Seqs, Loops, Funcs, Recursion - Tutorial 3

Sequences in Python

Sequence : Positionally ordered collection of items

Sequence data types : Lists, Tuples, Strings

List Sequence type

Examples of lists :

1. [1, 2, 3]
2. [True, 0, 1, “Hello”]
3. [[1, 2, 3], [1, 2, 3]]

Creating a list : [ ] or list()

```
l = [1, 2, 3, 4, 5] #list with 5 items
```

Accessing through indexing:

```
l = [1, 2, 3, 4, 5]

  0  1  2  3  4      #Normal/Forward indexing

 -5 -4 -3 -2 -1      #Reverese/Backward
```

Accessing elements & slicing :

Accessing individual elements

l[0] → 1

l[4] → 5

l[-1 → 5

l[-5] → 1

Slicing a list

Syntax : listvar[start : stop : step]

l[0:5:1] → [1, 2, 3, 4, 5]

l[: : 2] → [1, 3, 5]

Finding the length of list :

Keyword : len # returns the length of the sequence

len(l) → 5

len(city[: : 2]) → 3

String Sequence type

Examples :

city = “Kharagpur” #type of str

Or anything in *single/double/triple* quotes

Accessing string chars :

city[0] → “K”

city[-1] → “r”

city[: : 2] → “Kaapr”

city[: : -1] → “rupgarahK”

Finding length of the string:

Keyword : len # returns the length of the sequence

len(city) → 9

len(city[: : 2]) → 5

For syntax & examples

for {*variable}* in {*sequence}*:

Do something with variable &/ sequence

Examples:

1. Iterating with a list range sequence

```
for i in range(5):
    print(i)
```

2. Iterating with a explicit list

```
nums = [0, 1, 2, 3, 4]
for i in nums:
    print(i)
```

3. Iterating with a string sequence

```
city = “Kharagpur”
for char in city:
    print(char)
```

Palindrome

Functions

Function : A block of code that you want to use repeatedly

Syntax :

```
def func_name(inputs):
    return
```

Convert any code to a function :

1. Identify the primary inputs
2. Result of the code block

Convert palindrome code block to a function

```
def is_palindrome(my_str : str ):
    #palindrome code block
    return True / False
```

Recursion

Types:

1. Loop based recursion (easy)
2. Function based recursion (elegant)

IMPORTANT

Stop/Base Condition

Examples:

```
n = 3

sum = 0

for i in range(n+1):

    sum += i

print(sum)
```

```
def get_sum(n : int):

    if n == 1: # base condition

        return 1

    return n + get_sum(n-1) # recursive call
```

```
def is_palindrome_rec(my_str : str):

    if len(my_str) > 0:

        if my_str[0] == my_str[-1]:



    return True
```