22/8/23

Assignment - 1

* ASCII VS UNICODE

(1) ASCII :-

→ ASCII is used for the representation of text such as symbols, letters, digits, etc in computers.

→ ASCII stands for American standard Code for Information Interchange.

→ It is a character encoding standard for electronic communication.

→ It is used to representing 128 English characters in the form of numbers with each letter being assigned to a specific number in the range 0 to 127.

e.g.: ASCII Code for A → 65 & B → 65

(2) UNICODE :-

→ Unicode provides a unique way to define every character in every spoken language of the world by assigning it a unique number.

→ It defines 1,40,000 characters from more than 150 modern & historic scripts along with emoji.

→ It can be defined with different characters encoding like UTF-8, UTF-16, UTF-32 etc.

→ UTF-8 is most popular as it is used in over 90% of websites on the world wide web (WWW) as well as on most modern Operating Systems (OS) like windows.

POORNIMA

* Bitwise XOR (A^B) -

→ It results 1, if any of the operand bit is 1 but not both, otherwise it results bit 0.

Ex- a = 10 = 1010 (Binary)
b = 4 = 0100 (Binary)

```
a ^ b = 1010
        0100
        ----
        1110  => 14
```

* is Operator

→ The 'is' is known as the identity operator.

→ when the variable on either side of an operator point to the exact same object, the is operator's evaluation is true. Otherwise, it will evaluate as False.

Ex. x = 10
y = x
y is x
O/p True

* == operator

→ The '==' is known as the equality operator.

→ When the variable on either side have the exact same values, the == operator evaluation is true. Otherwise it will evaluate as false.

Ex. x = 10
y = x
x == y
O/p True

* Indexing :-

→ It begins from 0. The first element in the sequence is represented by 0 index.

→ Negative index begins from -1. The last element in the sequence is represented by index -1.

→ Each character in a string corresponds to an index number, & each character can be accessed by its index number.

→ Two way to accessing string characters.

(1) Positive indexing :-

Ex- start | C | H | I | R | A | G | end

0 1 2 3 4 5

length = 6

(2) Negative Indexing :-

Ex-

C	H	I	R	A	G
-6	-5	-4	-3	-2	-1

→ length = 6

start

end

Ex:- positive index ~~string~~ -

a = "chirag"

a[2]

o/p 'i'

Ex- Negative index :-

a = "chirag"

a[-3]

o/p 'r'

POORNIMA

* Slicing :-

→ The term slicing refers to obtaining a subset of elements from an iterable based on their indices.

→ We create a substring by slicing a string. which is ~~efficiently~~ effectively a string that exists within another string.

→ We utilizing slicing when we only need a portion of the string & not the entire string.

Syntax -

string [start : end : step]

start - index from where to start.

end - ending index.

step - number of jumps / increment to take between i.e stepsize.

Ex- string = "chirag pareek"

string[0:6:1]

o/p 'chirag'

string[4:8:1]

o/p 'ag p'

* Call by value :-

→ In the event that you pass arguments like whole numbers, strings or tuples to a function, the passing is like call by value because you can not change the value of the immutable objects being passed to the function.

* Call by reference :-

whereas passing mutable objects can be considered as call by reference because when their values are changed inside the function, then it will also be reflected outside the function.