**Lesson 5 - Review**

1. Functions

def name(parameters):

statements

Interaction with the user:

raw\_input()

input()

print()

Function should have a return, the part of code after return is dead code

Keep in mind sometimes we need data type conversion, for example number=int(raw\_input())

2. Selection

Boolean expression:

print(True) print(type(True))

x != y x > y x <= y x == y

Logical expression:

and, or, not

print(x > 0 and x < 10)

if-else condition:

if x % 2 == 0:

print(“even”)

else:

print(“odd”)

chained conditions:

if x < y:

print(“x less than y”)

elif x > y:

print(“x greater than y”)

else:

print(“x equal to y”)

3. Iteration

for loop

for f in alist:

print(f)

for i in range(len(alist)):

print(alist[i])

while loop

count = 0

while count < len(alist):

print(alist[count])

count = count + 1

Some glossaries

**counter**: a variable used for counting something, usually initialized to 0 and incremented in the body

**cursor**: a marker that keeps track of something that might happen (eg. char to be printed)

**definite iteration**: set an upper bound in the iteration body

**loop variable**: a variable used for terminating the loop

**reassignment**: make assignment to a variable in the loop

4. String

string operation: “Go”\*3 = “GoGoGo”

string index: s=“Uforse Education” print(s[0])

string methods:

mystr = “hello world”

mystr.upper() = 'HELLO WORLD'

mystr.lower()

mystr.capitalize()

mystr.strip()

mystr.lstrip()

mystr.rstrip()

mystr.count(“h”)

mystr.replace(“h”, “p”)

mystr.index(“e”)

len(mystr)

mystr[len(mystr)-1]

mystr[0:5]

mystr[:5]

mystr[3:]

Strings are immutable

mystr[0] = “j” # ERROR!

in and not in operators

“p” in “apple”

5. List

[10, 20, 30, 40]

[“hello”, 2.0, 5, [1, 2]]

Concatenation and repetition

fruit=[“apple”, “banana”]

numbers=[1,2,3]

fruit+numbers=[“apple”, “banana”, 1,2,3]

[0]\*4 = [0, 0, 0, 0]

Lists are mutable

fruit=[“apple”, “banana”]

fruit[0] = “pear”

cloning lists

a = [11,12, 13]

b = a[:]

List methods

alist.append(x)

alist.insert(1, x)

alist.pop()

alist.pop(1)

alist.sort()

alist.reverse()

alist.index(x) # return the position of the first occurence of item

alist.count(x)

alist.remove(x) # remove the first occurence of the item

for loop

for item in alist:

print(item)

for idx in range(len(alist)):

print(alist[idx])

In functions, if list is used as argument, then we pass a reference to the list. Any change to the reference will change the list that it references.

A better way to create a list

[<expression> for <item> in <sequence> if <condition>]

mylist = [1,2,3,4,5]

yourlist = [item\*\*2 for item in mylist]

Strings and lists

song = “we are the champions...”

words = song.split()

words = [“hello”, “ccc”]

sentence = “ ”.join(words)

6. Dictionary

inventory = {‘apple’: 10, ‘banana’: 20, ‘pear’: 15}

del inventory[“apple”]

Dictionary methods

adict.keys()

adict.values()

adict.items()

adict.get(k)

7. Recursion

* a recursive algorithm must have a base case
* a recursive algorithm must change its state and move to the base case
* a recursive algorithm must call itself, recursively

def listsum(alist):

if len(alist) == 1:

return alist[0]

else:

return alist[0] + listsum(alist[1:])

def factorial(n):

if n == 1:

return 1

else:

return n \* factorial(n-1)

8. Searching and Sorting

Linear Search

**Linear search is a linear scan of all the elements in the list**

def linearSearch(lst, value):

for v in lst:

if v == value:

return True

return False

Binary Search:

**Remember how binary search change the list we are investigating**

def binarySearch(lst, value):

lst.sort()

left = 0

right = len(lst)-1

found = False

while left <= right and not found:

mid = int((left + right) / 2)

if lst[mid] == value:

found = True

elif lst[mid] > value:

right = mid - 1

else:

left = mid + 1

return found

Bubble Sort:

**bubble sort compares the adjacent neighbors and bubble up**

def bubbleSort(alist):

for pos in range(len(alist)-1, 0, -1):

for i in range(pos):

if alist[i] > alist[i+1]:

temp = alist[i]

alist[i] = alist[i+1]

alist[i+1] = temp

Selection Sort:

**Selection sort put the largest value at the end and iterate**

def selectionSort(alist):

for spot in range(len(alist)-1, 0, -1):

pos = 0

for idx in range(1, spot+1):

if alist[idx] > alist[pos]:

pos = idx

temp = alist[spot]

alist[spot] = alist[pos]

alist[pos] = temp

Insertion Sort:

**Insert the item on the left and the left is all sorted. iterate…**

def insertionSort(alist):

for idx in range(1, len(alist)):

for jdx in range(idx):

if alist[idx] < alist[jdx]:

insertBefore(idx, jdx, alist)

break

def insertBefore(idx, jdx, alist):

temp = alist[idx]

for i in range(idx, jdx, -1):

alist[i] = alist[i-1]

alist[jdx] = temp

Merge Sort:

**Split the list into half until single item, and merge together**

def mergeSort(alist):

if len(alist) > 1:

mid = int(len(alist)/2)

leftList = alist[:mid]

rightList = alist[mid:]

mergeSort(leftList)

mergeSort(rightList)

## here we need to merge 2 sublist together

idx = 0

jdx = 0

kdx = 0

while idx < len(leftList) and jdx < len(rightList):

if leftList[idx] < rightList[jdx]:

alist[kdx] = leftList[idx]

idx += 1

else:

alist[kdx] = rightList[jdx]

jdx += 1

kdx = kdx + 1

if idx == len(leftList):

while jdx < len(rightList):

alist[kdx] = rightList[jdx]

kdx = kdx + 1

jdx = jdx + 1

if jdx == len(rightList):

while idx < len(leftList):

alist[kdx] = leftList[idx]

idx = idx + 1

kdx = kdx + 1