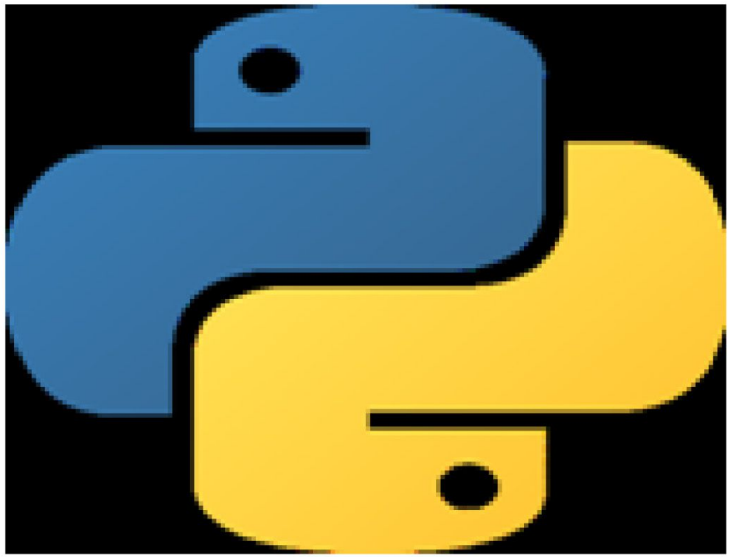


Programming With Python

Flow Control

What is Control flow?

Control flow – order in which the individual statements, instructions or function calls of an imperative program are executed or evaluated

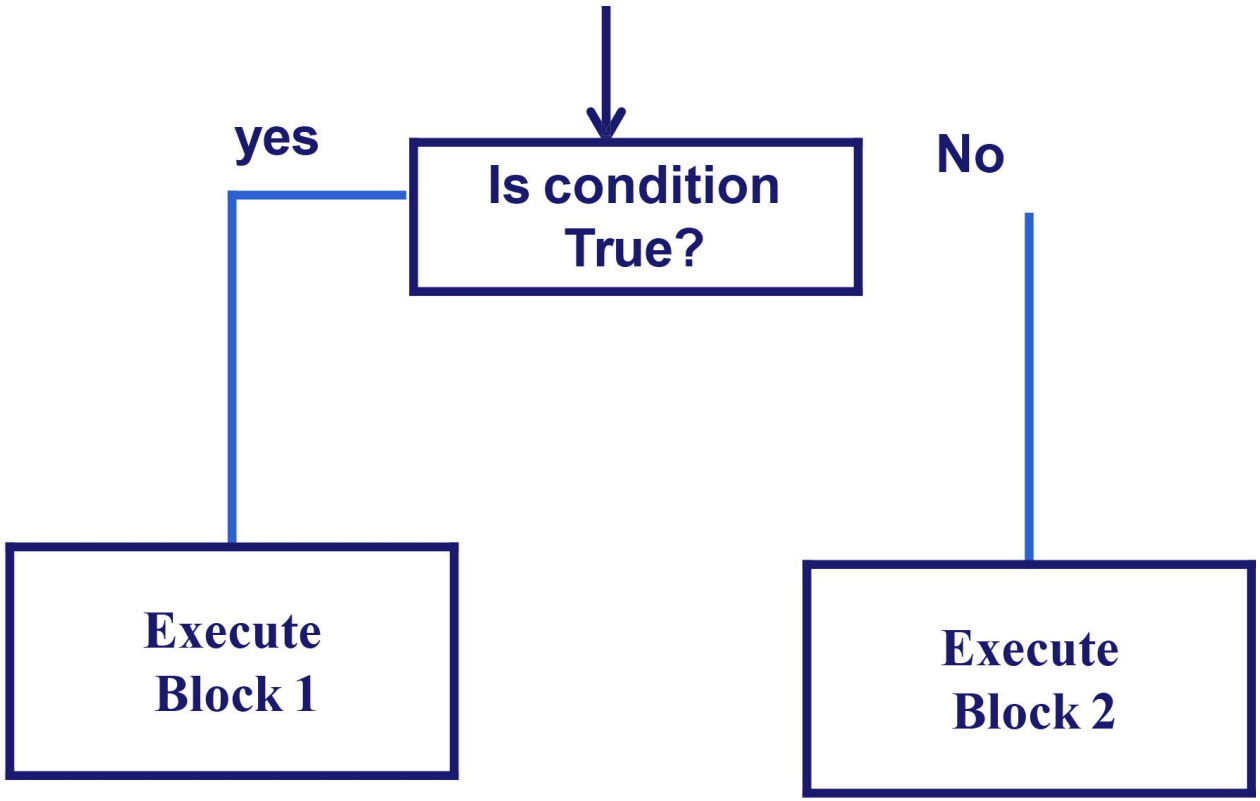
Two ways of control flow programming:

- **Conditional programming** (branching) - Boolean Expressions and Selection control (do one thing at a time (if True do this, else do that)

- **Looping** - Iterative control – For repetitive tasks

Conditional Programming (Branching)

if salary > 10 lakhs, then 30% of amount is deducted as tax; else "Tax deducted is 20% of your salary"



“if else” Statement

- **Syntax**

```
if expression:
        statement(s)
else:
        statement(s)
```

"if else" Statement…

- Boolean Expression - Contains two Boolean values – True and False

```
time = float(input("Enter Time in 24-hour format:"))

if time >= 9.30:
   Late = True
else:
   Late = False

if Late:                    # same as if bool(Late)
   print('Late entry!')    #1
else:
   print('On time!')       #2
```

Late acts as a "conditional expression", which is evaluated in a Boolean context

if…elif…else statement

```
review_score = float(input("What's the Review score:"))

    if (review_score >= 4):
       print('Must watch!')                    #1

    elif review_score >= 2.5:
       print('Can watch once!')                #2

    else:
       print('Do not Waste your precious time')     #3
```

elif clause

- a contraction for else-if
- different from a bare "**else**" **clause** in that it also has its own condition

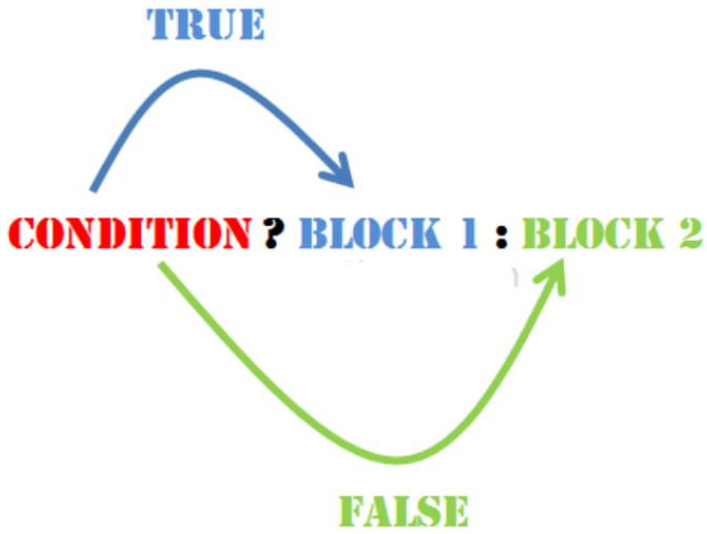
Example

Find out whether the given number is positive or Negative or Zero

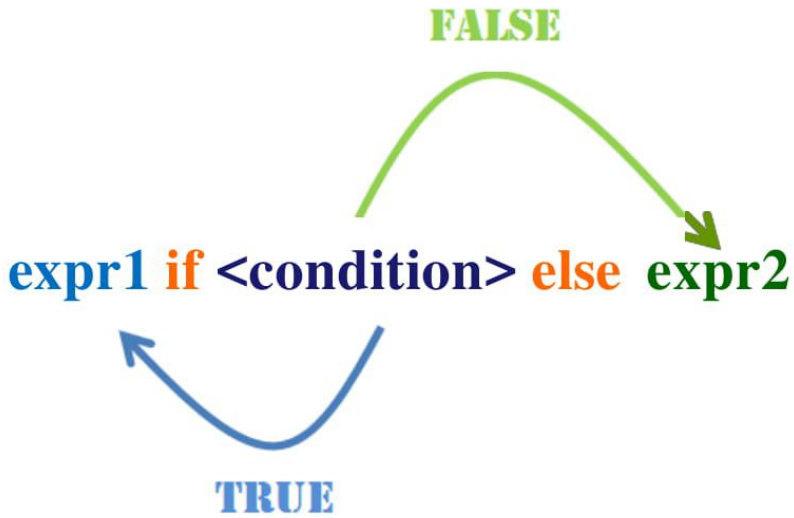
```
# Check if the number is positive or negative or zero

num = float(input("Enter a number: "))

if num > 0:
    print("Positive number")

elif num == 0:
    print("Zero")

else:
    print("Negative number")
```

Ternary Operator

Ternary Operator in C



Ternary Operator in Python




```
>>> age = 15

>>> # Conditions are evaluated from left to right

>>> print('kid' if age < 18 else 'adult')
kid
```

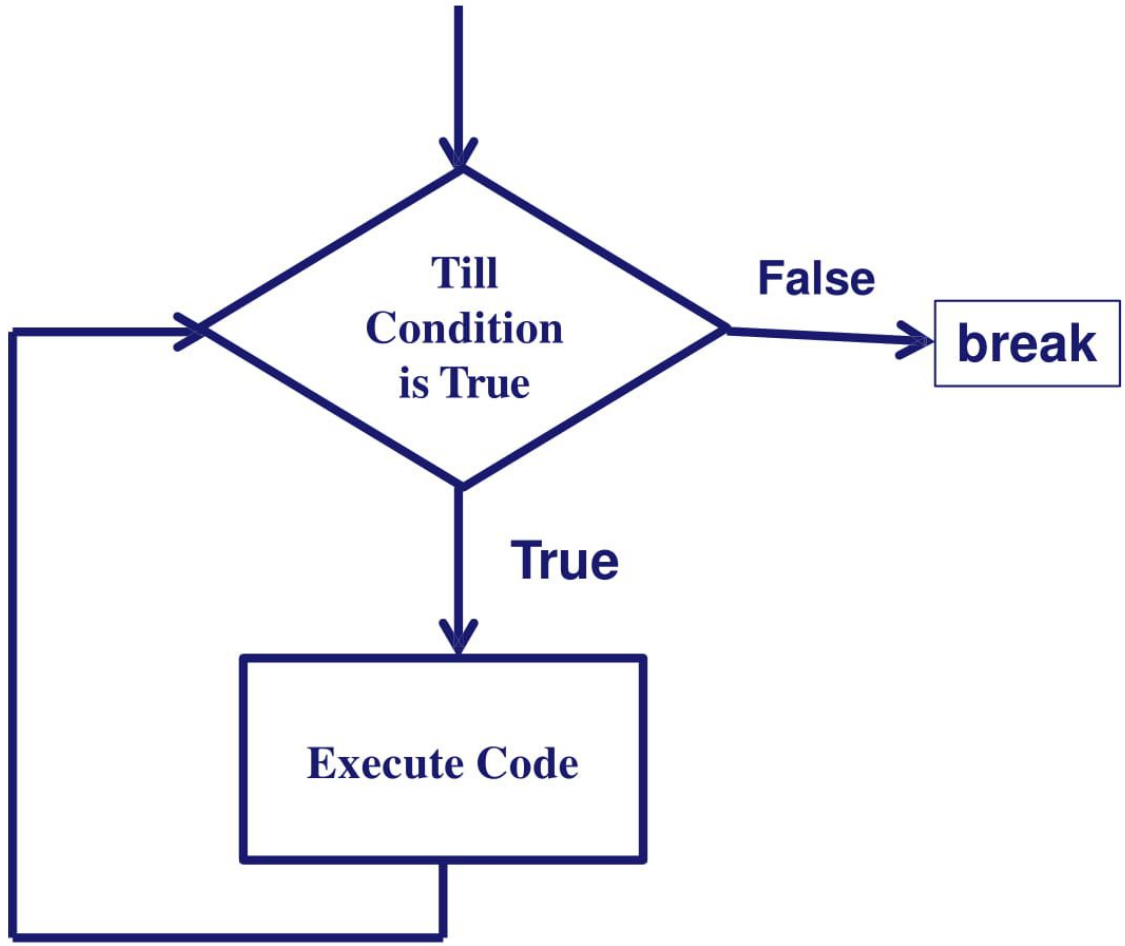
Loops

Looping- Iterative control

Looping

- Able to repeat the execution of a code block more than once, according to the loop parameters we're given.

- Python supports only two looping constructs:
  - **for**
  - **while**



While statement

The general syntax for the while statement looks like this:

```
while BOOLEAN_EXPRESSION:
    STATEMENTS
```

- The while statement is a compound statement consisting of a header and a body.
- A while loop executes an unknown number of times, as long as the BOOLEAN EXPRESSION is true.

While statement – Example I

```
# Fibonacci series: ...
# The sum of two elements defines the
next element of the series

   fib=1000
   a, b = 0, 1
   while a < fib:
      print(a)
      a, b = b, a+b
```

While statement – Example2

Find out whether the given number isArmstrong or not

A number that is equal to the sum of the cubes of its own digits. 370 = 3*3*3 + 7*7*7 + 0*0*0.

```
# Armstrong or not
num = int(input("Enter a number: "))
sum = 0
# find the sum of the cube of each digit
temp = num
while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

# display the result
if num == sum:
    print(num,"is an Armstrong number")
else:
    print(num,"is not an Armstrong number")
```

for loop

- Iterating over a range – using range function
- for loop is used when looping over a sequence - Similar to list, tuple, or a collection of objects.
- Each item in turn is (re-)assigned to the loop variable, and the body of the loop is executed.

```
for LOOP_VARIABLE in
    SEQUENCE: STATEMENTS
```

Example1: Print all the characters of a string

```
S = 'ACTS'
for ch in s:
   print(ch)
```

for – Example2

```
# Factors of a number

# take input from the user
x = int(input("Enter a number: "))

print("The factors of",x,"are:")

for i in range(1, x + 1):
    if x % i == 0:
        print(i)
```

for and while loop Comparison

- **for loop**
  - When you have to **iterate over a finite amount of elements**. It can be a huge amount, but still, something that at some point ends.

- **while loop**
  - When you just need to loop **until some condition is satisfied**, or even loop indefinitely until the application is stopped.
  - doesn't loop over a sequence

Break and Continue

Continue & Break:

- **Either skip a single iteration** (as many times you want), or you can **break out of the loop entirely**.

- On the other hand, if you're iterating over a collection of items, and you have found one of them that satisfies some need you have, you **may decide not to continue the loop entirely and therefore break out of it.**

Cancel with break

- If you want to loop until something occurs, but you're not sure when that might happen, you can use an infinite loop with a **break statement**.

Example:

```
# Capitalize a string
while True:
    word = input("String to capitalize [type
                 q to quit]: ")
    if word == "q":
        break
    print(word.capitalize())
```

Skip Ahead with continue

- Without break out of a loop
- **continue** - skip ahead to the next iteration

```
# Square only odd numbers
while True:
    value = input("Integer, please [q to quit]: ")

    if value == 'q':          # quit
        break
    number = int(value)

    if number % 2 == 0:       # an even number
        continue
    print(number, "squared is", number*number)
```

A special else clause – after while and for loops

- If while loop ended normally (no break call), control passes to an optional else.

```
numbers = [1, 5, 9, 11]
position = 0

while position < len(numbers):
    number = numbers[position]
    if number % 2 == 0:
        print('Found even number', number)
        break
    position += 1
else:# break not called  print('No
    even number found‘)
```

“switch” is missing in Python – Use if else

```
switch (day_number)
{
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:
                day = "Weekday";
                break;
        case 6:
                day =
                "Saturday";
                break;
        case 0:
                day =
                "Sunday";
                break;
                default:
        day = ""; alert(day_number
        + ' is not a valid
        day number.') }
```

```
day_number = int(input("Choose a day:
")) if 1 <= day_number <= 5:
    day = 'Weekday'
    print(day)
elif day_number == 6:
    day = 'Saturday'
    print(day)
elif day_number == 0:
    day = 'Sunday'
    print(day)
else:
    day = ''
    raise ValueError(
        str(day_number) + ' is not a valid
day number.')
~
```

Lab Exercises

7.Write a program to implement a simple calculator that can add, subtract, multiply and divide two integers. Take the required operation as choice from the user.

8. Write a simple python program to print the multiplication table of 5.

9. Write a program to check whether the given number is perfect number or not.
A number is called as a perfect number if the sum of the factors of that number is equal to the same number.
Example: 6 = 1 + 2 + 3

Thank You