Module 1:

Strings, Characters and Operators

By Ninad Gaikwad

Reference - “Core Python Programming”
Dr. R. Nageshwara Rao
Dreamtech Press

Strings

8.1 Creating Strings

- s1 = ‘Welcome to core python’
- s1 = “Welcome to core python”
- s1 = ‘’’Welcome to
  core python’’’
- s1 = “””Welcome to
  core python”””
- s1 = ‘Welcome to “core” python’

8.2 Other operations in python

1. Length of string:
   - str=“Welcome to ‘core’ python”
   - Length of a string: len(str)
2. Indexing strings: str[i]
3. Slicing the strings: stringname[start: stop: stepsize]
   - Eg str = ‘core python’
   - str[0:9:1] # will give “core pyth” as output
   - str[-4:-1] # will give “tho” as output

Strings

4. Repeating strings:
    - print(str*2) # will give “core pythoncore python”
    - print(str[5:7]*3) # will give “pypypy” as output
5. Concatenation of strings:
    - We can use + operator
    - Eg if s1, s2 =”core”, ”python”
      s1+s2=corepython
    - join() method can also be used to obtain the same result
6. Checking membership:
    - Eg if substring in mainstring:
      print(“Substring present”)
7. Comparing strings:
    - Eg s1, s2 =”Box”, ”Boy”

```
if(s1==s2):
        print(“both are same”)
elif(s1<s2):
        print(“s1 is less than s2”)
        # will print this message as x is less than y
```

Strings

8. Removing spaces from a string
    - lstrip(), rstrip(), strip()
9. Finding substring:
    - mainstring.find(substring, beginning, ending) # returns the index of first occurrence position where found else returns -1
    - mainstring.index(substring, beginning, ending) # same as find() except .index() method throws a ValueError error
    - rfind() and rindex() finds substring from right to left
10. Counting substring in a string
    - Eg n = stringname.count(substring)
11. Strings are immutable
    - Performance: It takes less time to access them
    - Security: Any attempt at modifying the object will create a new object
12. Replacing a string with another string
    - str1 = str.replace(old, new)
13. Splitting and joining:
    - list1 = str.split(“,”) # Split over comma and save in list
    - str1 = “-”.join(list) # join list or tuple by - and save in string str1

Strings

14. Changing case of a string
    - str.upper() # Converts all to uppercase
    - str.lower() # Converts all to lowercase letters
    - str.swapcase() # Converts upper to lower and vice versa
    - str.title() # First letter of each word is capitalized
15. Checking starting and ending of string
    - str.startswith(substring) # Returns true if string starts with substring
    - str.endswith(substring) # returns true if string ends in substring
16. String testing methods:
    - isalnum(), isalpha(), isdigit(), islower(), isupper(), istitle(), isspace()
17. Formatting the strings:
    - str = '{}, {}, {}'.format(id, name, sal)
    - str = '{2}, {0}, {1}'.format(id, name, sal)
    - str = '{one}, {two}, {three}'.format(one = id, two = name, three = sal)
    - str = 'Id= {:d}, name= {:s}, salary= {:10.2f}'.format(id, name, sal)
    - print('{:*>15d}'.format(num)) # Right align total digit places are 15
    - print('{:*^15d}'.format(num)) # Center align total digit places are 15
    - print('{:*<15d}'.format(num)) # Left align total digit places are 15
    - str = 'Hexadecimal = {:.>15x}\n Binary= {:.<15b}'.format(num1, num2)

Characters

8.3 Working with characters

- Characters are individual elements of a string
- ch.isalpha() - used to check whether a character is an alphabet
- str.sort() - sorts a list of strings in alphabetical order
- s==str[i] - comparing the strings
- Finding number of characters in a string - count individual characters
- Inserting substring into a string - Make an intermediate list of characters containing substring and make a string out of it by using .join() function

Type of Operators in Python

1. Arithmetic Operators (7)
2. Assignment Operators (8)
3. Unary minus Operator (1)
4. Relational Operators (6)
5. Logical Operators (3)
6. Boolean Operators (3)
7. Bitwise Operators (6)
8. Membership Operators (2)
9. Identity Operators (2)

Arithmetic Operators (7)

Operator	Meaning	Eg (a=13, b=5)
+	Addition	a+b =18
-	Subtraction	a-b=8
*	Multiplication	a*b=65
/	Division	a/b=2.6
%	Modulus. Gives remainder	a%b=3
**	Exponent. Exponential power value	a**b=371293
//	Integer division. Gives Integer quotient	a//b=2

Arithmetic Operators

- Arithmetic Operator evaluation precedence: Left to right for same level operators
  1. Parenthesis =>> ( )
  2. Exponentiation =>> **
  3. Multiplication, division, modulus & floor division =>> * / % //
  4. Addition and subtraction =>> + -
  5. Assignment Operations =>> =
- Python interpreter can be used as a calculator

Assignment (8), Unary minus (1), Relational Operators (6)

- Assignment Operators (8):
  - Store right side value in left side variable
  - =, +=, -=, *=, /=, %=, **=, //=
- Unary minus Operator (1)
  - Value is negated when used before a variable
  - Eg: -x
- Relational Operators (6)
  - Compare two quantities
  - Result will be true or false
  - >, >=, <, <=, ==, !=

Logical Operators (3)

- Used to **construct compound conditions** which are made of more than one simple conditions
- False indicates 0, True indicates any other number
- Eg if(x<y and y<z)

Operator	Example	Meaning
and	x and y	And operator. If x is false return x else return y.
or	x or y	Or operator. If x is false then return y else y.
not	not x	Not operator. If x is false then return true else false.

Boolean Operators (3)

- Operate on bool type literals ie. true and false
- Result provided by the boolean operator will be boolean

Operator	Example	Meaning
and	x and y	Boolean and operator. If both x and y are true then return true else return false.
or	x or y	Boolean or operator. If either x or y is true then return true else return false.
not	not x	Boolean not operator. If x is false then return true else false.

Bitwise Operators (6)

- Operators act on individual bits (0 or 1)
- Can be used both on binary as well as decimal numbers
- Result is always decimal when the operands are decimal
- Types: Complement (~), AND (&), OR (|), XOR(^) , Left shift (<<), Right shift(>>)

1. Bitwise complement (~)

- Symbol tilde (~)
- Changing 0’s to 1’s and vice versa
- Operation is performed by NOT gate circuit in electronics

Bitwise Operators

2. Bitwise AND (&)

- Represented by ampersand (&)
- By multiplying positional input bits of the two numbers we can get the output bit
- AND gate circuit in computer performs the operation

3. Bitwise OR (|)

- Pipe (|) symbol represents bitwise OR
- Adding the input bits we can get the output bits
- Performed by OR gate circuit in computer

Bitwise Operators

4. Bitwise XOR(^)

- Performs exclusive or operation on the bits of numbers
- (^) symbol is called cap, carat or circumflex
- When we have odd number of 1’s in the input bits, we get output bit as 1
- XOR gate circuit of computer will perform this operation

5. Bitwise Left shift (<<)

- x << n, Shifts the bits of x left by n positions
- Operator symbol (<<) is read as double less than sign
- Leftmost bits are lost, ie. replaced by zero

Bitwise Operators

6. Bitwise Right shift(>>)

- x >> n, shifts the bits of x right by n positions
- Symbol is read as double greater than sign (>>)
- The operation preserves the rightmost bit, ie. the sign bit is written to the shifted places

Membership Operators (2)

- Useful to test membership in a sequence: String, List, Tuple or Dictionary
- Membership operators: in, not in

1. in
   - Returns true if the element is found in the specified sequence, else returns false.
   - Can be used in for loop
   - Eg. postal = {‘Delhi’:110001, ‘Chennai’: 600001, ‘Mumbai’: 400001}

     for city in postal:

     print(city, postal[city]) // will display all the cities with their postal code

2. not in
   - Returns true if the element is not found in the specified sequence, else returns false

Identity Operators (2)

- Compare the memory locations of two objects
- Memory location of an object can be seen using id() function
- Identity Operators: is, is not

1. is
    - Used to compare if two objects are same or not
    - Will internally compare the identity number of the objects
    - Will return true if the identity numbers are the same else will return false
2. is not
    - Will return true if the identity number of the two objects is not the same else will return false

Operator precedence and associativity

Operator	**Name**
( )	Parenthesis
**	Exponentiation
-, ~	Unary minus, bitwise complement
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise left shift, bitwise right shift
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR

Operator precedence and associativity

Operator	**Name**
>, >=, <, <=, ==, !=	Relational (comparison) Operators
=, %=, /=, //=, -=, +=, *=, **=	Assignment Operators
is, is not	Identity operators
in, not in	Membership operators
not	Logical not
or	Logical or
and	Logical and

Mathematical Functions

- Import “math” module
- For complex numbers import “cmath” module

Experiment 2

Q: Write a menu-driven python program to accept a number and string passed, check if the string is a palindrome and find the factorial of the input number

A: Step 1: Design menu using while loop

Step 2: Accept a number and a string

Step 3: Check if the first half of the string is equal to the reverse of the remainder

Step 4: Find the factorial of the number using recursive and conditional statements