Control Statement: It is a statement that determines the control of a set of instructions.

Control Structure: It is a set of instructions and the control statements that controls the execution of the instructions.

Three fundamental forms of control in programming are:

1. Sequential Control: It does not have control statement and instructions are executed in the order they are written. Program consisting of only sequential control is known as Straight Line Program.

2. Selection Control: In it set of instruction is selected on the basis of control statement. Eg. if-else statement, switch-case, etc.

3. Iterative Control: In it set of instruction is repeatedly executed on the basis of control statement. Eg. for loop, while loop, etc.

1. Boolean Expression

Boolean Data Type: It contain two boolean values: True or False (without inverted commas).

Boolean Expression: Expression that evaluates to boolean value.

2. Relational Operators

Relational Operators are the operators used for comparison.

Expression that involves relational operators are known as relational expressions.

Relational operators can be applied to any set of values that has an ordering. Eg. Numeric values, string, etc.

Relational operators are: == , != , <, >, <=, >=

```
3<5
```

```
True
```

```
5<5
```

```
False
```

```
5<=5
```

```
True
```

```
8>8
```

```
False
```

```
8>=8
```

```
True
```

```
8==8
```

```
True
```

```
8!=8
```

```
False
```

```
8!=5
```

```
True
```

Strings are compared on the basis of their ASCII code.

```
'A'<'C'
```

```
True
```

```
'Aa'<'Ab'
```

```
True
```

3. Boolean (or Logical) Operators

In Python Boolean (or Logical) Operators are: **and** , **or** ,and **not**

```
(2<3)and(3<5)
```

```
True
```

```
(2<3)or(6<5)
```

```
True
```

```
not((2<3)or(6<5))
```

```
False
```

In most of the programming languages statement like:

value1 <= var <= value2

is invalid, but python handles the above statement as:

value1<=var and var<=value2

```
2<=3<=5
```

```
True
```

```
2<8<5
```

```
False
```

4. Operator Precedence and Associativity

Operator (Precedence from top to Bottom)

1. **
2. negation (-)
3. *, /, //, %
4. Addition (+) , subtraction (-)
5. All relational Operators (==, !=,<,>,<=,>=)
6. not
7. and
8. or

All arithmetic operators are performed before any boolean or relational operators.

All relational operators are performed before any boolean operators.

```
2+3<5*2
```

```
True
```

```
2+3<5*2 and 4<3
```

```
False
```

5. Short-Circuit (Lazy) Evaluation If the statements consisting of sub-statement that involves relational operators and all the sub-statements are connected with boolean operators, subsequent sub-statements are not evaluated if the previous sub-statements is/are sufficient to determine the result.

This type of evaluation is known as In Short-Circuit (Lazy) Evaluation.

Few programming languages including Python uses Lazy evaluation.

For logical **and** if the first expression evlautes to false, then regardless of the second expression, overall expression will be False.

EG. if n < 50 and n > 5 would evaluate to false if n = 55 just by evaluating n< 50 and without evaluating n>5

```
n=55
n < 50 and n > 5
```

```
False
```

Eg: n!=0 and 1/n < 30 would evaluate result as true for n=0 without evaluating second expression. Even though 1/0 leads to error as the value will be infinity. Programmer should be cautious with this errors which will not occur for n not equal to 0.

```
n=0
n!=0 and 1/n < 30
```

```
False
```

```
n=0
1/n < 30 and n!=0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-8-ed977d0bf4af> in <module>()
      1 n=0
----> 2 1/n < 30 and n!=0

ZeroDivisionError: division by zero
```

SEARCH STACK OVERFLOW

```
0/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-3-9ab73e148374> in <module>()
----> 1 0/0

ZeroDivisionError: division by zero
```

SEARCH STACK OVERFLOW

6. if Statement

Syntax:

```
if condition:
   Set A of statements
else:
   Set B of statements
```

The amount of indentation of each program line is significant and is used to associate and group statements together.

In above: first and third line is known as **header** and Set A and Set B of statements is known as **suite**.

Group of line 1 and 2 is known as **First Clause of if statement**; whereas Group of line 3 and 4 is known as **Second Clause of if statement**.

Boolean variable can also be mentioned instead of Condition.

```
if 2<3:
  print("2 is less than 3")
else:
  print("2 is not less than 3")
```

```
2 is less than 3
```

```
if '':
  print("2 is less than 3")
  a=3;
else:
  print("2 is not less than 3")
```

```
2 is not less than 3
```

```
# Empty string will  be taken as False
if '':
  print("2 is less than 3")
else:
  print("2 is not less than 3")
```

7. if-else Statement

Syntax:

```
if condition1:
  Set A of statements
else:
    if condition2:
    Set B of statements
  else:
       if condition3:
      Set C of statements
    else:
           Set D of statements
```

```
n=18
if n<10:
  print("n is less than 10")
else:
  if n<20:
    print("n lies between 10 and 20")
  else:
    if n<30:
      print("n lies between 20 and 30")
    else:
      print("n lies above 30")
```

```
n lies between 10 and 20
```

8. elif Statement

Syntax:

```
if condition1:
  Set A of statements
elif condition2:
  Set B of statements
elif condition3:
  Set C of statements
else:
    Set D of statements
```

```
n=18
if n<10:
  print("n is less than 10")
elif n<20:
  print("n lies between 10 and 20")
elif n<30:
  print("n lies between 20 and 30")
else:
  print("n lies above 30")
```

```
n lies between 10 and 20
```

9. while Statement

Syntax:

```
while condition:
  Set of statements
```

Infinite Loop:

```
a=1
while a==1:
  b=2
```

Above program leads to an infinite loop as the value of a is not changing and hence the value of a will always be 1.

To interrupt the program execution, press CTRL + C

```
a=1
while a<5:
  print(a)
  a=a+1
```

```
1
2
3
4
```

10. Membership Operators

Python provide Membership Operators: **in** and **not in** for determining if a specific value is in (or not in) a given list of values.

```
2 in (1,2,3)
```

```
True
```

```
21 in (1,2,3)
```

```
False
```

```
21 not in (1,2,3)
```

```
True
```

Assignment:

1. Write a Python program to get the difference between a given number and 17, if the number is greater than 17 return double the absolute difference.
2. Write a Python program to test whether a number is within 100 of 1000 or 2000. (that is you need to check if the difference between 1000 and given number is less than 100 or not, similarly you need to check if the difference between 2000 and given number is less than 100 or not)
3. Write a Python program to check whether a specified value is contained in a group of values.

   Test Data :

   3 -> [1, 5, 8, 3] : True

   -1 -> [1, 5, 8, 3] : False
4. Write a program to print the number in reverse order.

Colab paid products - Cancel contracts here