In [ ]:
```
# Q1. Declare a boolean value and store it in a variable.
# Check the type and print the id of the same.
```

In [2]:
```
x = True
print(x)
print(type(x))
print(id(x))
```

```
True
<class 'bool'>
140736202222112
```

In [ ]:
```

```

In [ ]:
```

# Q2. Take one boolean value between 0 - 256.
# Assign it to two different variables.
# Check the id of both the variables. It should come the same. Check why?
```

In [1]:
```
#solution1

a= 34
b= 34

print(id(a))
print(id(b))
```

```
140712740689736
140712740689736
```

In [2]:
```
#solution2

x = True
y = True

print(id(x))
print(id(y))
```

```
140712739220000
140712739220000
```

In [ ]:
```
# Q3. Arithmetic Operations on boolean data
# Take two different boolean values.
# Store them in two different variables.
# Do below operations on them:-
#  Find sum of both values
#  Find difference between them
#  Find the product of both.
#  Find value after dividing first value with second value
#  Find the remainder after dividing first value with second value
#  Find the quotient after dividing first value with second value
#  Find the result of first value to the power of second value.
```

In [2]:
```
# Take two different boolean values.
# Store them in two different variables.
x=True
y=False
```

In [3]:
```
# sum of both numbers
print(x+y)
# difference between them
print(x-y)
# the product of both numbers.
print(x-y)
# value after dividing first num with second number
print(x/y)
```

```
1
1
1

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[3], line 8
      6 print(x-y)
      7 # value after dividing first num with second number
----> 8 print(x/y)
      9 # the remainder after dividing first number with second number
     10 print(x%y)

ZeroDivisionError: division by zero
```

In [4]:
```
# the remainder after dividing first number with second number
print(x%y)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[4], line 2
      1 # the remainder after dividing first number with second number
----> 2 print(x%y)
      3 # the quotient after dividing first number with second number
      4 print(x//y)

ZeroDivisionError: integer modulo by zero
```

In [5]:
```
# the quotient after dividing first number with second number
print(x//y)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[5], line 2
      1 # the quotient after dividing first number with second number
----> 2 print(x//y)

ZeroDivisionError: integer division or modulo by zero
```

In [3]:
```
# the result of the first num to the power of the second number.
print(x**y)
```

```
1
```

In [ ]:
```

# Q4. Comparison Operators on boolean values
# Take two different boolean values.
# Store them in two different variables.
# Do below operations on them:-
#  Compare these two values with below operator:-
#  Greater than, '>'
#  less than, '<'
#  Greater than or equal to, '>='
#  Less than or equal to, '<='
# Observe their output(return type should be boolean)
```

In [12]:
```
# Take two different boolean values.
# Store them in two different variables.
x=True
y=False
```

In [ ]:
```
Compare these two values with below operator:-
#  Greater than, '>'
#  less than, '<'
#  Greater than or equal to, '>='
#  Less than or equal to, '<='
# Observe their output(return type should be boolean)
```

In [13]:
```
print(x>y)
print(x<y)
print(x>=y)
print(x<=y)
print(type(x))
print(type(y))
```

```
True
False
True
False
<class 'bool'>
<class 'bool'>
```

In [ ]:
```

```

In [ ]:
```

# Q5. Equality Operator
# Take two different boolean values.
# Store them in two different variables.
# Equate them using equality operators (==, !=)
# Observe the output(return type should be boolean)
```

In [14]:
```
# Take two different boolean values.
# Store them in two different variables.
x=True
y=False
```

In [16]:
```
# Equate them using equality operators (==, !=)
# Observe the output(return type should be boolean)
print(x)
print(y)
x==y
print(x)
print(y)
x!=y
print(x)
print(y)
```

```
True
False
True
False
True
False
```

In [ ]:

In [ ]:
```
# Q6. Logical operators
# Observe the output of below code
# Cross check the output manually
# print(True and True)
# #---------------------------------------->Output is True
# print(False and True)
# #---------------------------------------->Output is False
# print(True and False)
# #---------------------------------------->Output is False
# print(False and False)
# #---------------------------------------->Output is False
# print(True or True)
# #---------------------------------------->Output is True
# print(False or True)
# #---------------------------------------->Output is True
# print(True or False)
# #---------------------------------------->Output is True
# print(False or False)
# #---------------------------------------->Output is False
# print(not True)
# #---------------------------------------->Output is False
# print(not False)
# #---------------------------------------->Output is True
```

In [17]:
```
print(True and True)
print(False and True)
print(True and False)
print(False and False)
print(True or True)
print(False or True)
print(True or False)
print(False or False)
print(not True)
print(not False)
```

```
True
False
False
False
True
True
True
False
False
True
```

In [ ]:
```
# Q7. Bitwise Operators
# Do below operations on the values provided below:-
#  Bitwise and(&) --------------> True, True -------> Output is True
#  Bitwise or(|) --------------> True, False -------> Output is True
#  Bitwise(^) --------------> True, False -------> Output is True
#  Bitwise negation(~) ---------> True -------> Output is -2
#  Bitwise left shift ---------> True,2 -------> Output is 4
#  Bitwise right shift ---------> True,2 -------> Output is 0
# Cross check the output manually
```

In [1]:
```
print(True & True)
print(True | False)
print(True ^ False)
print(~True)
print(True << 2)
print(True >> 2)
```

```
True
True
True
-2
4
0
```

In [29]:
```
print(101 and 101)
```

```
101
```

In [ ]:
```


# Q8. What is the output of expression inside the print statement. Cross
# check before running the program.
# a = True
# b = True
# print(a is b) #True or False? #
# print(a is not b) #True or False?
# a = False
# b = False
# print(a is b) #True or False?
# print(a is not b) #True or False?
```

In [28]:
```
a = True
b = True
print(a is b)
print(a is not b)
a = False
b = False
print(a is b)
print(a is not b)
```

```
True
False
True
False
```

In [ ]:
```
# Q9. Membership operation
# in, not in are two membership operators and it returns boolean value
# print(True in [10,10.20,10+20j,'Python', True])
# print(False in (10,10.20,10+20j,'Python', False))
# print(True in {1,2,3, True})
# print(True in {True:100, False:200, True:300})
# print(False in {True:100, False:200, True:300})
```

In [30]:
```
print(True in [10,10.20,10+20j,'Python', True])
print(False in (10,10.20,10+20j,'Python', False))
print(True in {1,2,3, True})
print(True in {True:100, False:200, True:300})
print(False in {True:100, False:200, True:300})
```

```
True
True
True
True
True
```

In [ ]:

In [ ]: