20-09-2024 # Training Day- 5

Binary: 0 and 1. False and True

```
# #New Program
# print(True and True)
# print(True and False)
# print(True or False)
```

Bitwise Operators: Work on bits: bit by bit

1: Convert your data into binary numbers

```
"""
# a=3        #0011
# b=2        #0010
# y = a & b   #0010   #2
# print(y)
```

………..New Program……..

```
# a,b,c,d=2,3,4,5
# y=a and b and c and d
# print(y)
```

………..New Program……..

```
# a,b,c,d=0,False,4,5
# y=a or b or c or d
# print(y)
```

Membership Operators: 2, returns True or False, bool

in

not in

check whether element or substring is part of main string or iterator check the element in an iterator, so they find elements only from iterators not from single elements data types

………..New Program……..

```
# s="CETPA InfoTech"
# r="T" in s
# print(r)
```

………..New Program……..

```
# r=5 in 456      #TypeError: argument of type 'int' is not iterable
# print(r)
```

………..New Program……..

```
# r="5" in "456"
# print(r)

"""
```

7. Identity Operators: 2 in numbers: Returns True or False:
Checks whether arguments have same identity ie address or not
is
is not
works like id function
identity operators checks whether arguments are reprsenting
the same identity/same entity or not
………..New Program……..
```
# a=5     #a address 1000
# b=a     #b address 1000
# print(a is b)
# print(a is not b)
```
………..New Program……..
```
# a=5     #a address is 1000
# b=a     #b address 1000
# print(id(a))
# print(id(b))
# print(a is b)   #Checks the addresses
# print(a==b)     #Checks the values
```

"""
Python Drawbacks:
1. Python is a very slow language.
2. Python consumes hell lot of memory.

100 dollar: 10k Man hours
300 Man hours: Cost drastically reduce
For companies these days, manpower cost matters much more than hardware infra cost.

Why Python is slow?
Because python support dynamic data type definition and dynamic memory allocation.

In Python: We need not to define the data type
………..New Program……..
```
# a=5
# print(a, type(a))
# a=2.5
# print(a, type(a))
# a="CETPA"
# print(a, type(a)
```

"""
If we want to define the data type intentionally before passing the value
"""

……….New Program……..

```
# a=int()
# print(a,type(a))
# a=float()
# print(a,type(a))
# a=list()
# print(a,type(a))
```

Python Supports Dynamic Data Type Definition: How memory is assigned to variables in Python a=5 :
How this statement will execute in the background
Step 1: It will be checked that how much memory is required to store 5 in RAM.
Step 2: Your python program will request the OS to allocate that much memory.
Step 3: OS will search in the RAM, where that much memory is available in RAM to store 5.
Step 4: OS will occupy the free memory available will allocate to python program.
Step 5: Now 5 will be stored at that location and a will start representing that location.
Say in statement, say a=5 is representing 1000 location.
Now Python supports dynamic memory allocation, once we assign a new value to same variable of same data type or another data type, all above steps will be repeated.
a=7
Python fundamental rule, whenever we assign any new value to a variables, it will be stored at a new location which actually means a new variable will be created

Benefit of Dynamic Memory Allocation:

1. We can pass any type of value to a variable
2. We can pass any big value for a variable in Python

……….New Program……..

```
# a=5    #Location 1000 location
# print(id(a))
# a=7    #Location 2000 location
# print(id(a))
a=5    #1000 location
```

b=7   #1010 location
a="Welcome to CETPA"
"""
Variable: Is a data storing element, whose value vaies in the program
a=5   #5 will be stored at 1000 location and will refer to 1000
a=7   #a will refer to 2000 location
When we run any application then in the background multiple small processes execute. In those processes of python, one process is of garbage collector. Whenever we assign a value to a variable then automatically a reference counter is assigned to the address associated with that variable. Reference counter represents the count ie how many variables are representing to a memory address.
a=5   #a address 1000, ref_counter of 1000 location will become 1
b=a   #b address 1000, ref_counter of 1000 location will become 2
#So here 1000 location is referred by 2 variables ie a and b that
#why the ref_counter value of 1000 location will become 2.
a=7   #a address 2000, ref_counter_2000=1
#ref_counter_1000=1 because now a is referring to 2000
#location and 1000 location is referred only by b ie 1 variable.
Reference counter tells that how much variables are referring to a particular memory location. Each memory location will have a different reference counter.
1. All variables in python are of ref type.If we assign one variable to another then address of RHS variable is assigned to LHS variable so variables are of reference type.
2. Python supports dynamic memory allocation, means whenever we assign a new value to a variable then it will be stored at a new location.
3. Whenever any location is not referred by any variable in the program then that location is automatically made free by the garbage collector.
"""