Day 3- Python Operators and Lists (1)

February 4, 2023

0.1 Logical Operators

1. And
2. or
3. not

[1]:
```
True and True
```

[1]:
```
True
```

[2]:
```
True or False
```

[2]:
```
True
```

[3]:
```
# Lets define two variables having boolean values True and False-
START = True
STOP = False

# Print the values as it is
print(f"Defined value of START = {START}")
print(f"Defined value of STOP = {STOP} \n")

# if can also be written as -
print(f"Defined value of START = {START}")
print(f"Value of STOP by negating START= {not START} \n")

# alternatively -
print(f"Value of START by negating STOP = {not STOP}")
print(f"Defined value of STOP = {STOP} \n")
```

```
Defined value of START = True
Defined value of STOP = False

Defined value of START = True
Value of STOP by negating START= False

Value of START by negating STOP = True
Defined value of STOP = False
```

[4]:
```
not True
```

[4]:
```
False
```

[8]:
```
not int(bool(0))
```

[8]:
```
True
```

[11]:
```
not int(bool(1))
```

[11]:
```
False
```

[12]:
```
not 1
```

[12]:
```
False
```

[15]:
```
not 0
```

[15]:
```
True
```

[17]:
```
not 1000
```

[17]:
```
False
```

[18]:
```
not -1
```

[18]:
```
False
```

[20]:
```
START = 1
STOP = 0

# Print the values as it is
print(f"Defined value of START = {START}")
print(f"Defined value of STOP = {STOP} \n")

# if can also be written as -
print(f"Defined value of START = {START}")
print(f"Value of STOP by negating START= {int(not START)} \n")

# alternatively -
print(f"Value of START by negating STOP = {int(not STOP)}")
print(f"Defined value of STOP = {STOP} \n")
```

```
Defined value of START = 1
Defined value of STOP = 0

Defined value of START = 1
Value of STOP by negating START= 0
Value of START by negating STOP = 1
Defined value of STOP = 0
```

[21]:
```
zero = 0
one = 1

print(f"boolean value of no. {zero} is {bool(zero)}")
print(f"boolean value of no. {one} is {bool(one)}")
print(f"negation of {zero} is {not zero} and negation of {one} is {not one}")
print("\n#------------------------#\n")
```

```
boolean value of no. 0 is False
boolean value of no. 1 is True
negation of 0 is True and negation of 1 is False

#------------------------#
```

[28]:
```
not 5
```

[28]:
```
False
```

[22]:
```
Some_negative_integer = -5
Some_positive_integer = 5

print(f"boolean value of no. {Some_negative_integer} is␣
 ↪{bool(Some_negative_integer)}")
print(f"boolean value of no. {Some_positive_integer} is␣
 ↪{bool(Some_positive_integer)}")
print(f"negation of {Some_negative_integer} is {not Some_negative_integer} \
and negation of {Some_positive_integer} is {not Some_positive_integer}")

print("\n#------------------------#\n")
```

```
boolean value of no. -5 is True
boolean value of no. 5 is True
negation of -5 is False and negation of 5 is False

#------------------------#
```

[29]:
```
Some_negative_float = -5.99
Some_positive_float = 5.6

print(f"boolean value of no. {Some_negative_float} is␣
 ↪{bool(Some_negative_float)}")
```

```
print(f"boolean value of no. {Some_positive_float} is␣
 ↪{bool(Some_positive_float)}")
print(f"negation of {Some_negative_float} is {not Some_negative_float} \
and negation of {Some_positive_float} is {not Some_positive_float}")
```

```
boolean value of no. -5.99 is True
boolean value of no. 5.6 is True
negation of -5.99 is False and negation of 5.6 is False
```

0.2 Logical And

[30]:
```
VEGETABLES = True
SALT = False
DISH = VEGETABLES and SALT

print(f"Dish contains VEGETABLES: {VEGETABLES}")
print(f"Dish contains SALT: {SALT}")
print(f"Hence dish prepared was good: {DISH}\n")
```

```
Dish contains VEGETABLES: True
Dish contains SALT: False
Hence dish prepared was good: False
```

[34]:
```
not(False) * True
```

[34]:
```
True
```

[50]:
```
False * (not(False))
```

[50]:
```
0
```

[42]:
```
not(True) * False
```

[42]:
```
True
```

[45]:
```
False * False
```

[45]:
```
0
```

[49]:
```
not False * False
```

[49]:
```
True
```

[48]:
```
(1+2)*3
```

[48]:
```
9
```

```
[52]: not(False)*False == True
```

```
[52]: True
```

```
[54]: print((not(False)) * False)
```

```
0
```

```
[58]: (2*3+40)/5
```

```
[58]: 9.2
```

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

```
[61]: True
```

```
[63]: not(1)
```

```
[63]: False
```

```
[65]: not False * True == True
```

```
[65]: True
```

0.2.1 4.1.2 Equality Operators

Following operations are present in python for equlity check operation-

Operators	Meaning
is	***a is b*** returns true if variable/identifiers a and b *points* to the *same object*
is not	***a is not b*** returns true if variable/identifiers a and b *points* to the *different object*
==	***a == b*** returns true if variable/identifiers a and b has same value
!=	***a != b*** returns true if variable/identifiers a and b has different value

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

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

```
140428320195968
140428363610688
```

[69]:
```
lst_a is lst_b
```

[69]:
```
False
```

[70]:
```
lst_a=[1,2,3,4]
lst_b=lst_a
```

[71]:
```
print(id(lst_a))
print(id(lst_b))
```

```
140428064461056
140428064461056
```

[72]:
```
lst_a is lst_b
```

[72]:
```
True
```

[73]:
```
lst_a=[1,2,3,4]
lst_b=[1,2,3,4]

lst_a is not lst_b
```

[73]:
```
True
```

[74]:
```
lst_a == lst_b
```

[74]:
```
True
```

[75]:
```
a=2
b=2
```

[76]:
```
print(id(a))
print(id(b))
```

```
140428412780816
140428412780816
```

[77]:
```
str1="Krish"
str2="Krish"
print(id(str1))
print(id(str2))
```

```
140428053242224
140428053242224
```

[78]:
```
## immutable
str1="Krish"
str2="Krish1"
```

```
[79]: print(id(str1))
      print(id(str2))

      140428053242224
      140428053240112

[80]: lst_a

[80]: [1, 2, 3, 4]

[82]: lst_a[0]=10

[83]: lst_a

[83]: [10, 2, 3, 4]

[84]: str1

[84]: 'Krish'

[86]: str1[0]='N' ## immutable

      ---------------------------------------------------------------------------
      TypeError                                 Traceback (most recent call last)
      Cell In[86], line 1
      ----> 1 str1[0]='N'

      TypeError: 'str' object does not support item assignment

[87]: ## Comparison operation

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```

Operation	Meaning
<	less than
<=	less than or equal to
>	greater than

Operation	Meaning
>=	greater than or equal to

```
[91]: maxium_speed_of_bike = 150
      max_speed_of_car = 200

      print(f"bike is faster than car: {maxium_speed_of_bike >= max_speed_of_car}")
```

```
bike is faster than car: False
```

```
[95]: maxium_speed_of_bike = 200
      max_speed_of_car = 200

      print(f"bike is faster than car: {maxium_speed_of_bike > max_speed_of_car}")
```

```
bike is faster than car: False
```

0.3 Arithmethic Operations

Operation	Meaning
+	addition
-	subtraction
*	multiplication
/	true division
//	integer division
%	the modulo operator

```
[96]: a=25
      b=101
```

```
[97]: b*a
```

```
[97]: 2525
```

```
[98]: b+a
```

```
[98]: 126
```

```
[100]: a/b
```

```
[100]: 0.24752475247524752
```

```
[101]: a//b
```

```
[101]: 0
```

```
[102]: b//a
```

```
[102]: 4
```

```
[103]: b/a
```

```
[103]: 4.04
```

```
[104]: b%a
```

```
[104]: 1
```

```
[106]: ## Display the remainder
       a%b
```

```
[106]: 25
```

0.3.1 Bitwise Operators

Operation	Meaning
	bitwise complement (prefix unary operator)
&	bitwise and
|	bitwise or
ˆ	bitwise exclusive-or
«	shift bits left, filling in with zeros
»	shift bits right, filling in with sign bit

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

```
[107]: '0b1010'
```

```
[108]: ~var
```

```
[108]: -11
```

```
[110]: True | False
```

```
[110]: True
```

```
[111]: False & False
```

```
[111]: False
```

```
[113]: var >>1
```

```
[113]: 5
```

```
[114]: var << 1
```

```
[114]: 20
```

0.4 Strings

```
[116]: "Welcome to the Data Science masters"
```

```
[116]: 'Welcome to the Data Science masters'
```

```
[117]: str1="Welcome to Data Science Masters"
```

```
[119]: type(str1)
```

```
[119]: str
```

```
[121]: ## immutable
       str1[0]=12
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[121], line 1
----> 1 str1[0]=12

TypeError: 'str' object does not support item assignment
```

```
[122]: str1="Krish Naik"
```

```
[123]: str1
```

```
[123]: 'Krish Naik'
```

```
[126]: welcome="Hello World"
```

```
[127]: dir(welcome)
```

```
[127]: ['__add__',
        '__class__',
        '__contains__',
        '__delattr__',
        '__dir__',
        '__doc__',
        '__eq__',
        '__format__',
        '__ge__',
        '__getattribute__',
        '__getitem__',
        '__getnewargs__',
        '__gt__',
        '__hash__',
        '__init__',
        '__init_subclass__',
        '__iter__',
        '__le__',
        '__len__',
        '__lt__',
        '__mod__',
        '__mul__',
        '__ne__',
        '__new__',
        '__reduce__',
        '__reduce_ex__',
        '__repr__',
        '__rmod__',
        '__rmul__',
        '__setattr__',
        '__sizeof__',
        '__str__',
        '__subclasshook__',
        'capitalize',
        'casefold',
        'center',
        'count',
        'encode',
        'endswith',
        'expandtabs',
        'find',
        'format',
        'format_map',
        'index',
        'isalnum',
        'isalpha',
        'isascii',
        'isdecimal',
        'isdigit',
        'isidentifier',
        'islower',
        'isnumeric',
        'isprintable',
        'isspace',
        'istitle',
        'isupper',
        'join',
        'ljust',
        'lower',
        'lstrip',
        'maketrans',
        'partition',
        'removeprefix',
        'removesuffix',
        'replace',
        'rfind',
        'rindex',
        'rjust',
        'rpartition',
        'rsplit',
        'rstrip',
        'split',
        'splitlines',
        'startswith',
        'strip',
        'swapcase',
        'title',
        'translate',
        'upper',
        'zfill']
```

[128]:
```
string ="Pw Skills Data Science"
```

[131]:
```
string[5]
```

[131]:
```
'i'
```

[133]:
```
## slice operation
string[5:11]
```

[133]:
```
'ills D'
```

[134]:
```
string[-1]
```

[134]:
```
'e'
```

[136]:
```
string[-3:]
```

[136]:
```
'nce'
```

[138]:
```
string[-7:-3]
```

```
[138]: 'Scie'
```

```
[242]: string
```

```
[242]: 'Pw Skills Data Science'
```

```
[251]: string[-7:5]
```

```
[251]: 'S'
```

```
[140]: string[7:5]
```

```
[140]: ''
```

```
[141]: string[-10:10]
```

```
[141]: ''
```

```
[142]: string[:-1]
```

```
[142]: 'Pw Skills Data Scienc'
```

```
[143]: string[:]
```

```
[143]: 'Pw Skills Data Science'
```

```
[241]: course_name
```

```
[241]: 'Data Science Masters'
```

```
[240]: name="Karthik"
        name[-5:4]
```

```
[240]: 'rt'
```

```
[145]: string
```

```
[145]: 'Pw Skills Data Science'
```

```
[148]: string[::-1]
```

```
[148]: 'ecneicS ataD sllikS wP'
```

```
[149]: string[::-2]
```

```
[149]: 'eniSaa liSw'
```

```
[156]: string[::-1]
```

```
[156]: 'ecneicS ataD sllikS wP'
```

```
[157]: string[::-3]
```

```
[157]: 'eeSt lSP'
```

```
[160]: string[::5]
```

```
[160]: 'PiDSc'
```

```
[161]: name="Krish"
```

```
[170]: name[::-2]
```

```
[170]: 'hiK'
```

```
[174]: name[3:1:-1]
```

```
[174]: 'si'
```

```
[175]: course_name="Data Science Masters"
```

```
[235]: name="Karthik"
```

```
[239]: name[::-1]
```

```
[239]: 'kihtraK'
```

```
[233]: course_name[12:4:-1]
```

```
[233]: ' ecneicS'
```

```
[179]: course_name[5:12]
```

```
[179]: 'Science'
```

```
[188]: course_name
```

```
[188]: 'Data Science Masters'
```

```
[184]: course_name[11:4:-1]
```

```
[184]: 'ecneicS'
```

```
[185]: course_name[12:4:-1]
```

[185]: ' ecneicS'

[187]:
```
course_name[12:5:-1]
```

[187]: ' ecneic'

[194]:
```
## concatentaion
course_name + "Course"
```

[194]: 'Data Science MastersCourse'

[192]:
```
course_name[4:12:1]
```

[192]: ' Science'

[195]:
```
print("Hello" + "Worlds")
```

```
HelloWorlds
```

[197]:
```
course_name *5
```

[197]: 'Data Science MastersData Science MastersData Science MastersData Science MastersData Science Masters'

[198]:
```
len(course_name)
```

[198]: 20

[199]:
```
## find function
course_name.find("n")
```

[199]: 9

[201]:
```
course_name.find("a",2,10)
```

[201]: 3

[202]:
```
course_name.find("z")
```

[202]: -1

[204]:
```
course_name
```

[204]: 'Data Science Masters'

[203]:
```
## count()
course_name.count('a')
```

```
[203]: 3
```

```
[206]: course_name.count(' ')
```

```
[206]: 2
```

```
[207]: course_name.count('')
```

```
[207]: 21
```

```
[225]: course_name
```

```
[225]: 'Data Science Masters'
```

```
[227]: course_name[::-1]
```

```
[227]: 'sretsaM ecneicS ataD'
```

```
[ ]: # string split function
```

```
[211]: course_name.split(' ')
```

```
[211]: ['Data', 'Science', 'Masters']
```

```
[213]: course_name
```

```
[213]: 'Data Science Masters'
```

```
[212]: course_name.split('S')
```

```
[212]: ['Data ', 'cience Masters']
```

```
[216]: course_name.split('s')
```

```
[216]: ['Data Science Ma', 'ter', '']
```

```
[215]: course_name.partition('s')
```

```
[215]: ('Data Science Ma', 's', 'ters')
```

```
[217]: ## SString upper and lowercase
       course_name.upper()
```

```
[217]: 'DATA SCIENCE MASTERS'
```

```
[218]: course_name.lower()
```

[218]: 'data science masters'

[219]:
```
course_name
```

[219]: 'Data Science Masters'

[221]:
```
course_name.swapcase()
```

[221]: 'dATA sCIENCE mASTERS'

[222]:
```
course_name.title()
```

[222]: 'Data Science Masters'

[223]:
```
name="krish nbaik"
```

[224]:
```
name.title()
```

[224]: 'Krish Nbaik'

[252]:
```
bin(-2)
```

[252]: '-0b10'

[253]:
```
len(course_name)
```

[253]: 20

[258]:
```
course_name.count('')
```

[258]: 21

[266]:
```
course_name[-5:10]
```

[266]: ''

[262]:
```
course_name
```

[262]: 'Data Science Masters'

[268]:
```
## Assignments
size = int(input("Enter the triangle Length:"))
for i in range(size):
    for j in range(size-i):
        print("",end=" ")
    for k in range(i+1) :
        print("*",end="")
    for m in range(k-1):
        print("*", end="")
    print()
```

```
Enter the triangle Length: 9
         *
        **
       ****
      ******
     ********
    **********
   ************
  **************
 ****************
```

[269]:
```
course_name.upper().
```

[269]:
```
1
```

[270]:
```
## solution 2
n = 10
for i in range(n):
    print(" "*(n-i),end='')
    for j in range(i*2+1):
        print("*",end="")
    print()
```

```
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
  *****************
 *******************
```

[271]:
```
course_name
```

[271]:
```
'Data Science Masters'
```

[ ]:
```

```

[ ]:
```

```