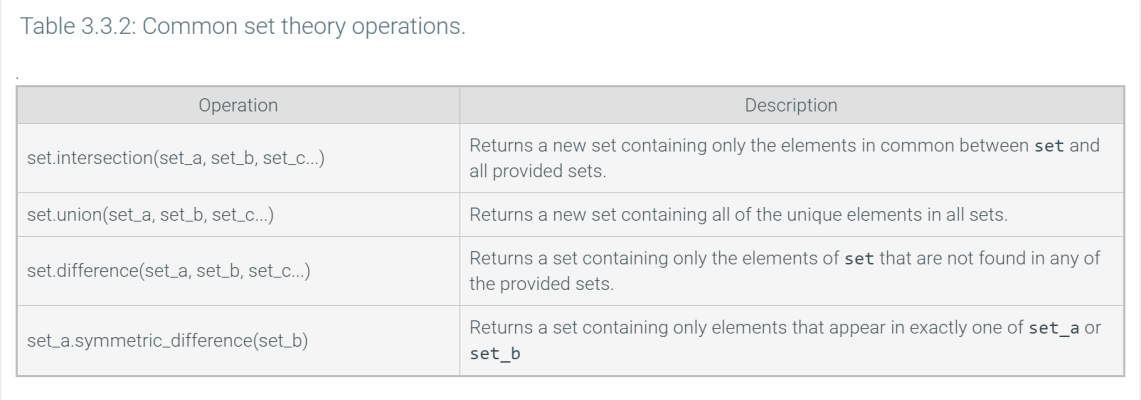
nums1[2] //no index – error

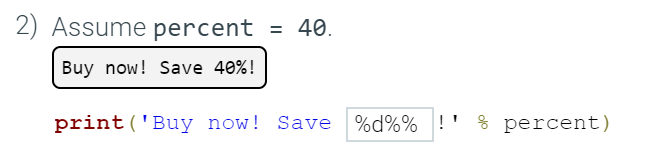
set() // duplicates to be omitted in the created set

first\_names = [ 'Harry', 'Hermione', 'Ron', 'Harry']

names\_set = set(first\_names)









**replace(old, new, count)** -- replaces the first count occurrences of old.

Chapter4

1 < 'abc' result in a TypeError

a < b < c

-Lists and tuples are compared via an ordered comparison of every element in the sequence.

-Dictionaries are compared by sorting the keys and values of each dictionary and then comparing them as lists.

web-based tools: CodePad (<http://www.codepad.org>), Repl ([http://www.repl.it](http://www.repl.it/languages/python3))

Exam1

No variables define needed

Import math >> math.sqrt() | math.pi

str.split()/str.split(‘,’)

Chaining performs comparisons left to right

a < b first - b < c (false) - stop

**Not > and | or**

Arithmetic operators (+, -, \*) > equality operators > relational operators

# print every other element in the list

for num in range(0, len(myList), 2):

print('{:20s}{:^5}{:>10,d}'.format('Annual Salary: ','$',annual\_salary))

integer\_list = [x.strip(' ') for x in integer\_list]

integer\_list = list(map(int, integer\_list))

sorted\_list = sorted(integer\_list)

print(2\*'No' + 3\*'!') >> NoNo!!!

**Part2**

For year\_considered >= user\_year

rather than

for considered\_year != user\_year

avoid infinite loops

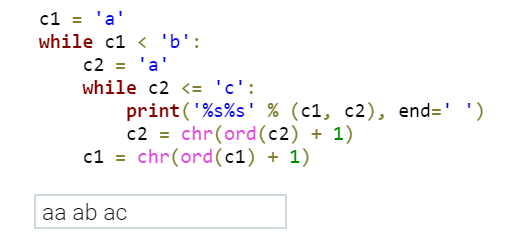
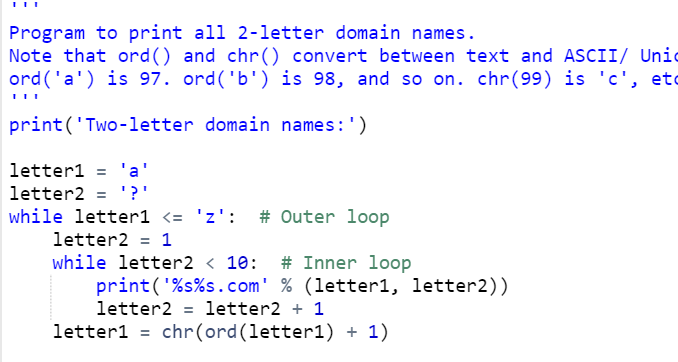
***“””multi-line comment”””***

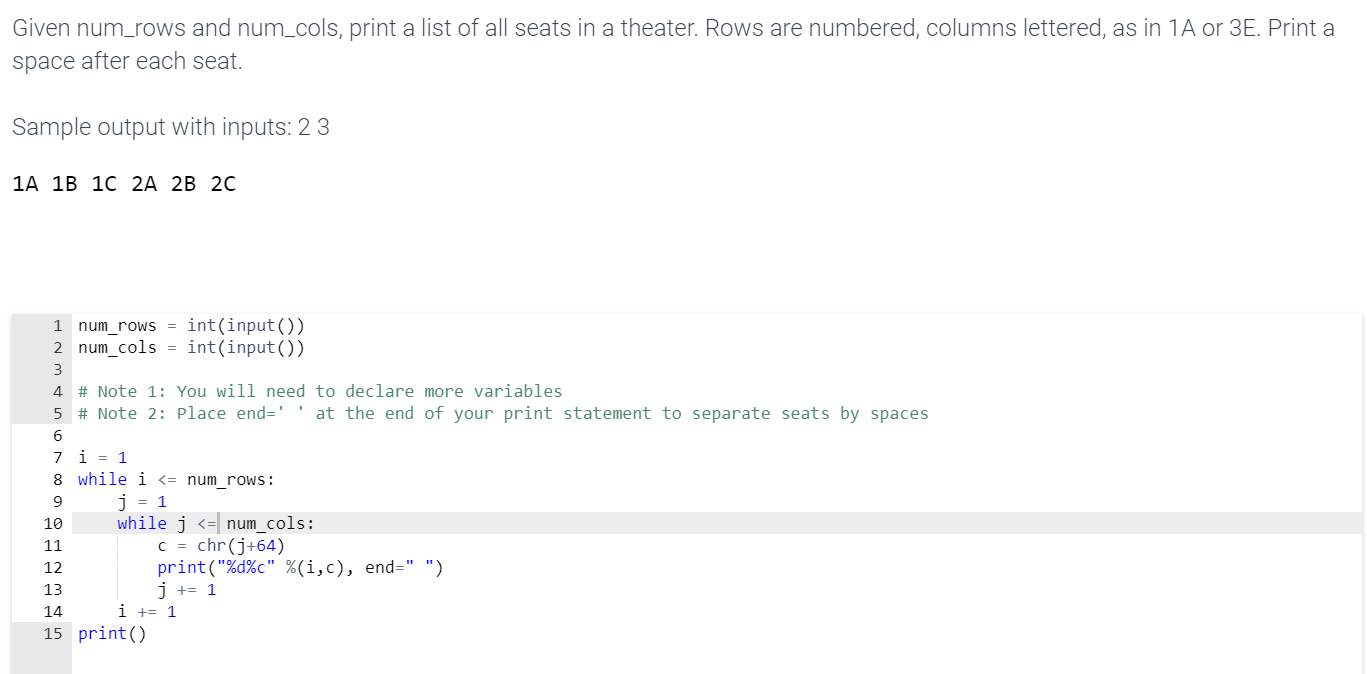
random.randint(0,5) #including

random.seed(5)

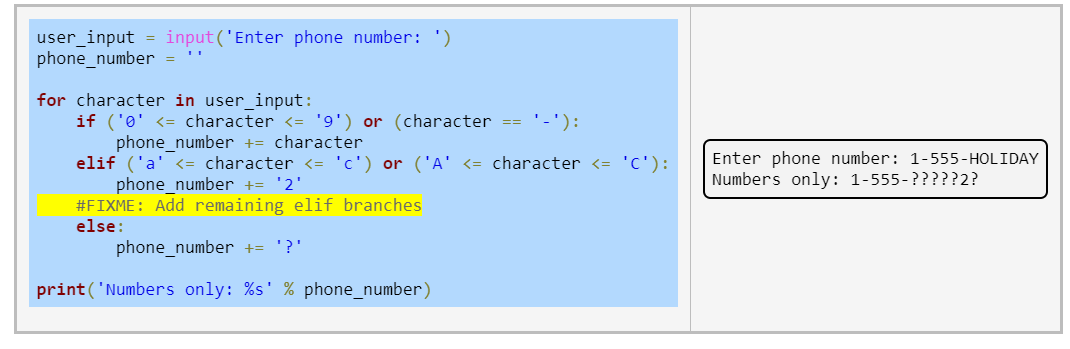
***reversed() #reverse elements order in*** list, dictionary, tuple, or string

ord() converts a 1-character string into an integer, and chr() converts an integer into a character. Thus, chr(ord('a') + 1) results in 'b'

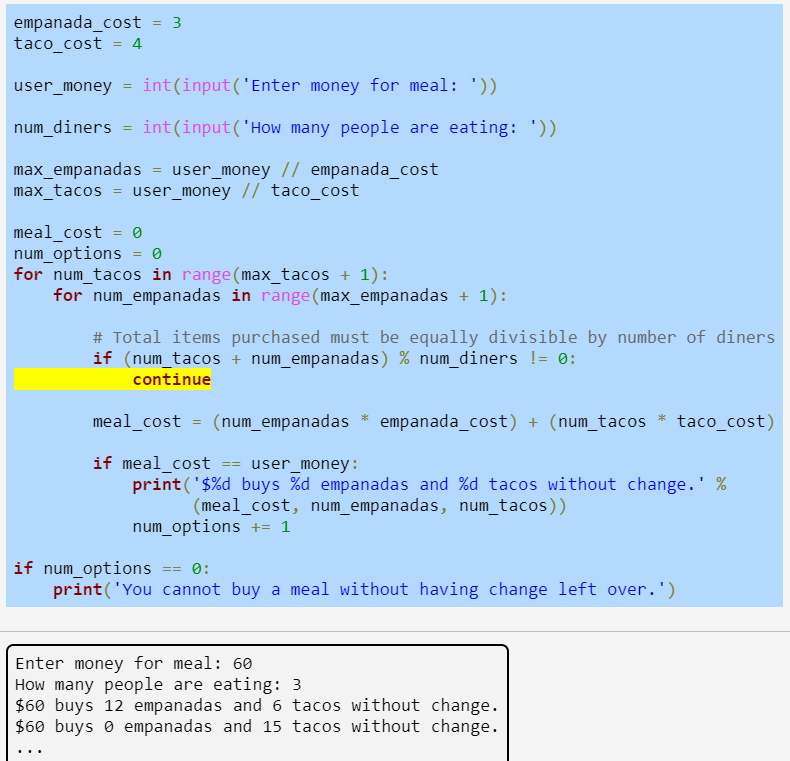
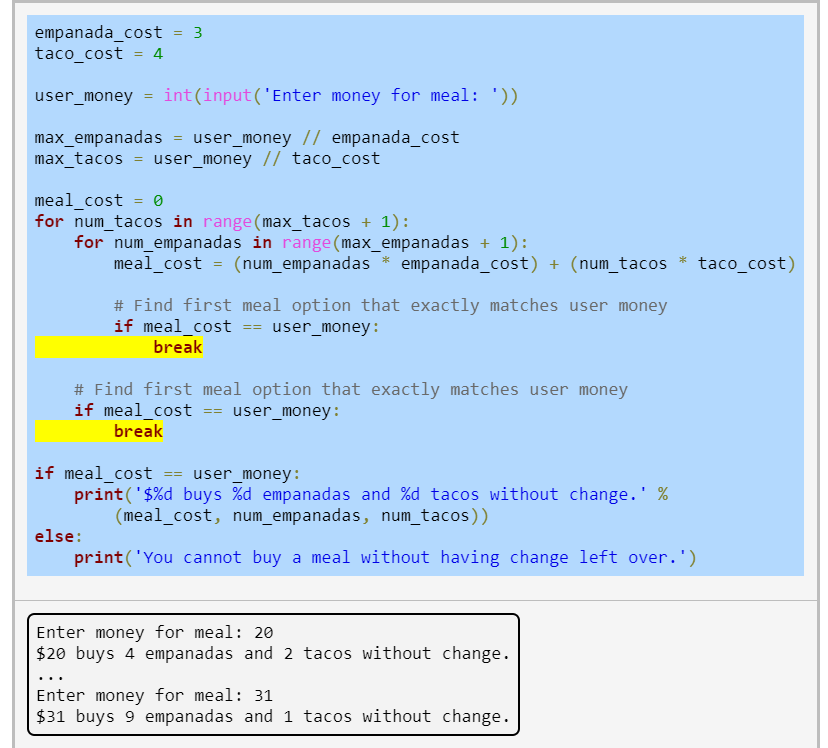




### Incremental programming



**FIXME comments provide a way for a programmer to remember what needs to be added. (FIXME(01/22/2018, John).)**



#break used for loop if condition check

#else executed if the loop didn't hit a break

third-party module called edit\_distance, which calculates string edit distance, or how many characters are different between two strings. For example, the edit distance of "DOG" and "DIG" is 1

import edit\_distance

#find a close match name in the name list

user\_name = input()

for name in legal\_names:

if edit\_distance.distance(name, user\_name) < 2:

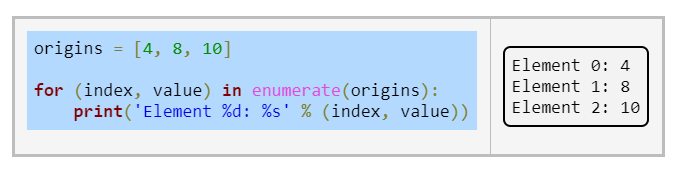
print('You might consider: %s,' % name, end=' ')

break

else:

print('No close matches were found.')

**enumerate()** function retrieves both the index and corresponding element value



**‘’’**

**functions:**

* A ***parameter*** is a function input specified in a **function definition**. Ex: A pizza area function might have diameter as an input.
* An ***argument*** is a value provided to a function's parameter during a **function call**. Ex: A pizza area function might be called as print\_pizza\_area(12.0)

A parameter is like a variable definition. Upon entering the function, the parameter is bound to the argument object provided by the call, creating a shared reference to the object. Upon return, the parameter can no longer be used.

A function with no return statement, or a return statement with no following expression, returns the value **None.** A function evaluates to its returned value. # y = print\_val(9.0) - true)

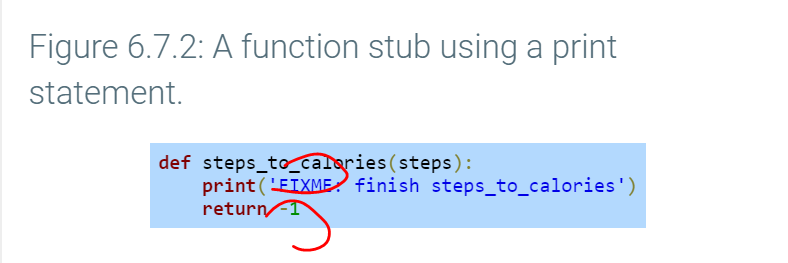
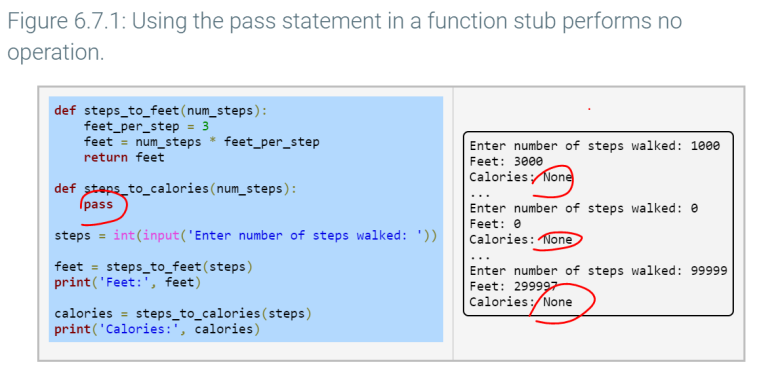
The function's behavior of being able to add together different types is a concept called **polymorphism**. #add(x,y) x,y -int / x,y -string

Python uses **dynamic typing** to determine the type of objects as a program executes(The type of num can change, depending on the value it references.) # add(5, '100') - Error when execute

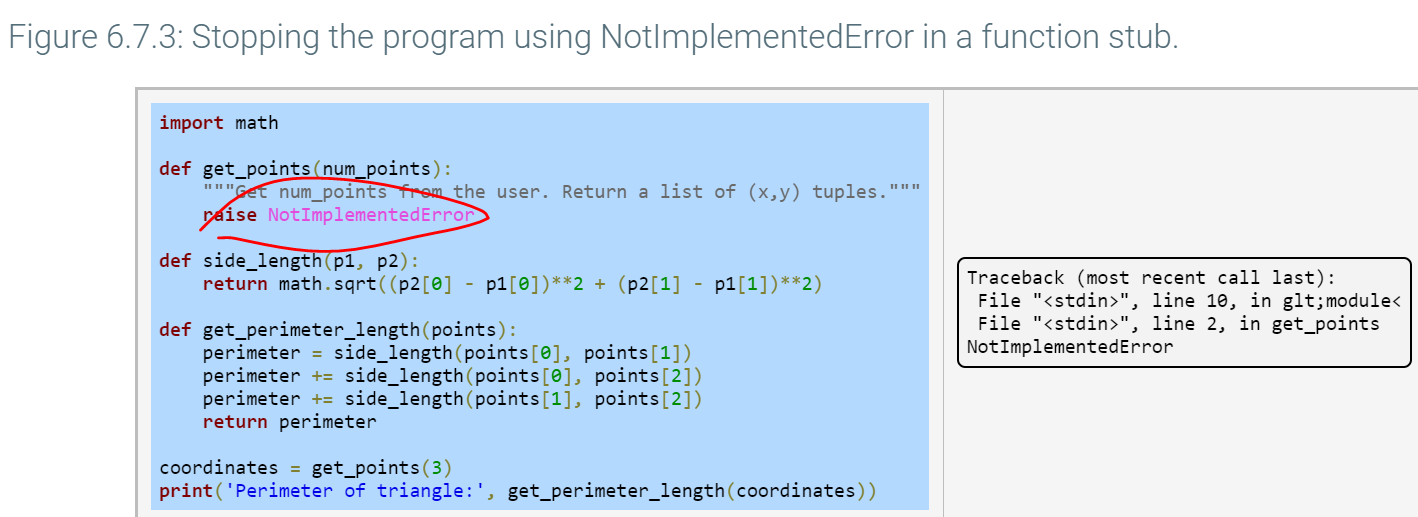
**Modular development** <= 30lines

**Incremental development-function stubs**: function definitions whose statements haven't been written yet

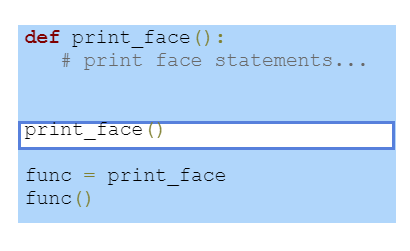
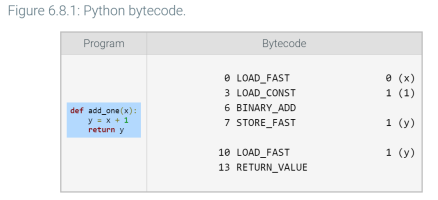
A programmer writing a function stub should consider whether or not calling the unwritten function is a valid operation



# want a program to **stop** executing if an unfinished function is called. Ex: A program that requires user input should not execute if the user-defined function that gets input is not completed:

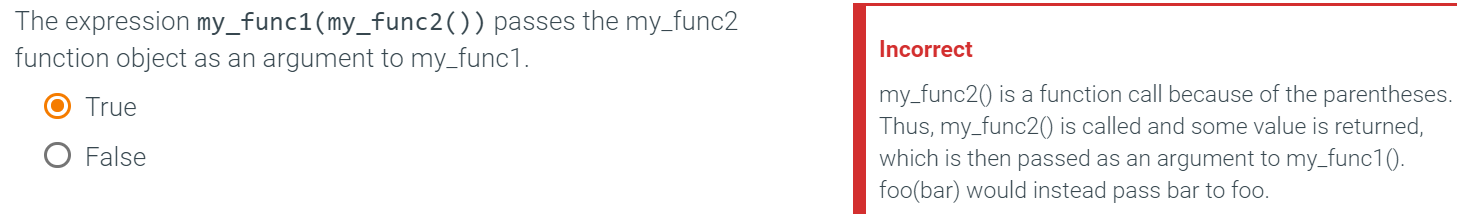


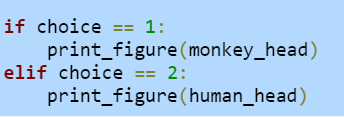
### **Functions as objects #**A function is also an object in Python, having a type, identity, and value. A function definition like def print\_face(): creates a new function object with the name print\_face bound to that object.

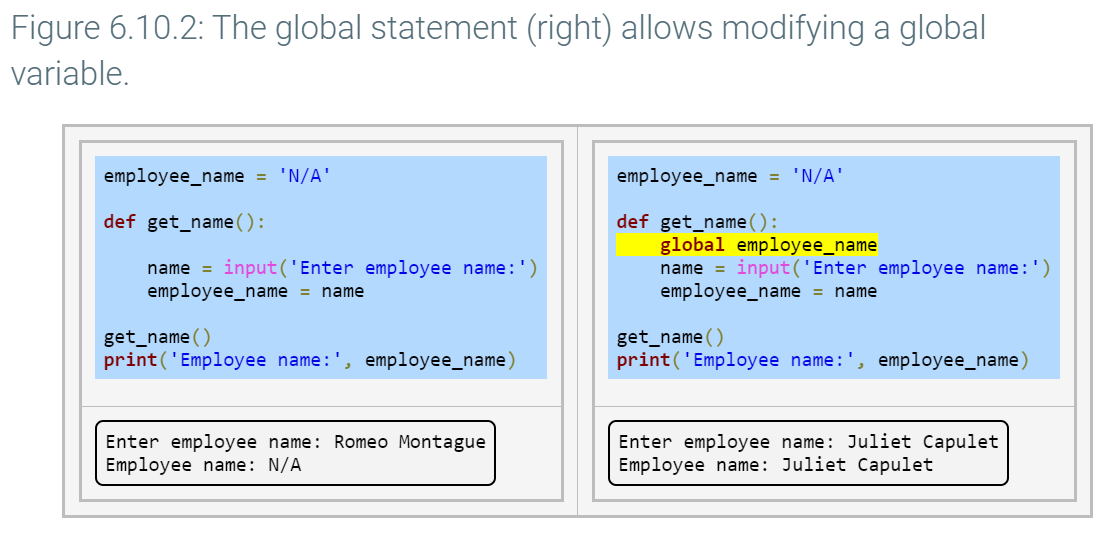


#assign functions as objects - create multiple names

#Functions can be passed as arguments



 instead # only function\_name as parameters



#Modification of mutable global variables, such as list or dict containers, does **not** require a global statement if a programmer is adding or removing elements from the container.

#If a local variable (including a parameter) has the same name as a global variable, then the name refers to the local item - **local overwrite global**

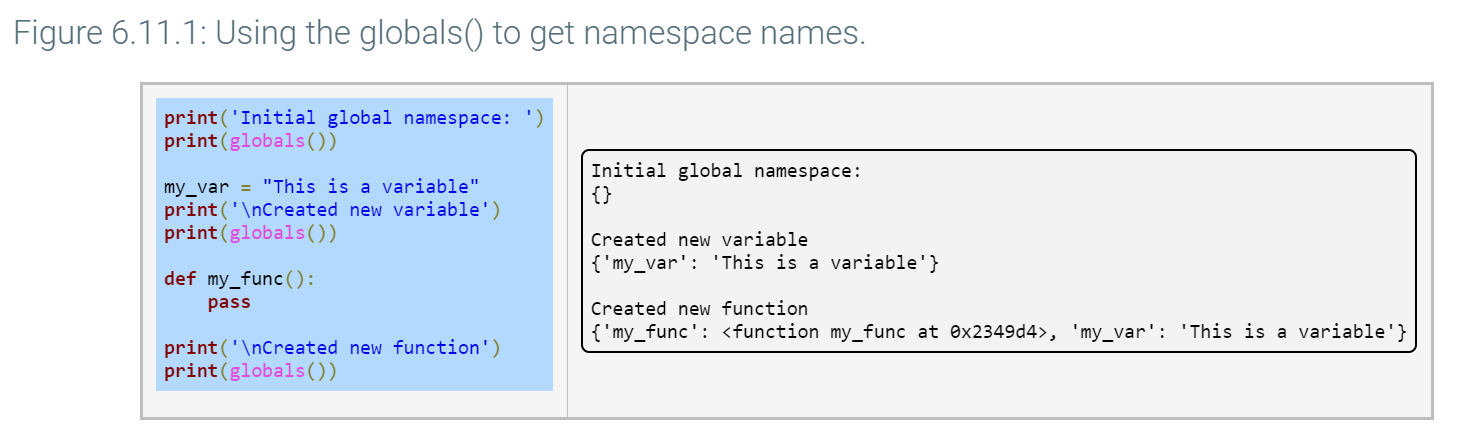
build-in <-global <-local scope

**function scope -** from definition to end of file

**local variable's scope** extends from a function definition's ending colon ":"first assignment to the end of the function.

**namespace**maps names to objects.

**Scope resolution** - searching namespace #non-exist -> NameError



‘’’

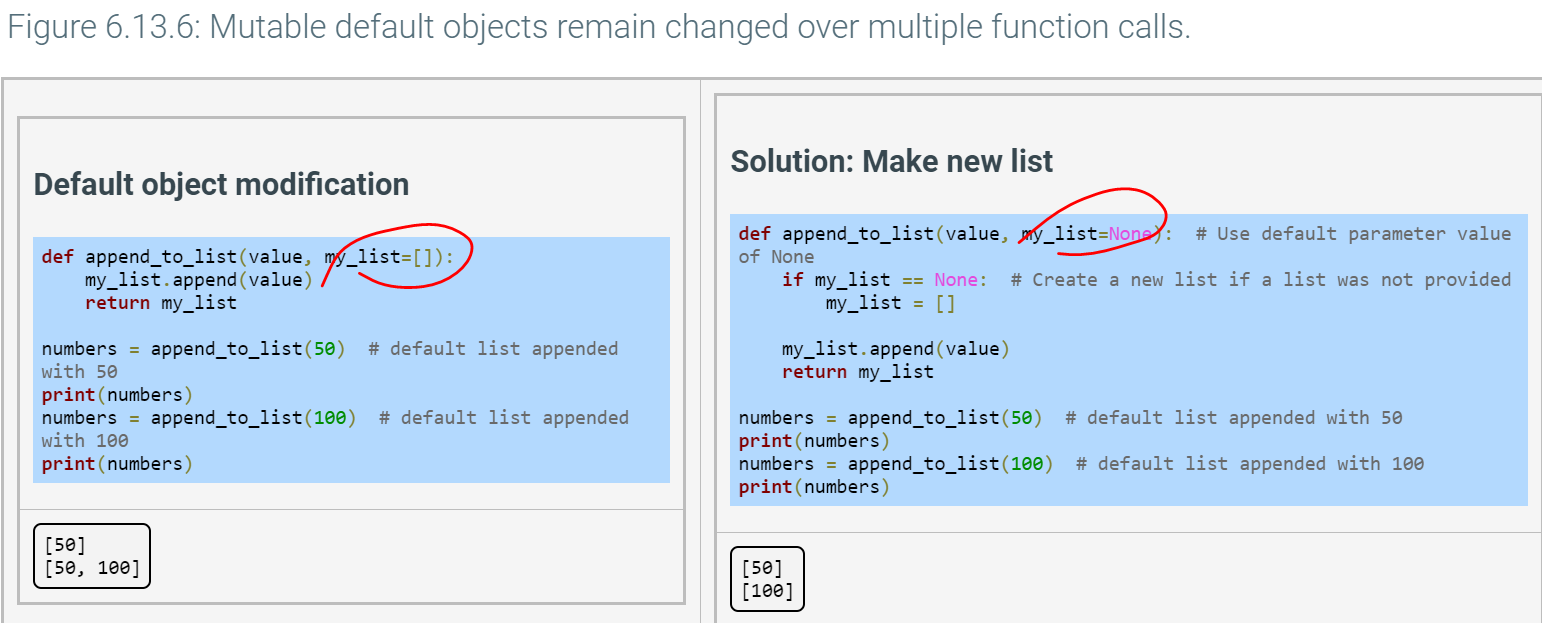
Python as **pass-by-assignment** #local X effects global ->if want to>>use return

>> C changed #eg. avg () - count

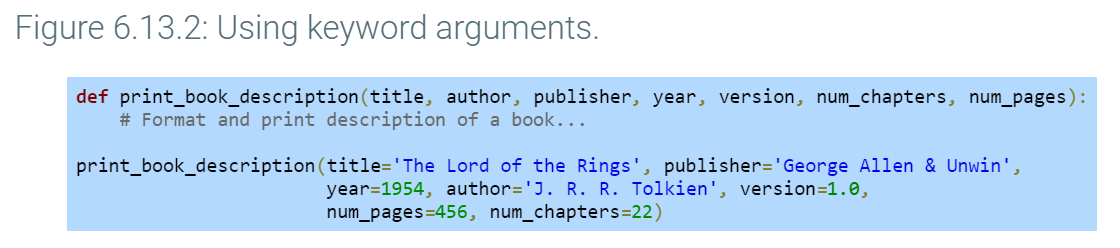
**immutable eg. int/string - not changed**

**mutable eg. list add/sort - changed**

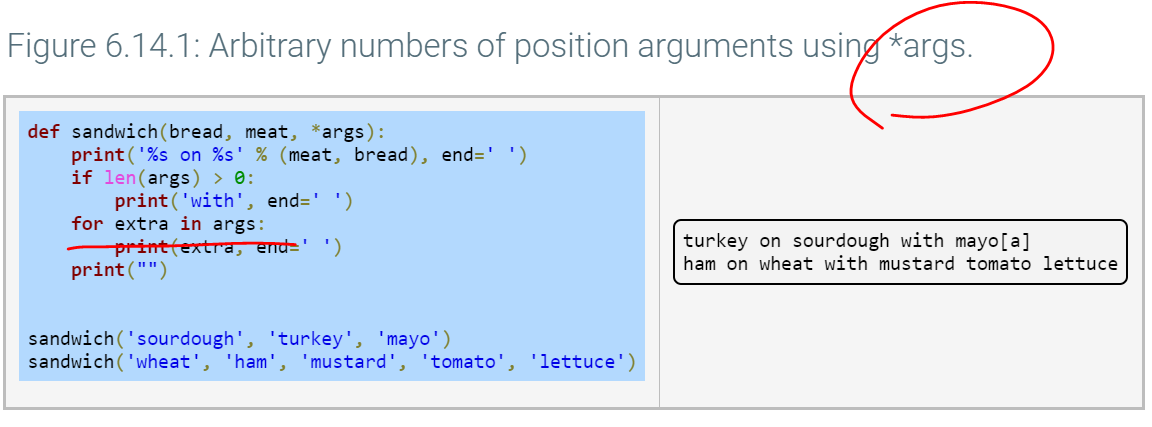
**>>avoid mutable object change - pass copy -> my\_func(my\_list[:])**

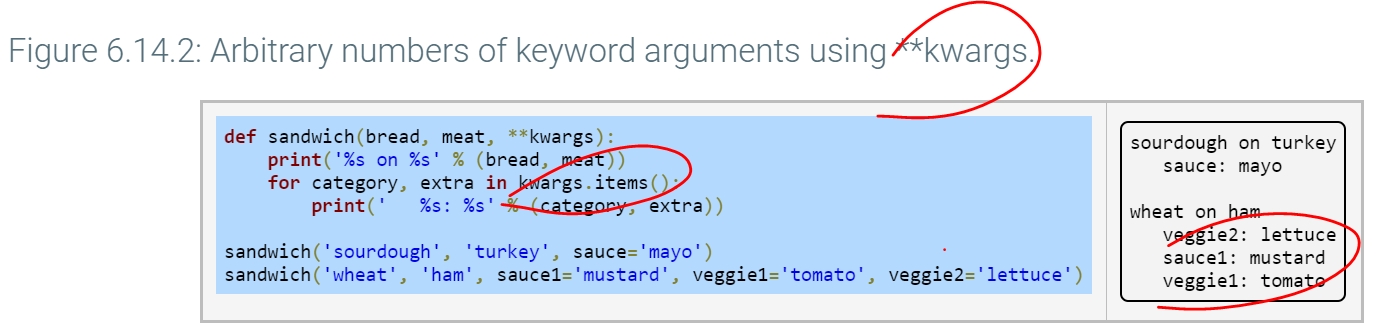


>>the default object persists across function calls ->Every time create a new empty list



All keyword arguments must come after every positional argument, if occupied>>Error

 **#tuple**

 **#dict**

for argument in kwargs:

5 command += ' --%s=%s' % (argument, kwargs[argument]) **#no index for dict**

return only contain one object

>> if return mean, std\_dev ->means return (mean, std\_dev)# tuple

>>can assign -> average, standard\_deviation = get\_grade\_stats(student\_scores #unpacking

**help(\_\_name\_\_)** runs the help function on the global scope of the editor

#round float to 2 decimals -> round(total / people,2)

**String**

my\_str[2:1] is ' ' - empty

'abc?'.islower() returns True

find( x ), find( x, start ), find( x, start, end )



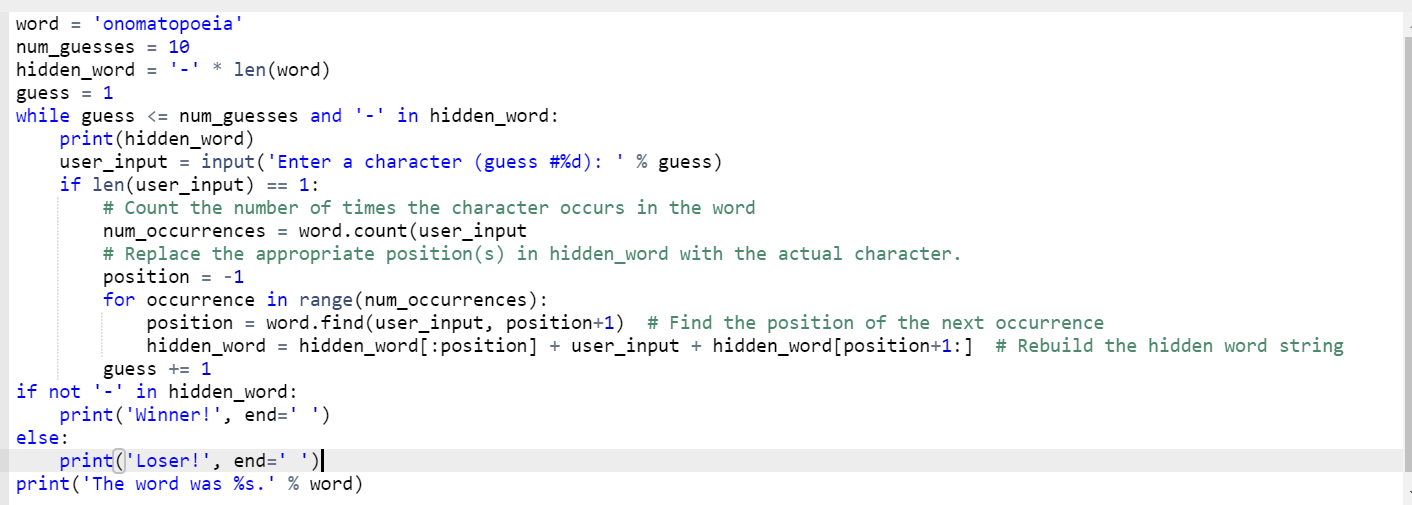
 ''.join(['http://', 'www.', 'ebay', '.com']) produces the string 'http://www.ebay.com'.

'{} + {1} is {2}'.format(2, 2, 4) is not allowed. - not combined



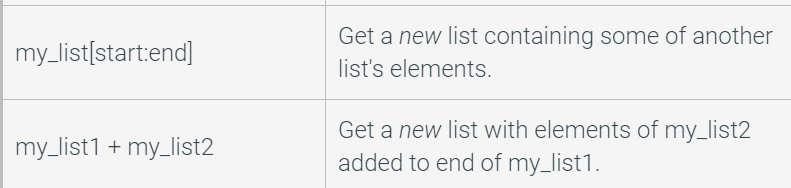
'{0} {{x}}'.format('val') produces the string 'val {x}'.

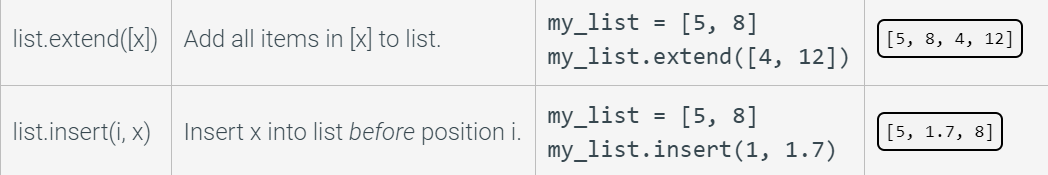
#guess game

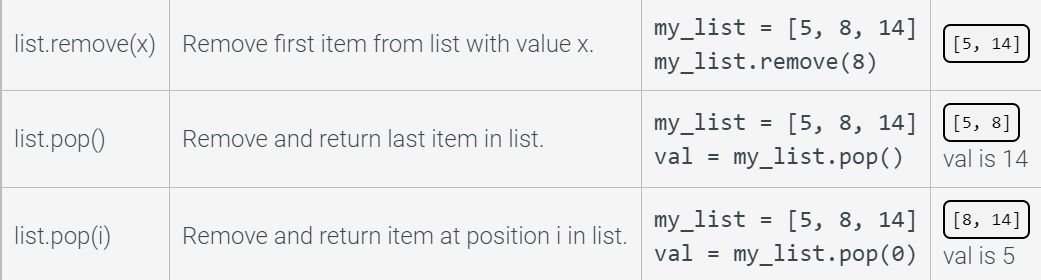


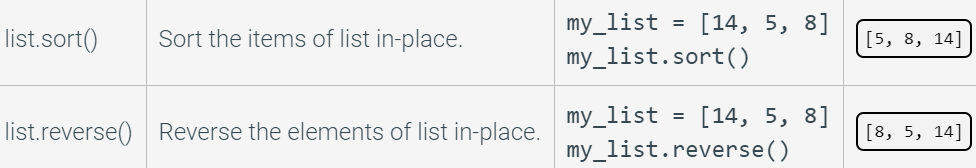
**List**

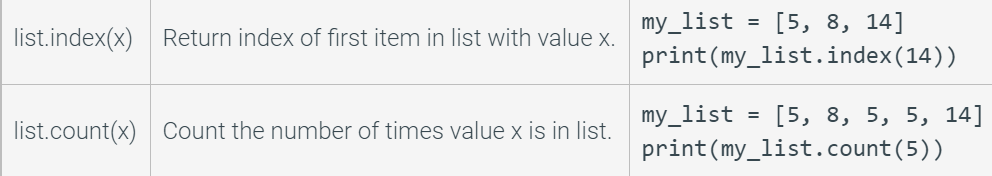
 list('abc') creates a new list with the elements ['a', 'b', 'c'].





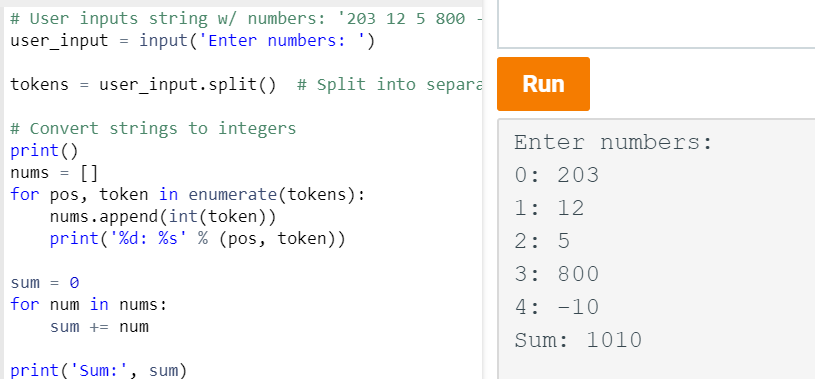


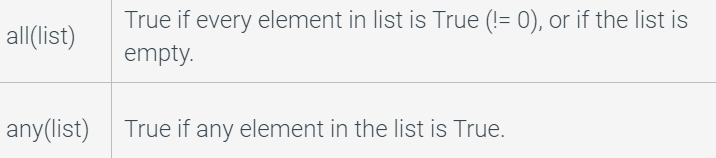


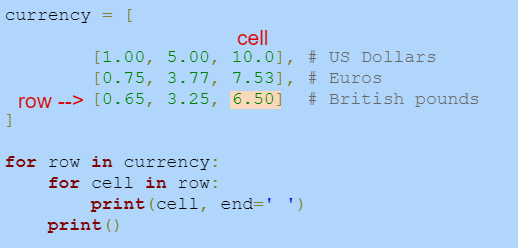


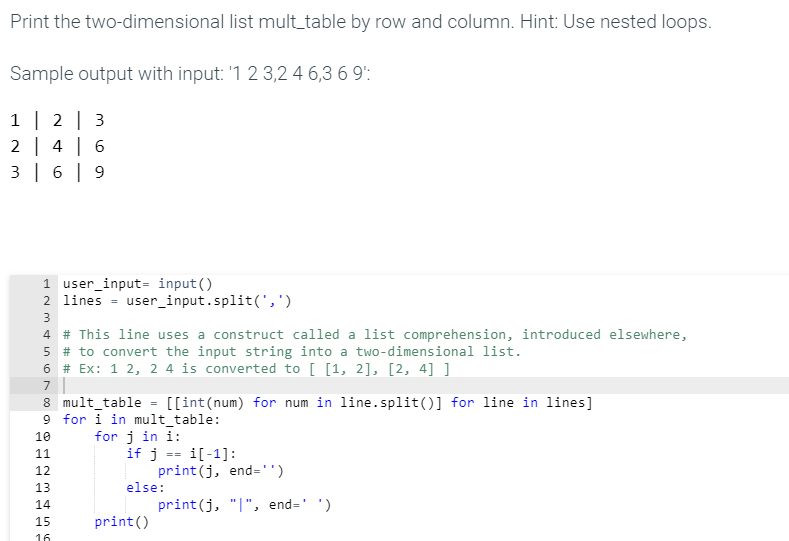
 - find max

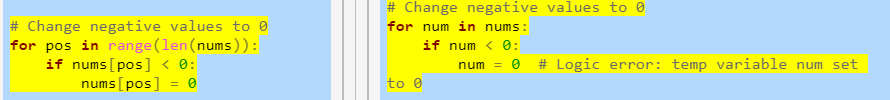
list[:>len(list)] - copy all

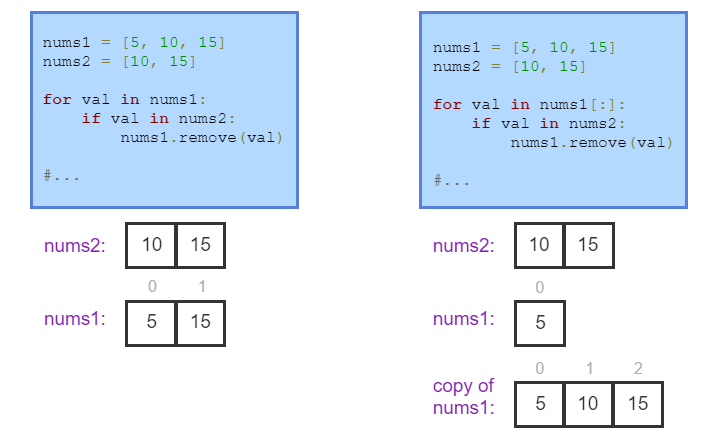


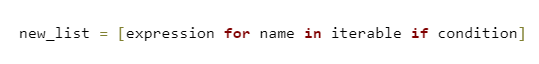


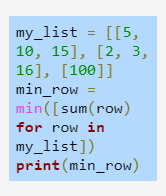
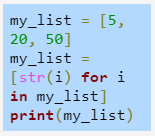




 **- Manipulate with index**



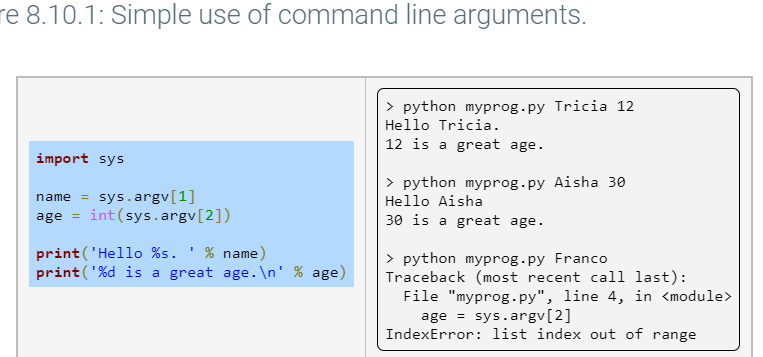


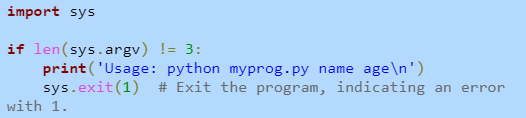




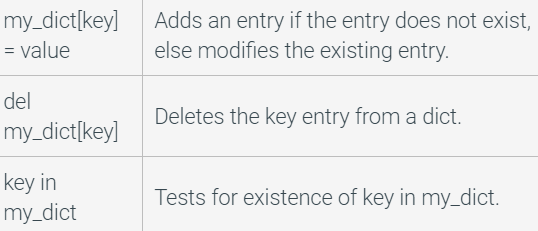




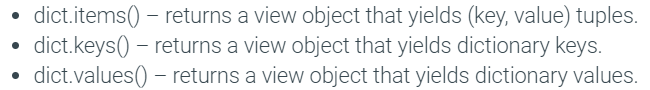


>> 









>> to dict: 