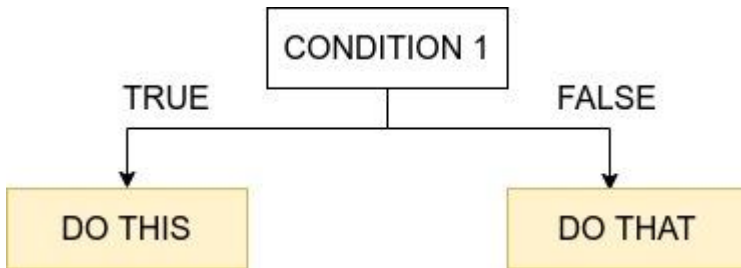# STATEMENTS

IF , ELIF , ELSE

IF , ELSE

- **Only want to execute when a particular condition has been met**
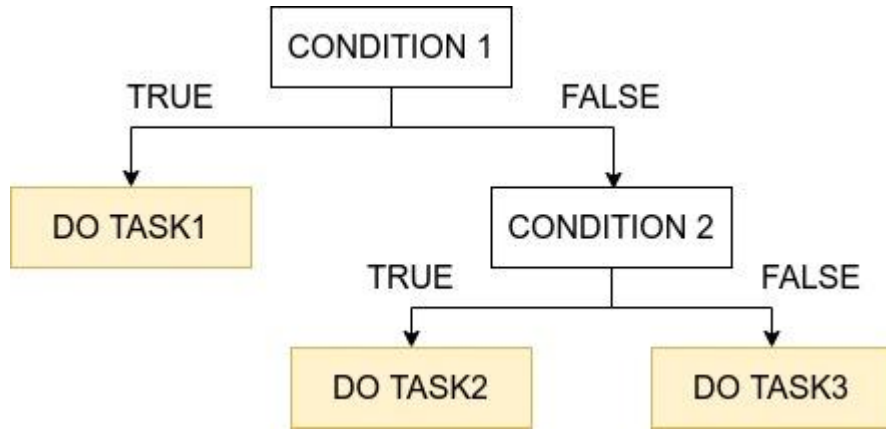

if CONDITION1 :

    DO THIS

else :

    DO THAT

If , ELIF , ELSE

if CONDITION1 :
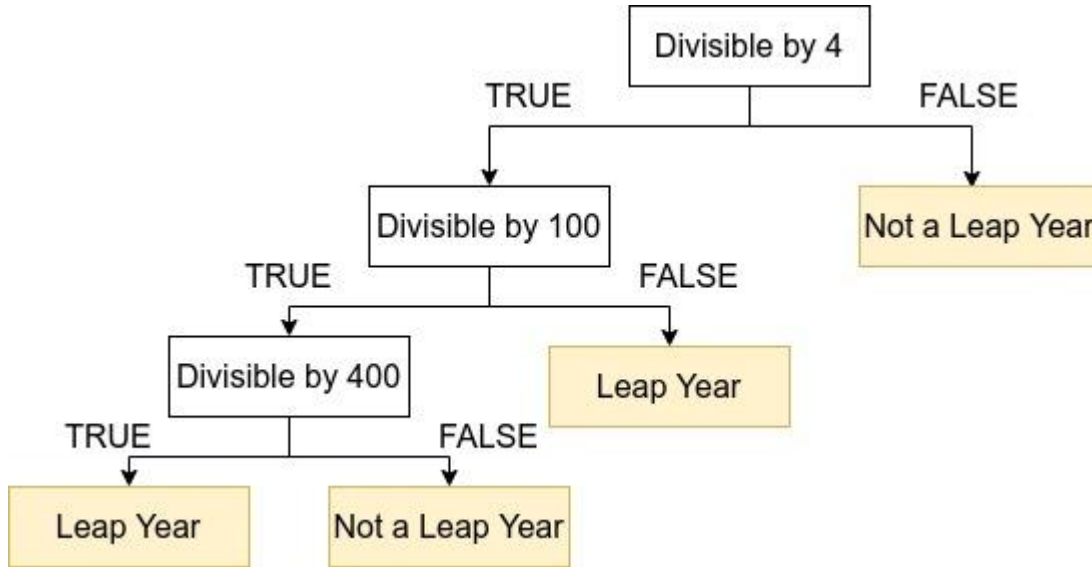
    DO TAKS1

elif CONDITION2 :

    DO TASK2

else :

    DO TASK3

NESTED IF


```
if year % 4 == 0 :

    if year % 100 == 0 :

        if year % 400 == 0:
            "leap year"
        else :
            "not a leap year"

    else :
        "leap year"

else :
    "not a leap year"
```
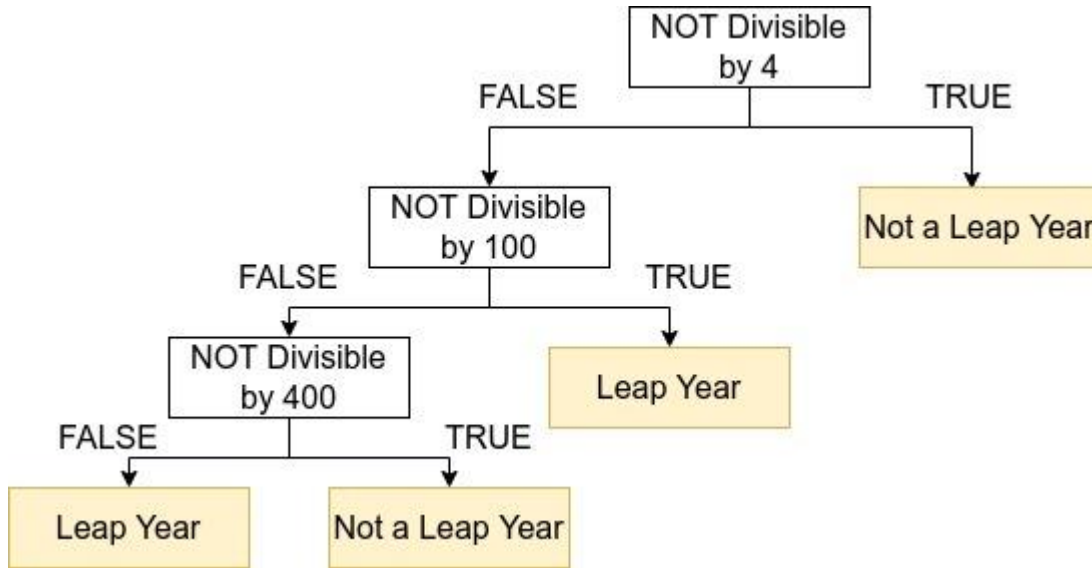
LEAP YEAR WITH IF-ELIF-ELSE

if year % 4 != 0 :

   "Not a Leap Year"

elif year % 100 != 0 :

   "Leap Year"

elif year % 400 != 0 :

   "Not a Leap Year"

else :

   "Leap Year"

FOR LOOPS

FOR LOOPS

- **Many objects in Python are "iterable" => we can iterate over every element in the object**

- **Eg – every element in a list or every character in a string, every key or every value in a dictionary**

  Syntax

  for item_name in my_iterable:

  do something to each item_name

SOME OPERATORS FOR 'FOR' LOOPS

- **range (start , stop , step)**
  - Iterable object
  - Start – start number
  - Stop – stop number (not included)
  - Step – how many jump , default = 1

SOME OPERATORS FOR 'FOR' LOOPS

for index , item in enumerate(my_iterable) :

    do something

for item1 , item2 in zip(my_iterable1 , my_iterable2) :

    do something

WHILE LOOPS

WHILE LOOPS

- **While loops will work , "while" condition is True.**

- **"while" it is rainy, stay inside.**

- **"while" having electricity, do some coding**

  Syntax

  while condition:

      do something

WHILE LOOPS

Can also use with else

Syntax

while **condition**:

    do something

else :

    do something

- Be careful for infinite loops

BREAK, CONTINUE

BREAK, CONTINUE

- **Use in loops to add additional functionality**
- **'break' – breaks out of the current closest enclosing loop**
- **'continue' – goes to the top of the closest enclosing loop**
- **'pass' – do nothing at all**