Untitled

March 18, 2023

0.1 logical AND

[2]:
```
True*False
```

[2]:
```
0
```

[4]:
```
## y? 1*0=0
```

[5]:
```
True*True
```

[5]:
```
1
```

[6]:
```
False*False
```

[6]:
```
0
```

[23]:
```
not True*True
```

[23]:
```
False
```

[25]:
```
not False*False
```

[25]:
```
True
```

[27]:
```
(not True)*True
```

[27]:
```
0
```

[28]:
```
(not False)*False
```

[28]:
```
0
```

[29]:
```
not(False*True)
```

[29]:
```
True
```

0.2 equality operations

0.2.1 1. is

0.2.2 2. is not

0.2.3 3. ==

0.2.4 4. !=

[30]:
```
lst_a=[1,2,3,4]
lst_b=[1,2,3,4]
```

[31]:
```
lst_a is lst_b
```

[31]:
```
False
```

[32]:
```
## false because of different id for a and b list
```

[33]:
```
lst_a is not lst_b
```

[33]:
```
True
```

[34]:
```
## true because of different id or object
```

[36]:
```
##therefore it can be said  that "is" function applicable
##for same id or same objects
##but the "is not" function applicable for different id or function
```

[37]:
```
print(id(lst_a))
print(id(lst_b))
```

```
140345127105536
140345126996544
```

[38]:
```
## this are different ids for list a and b
```

[42]:
```
## second scenerio is;
lst_a=[1,2,3,4]
lst_b=lst_a
```

[43]:
```
lst_a is lst_b
```

[43]:
```
True
```

[45]:
```
## y true? because list b is defined as list a
## their ids will aslo be same
print(id(lst_a))
print(id(lst_b))
```

```
140345433203072
140345433203072
```

[46]:
```
lst_a==lst_b
```

[46]:
```
True
```

[48]:
```
## y true? because double equals to checks the element
## list value not id
```

[53]:
```
## condition
a=2
b=2
```

[50]:
```
a==b
```

[50]:
```
True
```

[51]:
```
a is b
```

[51]:
```
True
```

[52]:
```
a is not b
```

[52]:
```
False
```

[54]:
```
print(id(a))
print(id(b))
```

```
140345476399376
140345476399376
```

[55]:
```
## becuse the id are same then it is "is" function
```

[58]:
```
## conclusion ;
## everything depends on the value!
## after = function is value!
## if the value is same for any given ffunction
#then id will be same expect for lst.
## lst is mutable(can be changed) other
## functions cannot be mutable
```

0.3 comparison operations

1. < less than ### 2. <= less than equal to ### 3. > greater than ### 4. >= greater than equal to

[ ]:
```

```

0.4 Arthmetic_Operations

[70]:
```
## // integer division
## % the modulo operator(displays the reminder)
```

[60]:
```
a=90
b=76
```

[61]:
```
a+b
```

[61]: 166

[62]:
```
a*b
```

[62]: 6840

[63]:
```
a/b
```

[63]: 1.1842105263157894

[64]:
```
a//b
```

[64]: 1

[65]:
```
b//a
```

[65]: 0

[66]:
```
b/a
```

[66]: 0.8444444444444444

[67]:
```
a%b
```

[67]: 14

[68]:
```
b%a
```

[68]: 76

[ ]:
```

```

1 Bitwise_Operations

1.0.1 1.~ bitwise complement(prefix unary operator)

1.0.2 2.& bitwise and

1.0.3 3.| bitwise or

1.0.4 4.^ bitwise exclusive-or

1.0.5 5.« shift bits,fillings in with zeros

1.0.6 6.» shif bits rights, filling in with sign bit

[71]:
```
var=10
bin(var)
```

[71]:
```
'0b1010'
```

[72]:
```
num=90
bin(num)
```

[72]:
```
'0b1011010'
```

[73]:
```
~var
```

[73]:
```
-11
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: