Nested if

Nested means within or inside
Defining if statement inside if statement is called nested if (OR) if followed by another if is called nested if

Syntax:

```
if condition1:
   if condition2:
      statement-1
   else:
      statement-2
else:
   statement-3
```

if condition1,condition2 are True, python executes statement-1
if condition1 True, condition2 is False, python executes statement-2
if condition1 false, python executes statement-3

In application development nested if is used for splitting and operation

Example:

```
# Login Application

uname=input("UserName :")
password=input("Password :")

if uname=="nit":
   if password=="n123":
```

```
      print("welcome")
   else:
      print("invalid password")
else:
    print("invlaid username")
```

Output

```
UserName :nit
Password :n123
welcome

UserName :nit
Password :abc
invalid password

UserName :abc
Password :n123
invlaid username
```

Example:

```
# Write a progtam to find max 3 numbers without
# using and operator

n1=int(input("Input First Number :"))
n2=int(input("Input Second Number :"))
n3=int(input("Input Third Number :"))
if n1>n2:
    if n1>n3:
        print(n1,"is max")
    else:
        print(n3,"is max")
elif n2>n3:
```

```
    if n2>n1:
        print(n2,"is max")
    else:
        print(n1,"is max")
elif n3>n1:
    if n3>n2:
        print(n3,"is max")
    else:
        print(n2,"is max")
```

Output

```
Input First Number :10
Input Second Number :20
Input Third Number :30
30 is max

Input First Number :10
Input Second Number :30
Input Third Number :20
30 is max

Input First Number :30
Input Second Number :20
Input Third Number :10
30 is max
```

match statement

The match statement, introduced in Python 3.10, facilitates structural pattern matching. It compares a given subject value against a series of patterns defined in case blocks. When a pattern matches, the corresponding code block is executed. This is superficially similar to

switch statements in other languages, but match statements are more versatile due to their ability to match against various data structures and patterns

Match statement execute block of statements based on equality of value.

Syntax:

```
match(value):
    case pattern1:
        statement-1
    case pattern2:
        statement-2
    case pattern3:
        statement-3
    …
   case _:
        statement-x
```

if value does not match with any pattern, it execute default case which defined with _.

Example:

```
n=3
match(n):
   case 1:
      print("ONE")
   case 2:
      print("TWO")
   case 3:
      print("THREE")
```

Output
THREE

Example:

```
print("***MENU***")
print("1.Area of circle")
print("2.Area of triangle")
print("3.Area of rectangle")
print("4.Exit")
opt=int(input("Enter your option "))
match(opt):
    case 1:
        r=float(input("Enter Radious :"))
        area=3.147*r*r
        print("Area of circle ",round(area,2))
    case 2:
        b=float(input("Enter Base :"))
        h=float(input("Enter Height :"))
        area=0.5*b*h
        print("Area of triangle ",round(area,2))
    case 3:
        l=float(input("Enter Length :"))
        b=float(input("Enter Breadth :"))
        area=l*b
        print("Area of rectangle ",round(area,2))
    case 4:
        print("Thanks")
    case _:
        print("Invalid Option try aganin")
```

Output

```
***MENU***
```

1.Area of circle
2.Area of triangle
3.Area of rectangle
4.Exit
Enter your option 3
Enter Length :1.5
Enter Breadth :1.7
Area of rectangle 2.55

MENU
1.Area of circle
2.Area of triangle
3.Area of rectangle
4.Exit
Enter your option 5
Invalid Option try again

Match with | (or) operator

| is called or operator, which is used define multiple patterns.

```
match(value):
      case pattern1|pattern2|pattern3:
               statement-1

    case pattern3 | pattern4 |pattern5:
             statement-2
```

Example:

```
n=5
match(n):
   case 1|2|5:
```

```
        print("PYTHON")
    case 6|8|9:
        print("JAVA")
    case _:
        print("ORACLE")
```

Output

PYTHON

Example:

```
# Write a program to find input character is vowel or not

ch=input("Enter any character ")
match(ch):
    case 'a'|'e'|'i'|'o'|'u'|'A'|'E'|'I'|'O'|'U':
        print("Vowel")
    case _:
        print("Not Vowel")
```

Output

Enter any character A
Vowel

Match with if statement

Match with if statement allows executing block of statements based condition.

Syntax:

```
match(value):
    case pattern1 if condition:
            statement-1
```

```
    case patten2 if condition:
            statement-2
    case pattern3 if condition:
            statement-3
    case _:
            statement-4
```

```
# Write program to find input number(2,3)is even or odd

num=int(input("Enter any number "))
match(num):
   case 2 if num%2==0:
      print("EVEN")
   case 3 if num%2!=0:
      print("ODD")
```

Output

```
Enter any number 2
EVEN

Enter any number 3
ODD
```

Example:

```
# Banking Application

accno=int(input("AccountNo :"))
cname=input("CustomerName :")
balance=float(input("Balance :"))
amt=float(input("Transction Amount "))
```

```
print("1.Deposit")
print("2.Withdraw")
opt=int(input("Enter Your Option"))
match(opt):
    case 1:
        balance=balance+amt
    case 2 if amt<balance:
        balance=balance-amt


print("AccountNo :",accno)
print("CustomerName :",cname)
print("Balance :",balance)
```

Output

```
AccountNo :2
CustomerName :ramesh
Balance :45000
Transction Amount 20000
1.Deposit
2.Withdraw
Enter Your Option2
AccountNo : 2
CustomerName : ramesh
Balance : 25000.0
```

Soft keywords

Soft keywords are introduced in python 3.10 version

```
>>> import keyword
>>> keyword.softkwlist
['_', 'case', 'match', 'type']
```

Keywords are 35
Soft keyword are 4

Soft keywords can be used as identifiers
Keywords cannot use as identifiers

```
>>> pass=1
SyntaxError: invalid syntax
>>> match=100
>>> match
100
```

Membership Operator

Membership operator is binary operator and required 2 operands
Membership operator is used for searching or finding a given value into collection of values.
"in" keyword represents membership operator

1. in
2. not in

Membership operator returns boolean value (True/False)
If given value found within collection of values, it returns True else False

Syntax: value in collection-type

Operand1 can be of any type
Operand2 must be collection-type