Ternary Operator

- It is called a ternary operator because it uses 3 operands or values in it
- It is used to generate a result based on some condition

Syntax for a conditional statement with 2 operands

```
expression_1 if condition_is_True else expression_2
```

- When the if condition is True, then expression_1 is the result
- Otherwise, expression_2 will be the result

In [2]:
```
# WAP to see that the number given by the user is even or odd

num = int(input('Enter a number: '))

print('Even' if num%2 == 0 else 'Odd')
```

```
Enter a number: 12
Even
```

Syntax for a conditional statement with 3 operands

```
expression_1 if condition_1 else expression_2 if condition_2 else exp
ression_3
```

- When the condition_1 is True, then expression_1 is the result
- When the condition_1 is False, then condition_2 is checked and if condition_2 is True, then expression_2 is the result
- When both the given conditions are False, then expression_3 is the result

In [4]:
```
a = 100
b = 150
c = 136

# find the maximum number among them

print(a if (a>b and a>c) else b if (b>a and b>c) else c )
```

```
150
```

Unary minus (-)

- It will take one operand or value
- It will negate the value that we have passed

In [5]:
```
n = 10
num = -10
print(num)
```

```
-10
```

Precedence of Different Operator

- Paranthese()
- Exponential **
- Unary minus(-), bitwise complement (~)
- Multiplication(*), Float Divison(/), FLoor Division(//), Modulus(%)
- Addition(+), Subtraction(-)
- Bitwise Left shift operator (<<), Bitwise Right Shift operator(>>)
- Bitwise AND oprerator (&)
- Bitwise XOR operator (^)
- Bitwise OR operator (|)
- Comparision Operators and equality Operators (>, >=, <, <=, ==, !=)
- Assignment operator (=) , compounding operators (%=, /=, //=, -=, +=, **=, *=)
- Identity Operator (is, is not)
- Membership Operator (in , not in)
- Logical NOT
- Logical or
- Logical And

Strings

```
- Anything written inside the single quotes('...'), double quotes
("...."), single triple quotes ('''.....''') and double triple quotes
("""....."""")
- Though single quotes and double quotes will be used generally for s
trings, single triple quotes and double triple quotes will be used fo
r passing multi-line strings or Doc-strings
```

In [6]:
```
a = 'Hello'
print(a, type(a))
```

```
Hello <class 'str'>
```

In [7]:
```
a = "Hello"
print(a, type(a))
```

```
Hello <class 'str'>
```

In [8]:
```
a = 'My name is Mayank Atul Ghai'
print(a, type(a))
```

```
My name is Mayank Atul Ghai <class 'str'>
```

In [9]:
```
a = "I am a Sr. Data Analyst"
print(a, type(a))
```

```
I am a Sr. Data Analyst <class 'str'>
```

In [11]:
```
a = '''Hello world
Welcome, to python class by Mayank'''
print(a, type(a))
```

```
Hello world
Welcome, to python class by Mayank <class 'str'>
```

In [12]:
```
a = """We have done with operator
and we are doing strings"""
print(a, type(a))
```

```
We have done with operator
and we are doing strings <class 'str'>
```

In [13]:
```
a = 'I'm mayank atul ghai'
print(a, type(a))
```

```
  Cell In[13], line 1
    a = 'I'm mayank atul ghai'
                ^
SyntaxError: unterminated string literal (detected at line 1)
```

In [14]:
```
a = "I'm mayank atul ghai"
print(a, type(a))
```

```
I'm mayank atul ghai <class 'str'>
```

In [ ]:
```
__Note__:

    Strings store a the whole sentence word by word
```

Operations that we can do on Strings

- Length
- Concatenate
- Repetition

Length

- How many elements or characters are there inside a string
- **len()** function is used to find the length of the string

Syntax:

```
a = '......' or "......"
print(len(a))
```

len(a) ---> check the value stored in a ---> count the number of elements in the container datatype ---> return the length of that container datatype

In [15]:
```
a = "mayank atul ghai" # 'M'+'a'+'y'+'q'+'n'+'k'+' '+'A'+'T'+'u'+'l'+' '+
print(len(a))
```

```
16
```

In [17]:
```
b = 'python'
print(len(b))
```

```
6
```

In [18]:
```
a = input("enter a word or a sentemce: ")
print(a, len(a))
```

```
enter a word or a sentemce: Learnbay
Learnbay 8
```

In [19]:
```
a = input("enter a number: ")
print(a, len(a))
```

```
enter a number: 100
100 3
```

In [20]:
```
a = int(input("enter a number: "))
print(a, len(a))
```

```
enter a number: 100

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 2
      1 a = int(input("enter a number: "))
----> 2 print(a, len(a))

TypeError: object of type 'int' has no len()
```

Concatenation

- Join two or more strings together then this concept is used and this concept is known as concatenation
- We have to use + operator (here in strings, + is not an arithematic operator)

In [21]:
```
my_str = "hello"
my_str2 = 'World'

print(my_str+my_str2)
```

```
helloWorld
```

In [22]:
```
my_str = "hello"
my_str2 = 'World'

print(my_str+' '+my_str2)
```

```
hello World
```

In [23]:
```
print('10'+'50')
```

```
1050
```

In [24]:
```
print(10+'50')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[24], line 1
----> 1 print(10+'50')

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [25]:
```
print(str(10)+'50')
```

```
1050
```

Reptition

- (*) operator is used with the strings to repeat a string over and over for a specific number of times

In [28]:
```
a = 'Python_class '
print(a*5)
```

```
Python_class Python_class Python_class Python_class Python_class
```

Indexing and Slicing

Indexing

```
- A string is an ordered collection of data(characters) because it fo
llows indexing
- It is also known as sequential datatype
- Each character or an element inside a string is stored at a particu
lar index
- String supports both Positive Indexing and Negative Indexing
```

```
M A Y A N K
0 1 2 3 4 5
```

Positive Indexing

- It will start from left to right
- It will start from 0 (by default)
- The index of the last element is going to be len(string)-1

```
0 1 2 3 4 5 -----> Positive indexing
M A Y A N K
```

In [29]:
```
a = 'Mayank'
```

In [30]:
```
1 a[2]
```

Out[30]: 'y'

In [31]:
```
1 a[5]
```

Out[31]: 'k'

In [32]:
```
1 a[6]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[32], line 1
----> 1 a[6]

IndexError: string index out of range
```

Negative Indexing

- It will going to start from right to left
- By default, the index of the first element will be -1
- The index of the last element is going to be -len(String)

```
 M  A  Y  A  N  K
-6 -5 -4 -3 -2 -1 -----> Negative indexing
```

In [34]:
```
1 a
```

Out[34]: 'Mayank'

In [35]:
```
1 a[-4]
```

Out[35]: 'y'

In [38]:
```
1 a[-2]
```

Out[38]: 'n'

```
 0  1  2  3  4  5 -----> Positive indexing
 M  A  Y  A  N  K
-6 -5 -4 -3 -2 -1 -----> Negative indexing
```

In [40]:
```
a
```

Out[40]: 'Mayank'

In [41]:
```
print(a[0]) # Positive indexing
print(a[-6]) # Negative Indexing
```

```
M
M
```

enumerate function

- helps to get both the index and the character at that particular index
- It will always going to follow positive indexing only

In [42]:
```
a = "Mayank Atul Ghai"

for i, j in enumerate(a):
    print(i, j)
```

```
0 M
1 a
2 y
3 a
4 n
5 k
6
7 A
8 t
9 u
10 l
11
12 G
13 h
14 a
15 i
```

In [43]:
```
a['M']
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[43], line 1
----> 1 a['M']

TypeError: string indices must be integers, not 'str'
```

In [44]:
```
a = "Mayank Atul Ghai"

for x, y in enumerate(a):
    print(x, y)
```

```
0 M
1 a
2 y
3 a
4 n
5 k
6
7 A
8 t
9 u
10 l
11
12 G
13 h
14 a
15 i
```

In [45]:
```
a = "Mayank Atul Ghai"

for index, char in enumerate(a):
    print(index, char)
```

```
0 M
1 a
2 y
3 a
4 n
5 k
6
7 A
8 t
9 u
10 l
11
12 G
13 h
14 a
15 i
```

Slicing

- Fetch a sub-string from a string

Positive Slicing

- it will use positive indexing

Syntax:

```
    my_str[starting_index:ending_index(excluded):step_size]

where
    starting index - index from where we want to start the slicing (b
y default, it is going to take 0)
    ending index - index till which i want to slice to be executed (e
xcluded---- n+1) (by default it will go
    till the len(string))
    step_size - how many jumps I want to make while slicing (by defau
lt, the value of step_Size is 1)
```

In [46]:
```
1 a
```

Out[46]: 'Mayank Atul Ghai'

In [47]:
```
1 a[::]
```

Out[47]: 'Mayank Atul Ghai'

In [51]:
```
1 print(a[:6])
2 print(a[0:6])
3 print(a[0:6:1])
```

```
Mayank
Mayank
Mayank
```

In [53]:
```
1 len(a)
```

Out[53]: 16

In [54]:
```
1 print(a[12:])
2 print(a[12:len(a)+1])
3 print
```

```
Ghai
Ghai
```

In [65]:
```
1 print(a[12:len(a):1])
```

```
Ghai
```

In [60]:
```
my_str = 'Python'

```

In [64]:
```
print(my_str[:6])
print(my_str[0:6])
print(my_str[0:6:1])
print(my_str[::])
print(my_str[:len(my_str)])
print(my_str[0:len(my_str):1])
```

```
Python
Python
Python
Python
Python
Python
```

P Y T H O N
0 1 2 3 4 5

In [68]:
```
a = "python"
```

In [69]:
```
a[::2]
```

Out[69]: 'pto'

In [ ]:
```
a[::2] ---> a[0:len(a):2] ----> Indexing and Concatenation

a[0::len(a):2] ---> a[0]+a[0+2]+a[2+2]
```

In [70]:
```
a[0]+a[0+2]+a[2+2]
```

Out[70]: 'pto'

In [71]:
```
a = 'Mayank Atul Ghai'

a[1:len(a):2] #a[1]+a[3]+a[5]+........ = aakAu hi
```

Out[71]: 'aakAu hi'

In [72]:
```
a[1:len(a):1] # a[1]+a[2]+......
```

Out[72]: 'ayank Atul Ghai'

In [ ]:
```
1
```

The question is an Hackerrank question the solution will be done when we will cover for loops

In [ ]:
```
Take your name
even iteration - 2, 4, 6, ......
odd iteration -  1,3,5,.....

 In odd iteration we will take the element at the end of the name and put
 In even iteration we will take last two elemnts from the end and put it

In how many iteartion are you getting the same name as you have given


Mayank
1 -----kMayan
2 ---- anKMay
3 ---- yankMa
4 -----Mayank


Girishwar
1 ----> rGirishwa
2 ----> warGirish
3 ----> hwarGiris
4 ----> ishwarGir
5 ----->rishwarGi
6 -----> Girishwar
```

In [94]:
```
a = "mayank"
a
```

Out[94]: 'mayank'

In [85]:
```
a = a[5]+a[:5]
a
```

Out[85]: 'kmayan'

In [88]:
```
a[-2::1]
```

Out[88]: 'an'

In [92]:
```
a = 'Mayank'
b = a[:5:]
```

In [93]:
```
1 b
```

Out[93]: 'Mayan'

In [ ]:
```
1 M A Y A N K
2 0 1 2 3 4 5
3
```

In [96]:
```
1 a[:6] #a[:(5+1)]
```

Out[96]: 'mayank'

In [99]:
```
for i in range(2, 100, 2):
    print(i)
```

```
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98
```

In [112]:
```
1 12<<4
```

Out[112]: 192

In [101]:
```
1 bin(12)
```

Out[101]: '0b1100'

In [111]:
```
1 0b00000000000000000000000011000000
```

Out[111]: 192

In [116]:
```
1 a = 'May ank'
2 print('' in a)
```

True

In [ ]:
```
1
```