Computers can process text too

Strings

A sequence is a collection or composition of elements in a specific order

A Python string is:

* A sequence of characters
* The sequence be explicitly written between two quotation marks
  + Python allows single or double quotes

Examples:

* ‘hello world’
* “hello world”
* ‘’
* ‘a’
* “abc”

Like numbers, string can be assigned to variables

Str = ‘Hello World’

You can access parts of strings via indexing

Str[n] means give me the nth – 1 character of the string str

Str[0] == ‘H’

Str[1] == ‘e’

Str[2] == ‘l’

Str[5] == ‘’

Another ways to access parts of a string is via slicing

* Str[m:n] means to give me the part of the string from character m up to but not including n
* Both m and n are optional
  + If m is omitted then it starts from the beginning of the string
  + If n is omitted it ends at the end of the string
* Examples:
  + Str == ’hello world’
  + Str[1:4] == ’ell’
  + Str[:5] == ”hello”
  + Str[1:] == ’ello world’

You can concatenate string or put two string together

The plus sign(+)

Examples

* ‘hello’ + ‘world’ == ‘helloworld’
* ‘a’ + ‘ab’ == ‘aab’
* ‘ab’ + ‘c’ == ‘abc’
* ‘a’ + ‘b’ + ‘c’ == ‘abc’

Python strings are immutable

* Means parts of string scannot be changed using the assignment operatior
* If we assign a new value to a string is replaces the old value
  + Basically a new string

Python String Operations

Def strIsEmpty(str):

If str==’’

Return true

Else:

Return false

Def strHead(str):

Return str[:1]

Def strtail(str):

Return str[1:]

Def strConcart(str1,str2):

Return str1 + str2

LengthRec(string)

Reverse string

Def reverseRec(str):

If str== ‘’:

Return ‘’

Else:

Return reverseRec(strTail(str) + strHead(str)

Reverse Trace

‘abc’

reverseRec(‘abc’)

=revRec(‘bc’)+’a’

=revRec(‘c’)+’b’+’a’

=revRec(‘’)+’c’+’b’+’a’

=’’+’c’+’b’+’a’

=’cba’

String Iteration

We often can replace recursion with iteration

* For loop
  + For ch in ‘abc’
    - Print(ch)
  + This will print:
    - A
    - B
    - C
* With a for loop we can avoid recursion

Def lengthIter(str\_:

Ac=0

For ch in str:

Ac = 1 + ac

Return ac

Len(str)

Sometimes we want to access parts of a string by index

For I in range(0,len(str))

For I in range(2,5)…this will do all I’s 2,3,4

St == ‘abcd’

For i in range(0, len(st))\_

If not i == 2:

Print(st[i])

Def fact(n):

Val = 1

For i in range(1, n+1)”

Val = val \* i

Return val

Files

Def hide(textFileString, hiddenWord):

forCurrentWord in textFileString:

ifCurrent

for line in open(‘text1.txt’):

print(line)

for line in open(‘text1.txt’):

print(line==’word1’)

Str.strip()

Str.rstrip()

Str.lstrip()

Removes all whitespace from a string

It returns the ‘stripped’ string

Str=str.strip()

Linear function

Worst-case analysis

For each element we spend some constant time (n)

Plus some fixed time to startup and end the search (c)

If processing of an element takes constant time k

(k\*N)+c

The mathematical way to ignore this is to say the time complexity of linear search is O(N)

* It pronounced Big-O of N

Makes it easy to compare algorithms

We prefer O(1) to O(N) and O(N2)