Selection Control Statements

Syntax

if-statement

```
if condition:
    if_part
```

if-else statement

```
if condition:
    if_part
else:
    else_part
```

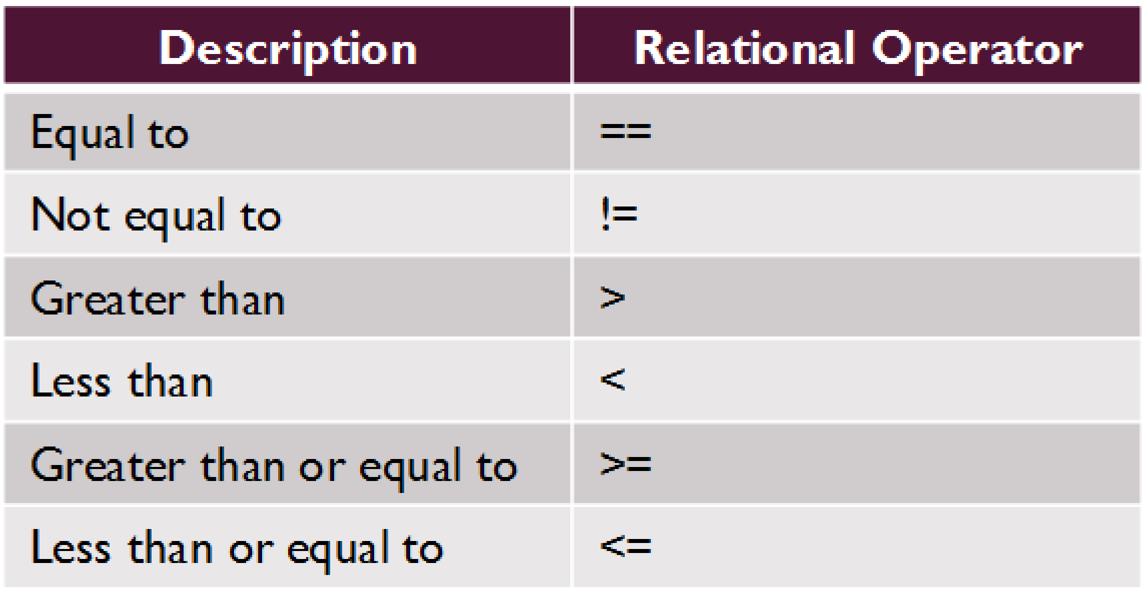
nested if statement

```
if condition1:
    if_part1
elif condition2:
    if_part2
elif condition3:
    ...
else:
    else_part
```

- We can use the relational operators to compare integer numbers, floating-point numbers, strings, and many other types.

```
if name == "Jane":
    print("Hello boss")
```

- The relational operator == is used to check if two variables have the same value.



Description	Relational Operator
Equal to	==
Not equal to	!=
Greater than	>
Less than	<
Greater than or equal to	>=
Less than or equal to	<=

- The "is" operator is used to check if two variables are the same object.

- In C, $*x == *y$ is used to check if two pointers x and y point to variables having the same value, and x == y is used to check if pointers x and y point to the same object.

In [1]:

```
young= 0
old  = 0
age  = int(input('age?'))
if age<=18:
    young += 1
    print("A young people")
else:
    old+=1
    print("An old people")
print("outside the if block")
```

```
age?20
An old people
outside the if block
```

In [2]:

```
a = [1,2,3,4]; b = [1,2,3,4]; c = a

if a == b:
    print("a and b have the same value")

if a is not b:
    print("a and b are not the same object")

if a is c:
    print("a and c are the same object")

if a == c:
    print("a and c have the same value")
```

```
a and b have the same value
a and b are not the same object
a and c are the same object
a and c have the same value
```

The bool Type

- The Boolean type in Python is denoted as bool, the true and false values are denoted by True and False, respectively.
- In Python, Boolean operators include
  - and
  - or
  - not
- Like C, Python also uses short-circuit evaluation to evaluate Boolean expressions.

```
if x!=0 and 1000//x>10:
    do_some_thing

is equivalent to

if x!=0:
    if 1000//x>10:
        do_some_thing

the former is more readable!!
```

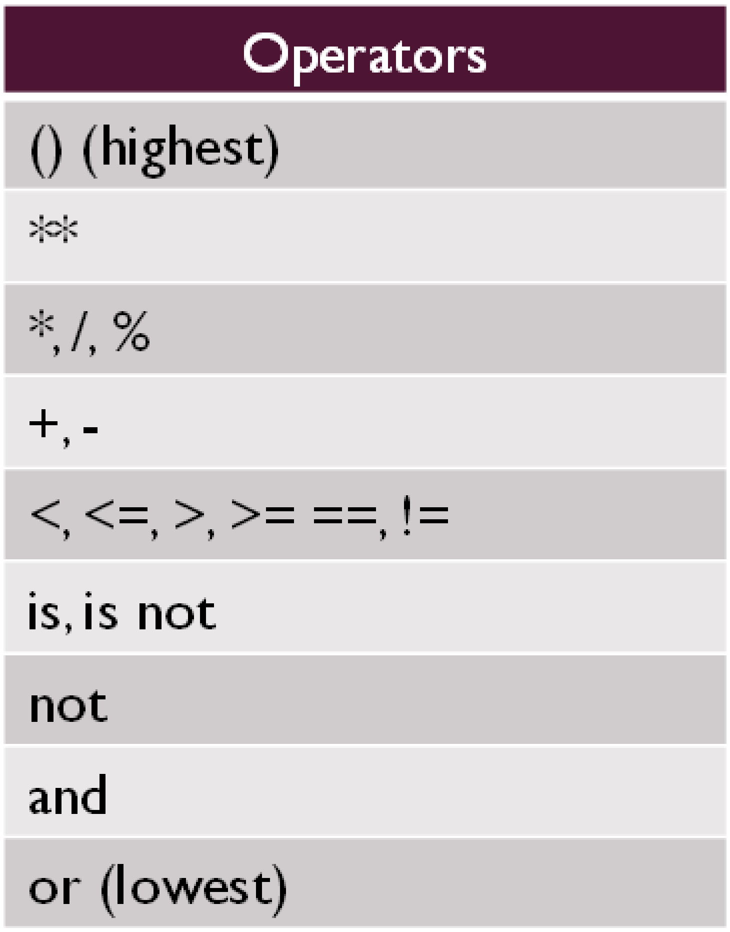
In [9]:

```python
name = "Jane"

# This is shorthand for checking if name evaluates to True:
if name:
    print("1. Hello, {}!".format(name))

# It means the same thing as this:
if bool(name) == True:
    print("2. Hello, {}!".format(name))

# name == True gets a False:
if name == True:
    print("3. Hello, {}!".format(name))
```

```
1. Hello, Jane!
2. Hello, Jane!
```

The Precedence Rules for Boolean Expressions

- Precedence rules for operators:



Operators
() (highest)
**
*, /, %
+, -
<, <=, >, >= ==, !=
is, is not
not
and
or (lowest)

- Use brackets properly to clarify expressions!!
- Use DeMorgan’s law to write more efficient and readable Boolean expressions!!

  1. NOT (a AND b) = (NOT a) OR (NOT b)
  2. NOT (a OR b) = (NOT a) AND (NOT b)
- For example, the statement

```
if age<=0 or age > 120:
    print("invalid age")
```

  is more readable than the statement

```
if not (age>0 and age <= 120):
    print("invalid age")
```

The None Value

- In Python, None is a special value which means “nothing”. Its type is NoneType and only one None value exists at a time.
- None is evaluated to False in Boolean expressions.

In [3]:

```
my_string = None
if my_string:
    print("My string is {}".format(my_string))
else:
    print("My string is None")
```

```
My string is None
```

- Check if a variable is None.

In [4]:

```
x = None
print(x is None)
print(x is not None)
print(x == None)
```

```
True
False
True
```

In [5]:

```
x = []
y = []
print(x is not None)
print(x == [])
print(x is [])
print(x == y)
print(x is y)
```

```
True
True
False
True
False
```

Switch statements and dictionary-based dispatch

- Python does not have the switch statement. In Python, we can use a technique called the dictionary-based dispatch to achieve similar results.
    - A dictionary is a collection of key-value pairs, which provides fast retrieval of values by keys.

Example 1:

```
      DEPARTMENT_NAMES = {
      "CSC": "Computer Science",
      "MAM": "Mathematics and Applied Mathematics",
      "STA": "Statistical Sciences", # Trailing commas like this are allowed in Py
thon!
      }
      if course_code in DEPARTMENT_NAMES: # this tests whether the variable is one
 of the dictionary's keys
          print("Department: {}".format(DEPARTMENT_NAMES[course_code]))
      else:
          print("Unknown course code: {}".format(course_code))
```

Example 2:

```
      def reverse(string):
          print("'{}' reversed is '{}'.".format(string, string[::-1]))
      def capitalise(string):
          print("'{}' capitalised is '{}'.".format(string, string.upper()))

      ACTIONS = {
      "r": reverse, # use the function name without brackets to refer to the funct
ion without calling it
      "c": capitalise,
      }

      my_function = ACTIONS[my_action] # now we retrieve the function
      my_function(my_string) # and now we call it
```

The Conditional Operator

Python has a way to write a selection statement in a program like the ternary conditional operator ? : in C.

Syntax

true expression if condition else false expression

Example

Mathematical formula: $y = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$

Python code 1:

```
y = x if x >= 0 else 0
```

Python code 2:

```
if x>=0:
    y=x
else:
    y=0
```

In [ ]:

```
```