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

In [1]:
```
x = 50
print(type(x))
print(id(x))
```

```
<class 'int'>
140726337182024
```

In [ ]:
```
Q2. Take one int 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 [2]:
```
a = 100
b = 100
print(id(a))
print(id(b))
```

```
140726337183624
140726337183624
```

In [ ]:
```
Q3. Take one int value either less than -5 or greater than 256.
Assign it to two different variables.
Check the id of both the variables. It should come different.Check why?
```

In [3]:
```
a = -100
b = -100
print(id(a))
print(id(b))
```

```
1579056679920
1579056678896
```

In [ ]:
```
Q4. Arithmetic Operations on integers
Take two different integer values.
Store them in two different variables.
Do below operations on them:-
 Find sum of both numbers
 Find difference between them
 Find the product of both numbers.
 Find value after dividing first num with second number
 Find the remainder after dividing first number with second number
 Find the quotient after dividing first number with second number
 Find the result of the first num to the power of the second number.
```

In [4]:
```
x=10
y=2

# 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)
# the remainder after dividing first number with second number
print(x%y)
# the quotient after dividing first number with second number
print(x//y)
# the result of the first num to the power of the second number.
print(x*y)
```

```
12
8
8
5.0
0
5
20
```

In [ ]:
```
Q5. Comparison Operators on integers
Take two different integer values.
Store them in two different variables.
Do below operations on them:-
 Compare the two numbers with below operator:-
 Greater than, '>'
 Smaller than, '<'
 Greater than or equal to, '>='
 Less than or equal to, '<='
Observe their output(return type should be boolean)
```

In [5]:
```
x=15
y=20

#compare
print(x>y)
print(x<y)
print(x>=y)
print(x<=y)
```

```
False
True
False
True
```

In [ ]:
```
Q6. Equality Operator
Take two different integer values.
Store them in two different variables.
Equate them using equality operators (==, !=)
Observe the output(return type should be boolean)
```

In [6]:
```
x=15
y=20

#Equate
print(x==y)
print(x!=y)
```

```
False
True
```

In [ ]:
```
Q7. Logical operators
Observe the output of below code
Cross check the output manually
print(10 and 20)
#---------------------------------------->Output is 20
print(0 and 20)
#---------------------------------------->Output is 0
print(20 and 0)
#---------------------------------------->Output is 0
print(0 and 0)
#---------------------------------------->Output is 0
print(10 or 20)
#---------------------------------------->Output is 10
print(0 or 20)
#---------------------------------------->Output is 20
print(20 or 0)
#---------------------------------------->Output is 20
print(0 or 0)
#---------------------------------------->Output is 0
print(not 10)
#---------------------------------------->Output is False
print(not 0)
#---------------------------------------->Output is True
```

In [7]:
```
print(10 and 20)
print(0 and 20)
print(20 and 0)
print(0 and 0)
print(10 or 20)
print(0 or 20)
print(20 or 0)
print(0 or 0)
print(not 10)
print(not 0)
```

```
20
0
0
0
10
20
20
0
False
True
```

In [ ]:
```
Q8. Bitwise Operators
Do below operations on the values provided below:-
 Bitwise and(&) ----------------------------------------> 10, 20
-------> Output is 0
 Bitwise or(|) ----------------------------------------> 10, 20
-------> Output is 30
 Bitwise(^) ----------------------------------------> 10, 20
-------> Output is 30
 Bitwise negation(~) ----------------------------------> 10
-------> Output is -11
 Bitwise left shift ----------------------------------> 10,2
-------> Output is 40
 Bitwise right shift ----------------------------------> 10,2
-------> Output is 2
Cross check the output manually
```

In [4]:
```
print(bin(10))
print(bin(20))
print(bin(40))
print(bin(2))
print(bin(20))
print(int(0b11110))
print(int(0b101000))
print(int(-0b1011))
print(int(0b0101))
print(bin(-11))
print(bin(10))
print(bin(30))
```

```
0b1010
0b10100
0b101000
0b10
0b10100
30
40
-11
5
-0b1011
0b1010
0b11110
```

In [3]:
```
print(int(0b10))

0b01010
0b10100
```

```
2
```

In [13]:
```
print(10 & 20)
print(10 | 20)
print(10 ^ 20)
print(~10)
print(10 << 2)
print(10 >> 2)
```

```
0
30
30
-11
40
2
```

In [ ]:
```
Q9. What is the output of expression inside print statement. Cross check
before running the program.
a = 10
b = 10
print(a is b) #True or False?
print(a is not b) #True or False?
a = 1000
b = 1000
print(a is b) #True or False?
print(a is not b) #True or False?
```

In [14]:
```
a = 10
b = 10
print(a is b)
print(a is not b)
a = 1000
b = 1000
print(a is b)
print(a is not b)
```

```
True
False
False
True
```

In [ ]:
```
Q10. What is the output of expression inside print statement. Cross check
before running the program.
print(10+(10*32)//2**5&20+(~(-10))<<2)
```

In [15]:
```
print(10+(10*32)//2**5&20+(~(-10))<<2)
```

```
20
```

In [ ]:
```
Q11. Membership operation
in, not in are two membership operators and it returns boolean value
print('2' in 'Python2.7.8')
print(10 in [10,10.20,10+20j,'Python'])
print(10 in (10,10.20,10+20j,'Python'))
print(2 in {1,2,3})
print(3 in {1:100, 2:200, 3:300})
print(10 in range(20))
```

In [5]:
```
print('2.8' in 'Python2.7.8')
print(10 in [10,10.20,10+20j,'Python'])
print(10 in (10,10.20,10+20j,'Python'))
print(2 in {1,2,3})
print(3 in {1:100, 2:200, 3:300})
print(10 in range(20))
```

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

In [16]:
```
print('2' in 'Python2.7.8')
print(10 in [10,10.20,10+20j,'Python'])
print(10 in (10,10.20,10+20j,'Python'))
print(2 in {1,2,3})
print(3 in {1:100, 2:200, 3:300})
print(10 in range(20))
```

```
True
True
True
True
True
True
```

In [ ]:
```
Q12. An integer can be represented in binary, octal or hexadecimal form.
Declare one binary, one octal and one hexadecimal value and store them
in three different variables.
Convert 9876 to its binary, octal and hexadecimal equivalent and print
their corresponding value.
```

In [17]:
```
# Declare binary, octal, and hexadecimal values
binary_value = 0b1101  # Binary representation of 13
octal_value = 0o35     # Octal representation of 29
hexadecimal_value = 0xD  # Hexadecimal representation of 13

# Convert 9876 to binary, octal, and hexadecimal
decimal_value = 9876
binary_equivalent = bin(decimal_value)
octal_equivalent = oct(decimal_value)
hexadecimal_equivalent = hex(decimal_value)

# Print the results
print(f"Binary representation of 9876: {binary_equivalent}")
print(f"Octal representation of 9876: {octal_equivalent}")
print(f"Hexadecimal representation of 9876: {hexadecimal_equivalent}")
```

```
Binary representation of 9876: 0b10011010010100
Octal representation of 9876: 0o23224
Hexadecimal representation of 9876: 0x2694
```

In [ ]:
```
Q13. What will be the output of following:-
a = 0b1010000
print(a)
b = 0o7436
print(b)
c = 0xfade
print(c)
print(bin(80))
print(oct(3870))
print(hex(64222))
print(bin(0b1010000))
print(bin(0xfade))
print(oct(0xfade))
print(oct(0o7436))
print(hex(0b1010000))
print(hex(0xfade))
```

In [18]:
```
a = 0b1010000
print(a)
b = 0o7436
print(b)
c = 0xfade
print(c)
print(bin(80))
print(oct(3870))
print(hex(64222))
print(bin(0b1010000))
print(bin(0xfade))
print(oct(0xfade))
print(oct(0o7436))
print(hex(0b1010000))
print(hex(0xfade))
```

```
80
3870
64222
0b1010000
0o7436
0xfade
0b1010000
0b1111101011011110
0o175336
0o7436
0x50
0xfade
```

In [ ]: