Python

Decision / Logic

CONDITIONS

TYPES:

Relational Operators

Boolean Conditional

Relational Operators

Relationship (aka Condition)

First item compared to second item

What is the condition of first item when compared to second item

Both items can be variables or one can be a literal (implicit code)

Relational 1

Variables a and b

a is less than b

$a < b$

a is greater than b

$a > b$

Relational 2

Variables a and b

NOTE
Two equal signs

a equal b

```
a == b
```

a not equal b

```
a != b
```

Relational 3

Variables a and b

a is less than or equal to b

a <= b

a is greater than or equal tol b

a >= b

Relational Summary

Relation and equality operators	Description
a **<** b	a is **less-than** b
a **>** b	a is **greater-than** b
a **<=** b	a is **less-than-or-equal-to** b
a **>=** b	a is **greater-than-or-equal-to** b
a **==** b	a is **equal to** b
a **!=** b	a is **not-equal to** b

A = 5 B = 6 C = 7 D = 2 E = 6

A > B	False
C > D	True
D == A	False
C <= B	False
B == E	True
D != B	True
C == 7	True
E => 6	True

DATA TYPE Conditions

Conditions can only compare variables of the same data type.

String to String

Number to Number

A =2 B = 5 C = “5” D = ‘A’ E = A

A > B	False
C > D	False
D == A	False
C <= B	ERROR
B == E	FALSE
D != B	True
C == 7	False
E >= 6	False

log1

Boolean

Boolean

Compare two or more conditions

Boolean

AND

Conditions	result
true and true	true
true and false	false
false and true	false
false and false	false

Boolean

OR

Conditions	result
true or true	true
true or false	true
false or true	true
false or false	false

Boolean

NOT

Opposite of the tested condition

Example:

amount = 100

not (amount =100)

A = 5 B = 6 C = 7 D = 2 E = 6

(A == B) and (A ==C)	FALSE
(A > B) and (B > C) and (C > D)	FALSE
(B == E) and (C > E)	TRUE
(A != E) and (C == 7)	TRUE
(B > E) or (C > A)	TRUE
(A == E) or (B == D) or (E == B)	TRUE
(A!= D) or (B!= E) or (B ==5)	TRUE
(D == 2) or (C == 7) or (B == E)	TRUE
not (E==6)	FALSE

log2

MORE

- Parenthesis are allowed around conditions.
- Parenthesis are allowed inside of boolean statements to clarify an evaluation.

(condition) OR (condition)

Order of Execution
Precedence rules

Operator/Convention	Description	Explanation
( )	Items within parentheses are evaluated first	In `(a * (b + c)) - d`, the + is evaluated first, then *, then -.
not	Logical not is next	`not(x == 1) or y` is evaluated as `(not(x == 1)) or y`
* / % + -	Arithmetic operators (using their precedence rules; see earlier section)	`z - 45 * y < 53` evaluates * first, then -, then <.
`< <= > >=`	Relational operators	`x < 2 or x >= 10` is evaluated as `(x < 2) or (x >= 10)` because < and >= have precedence over the or operator.
`== !=`	Equality and inequality operators	`x == 0 and x >= 10` is evaluated as `(x == 0) and (x >= 10)` because == and >= have precedence over the and operator.
and	Logical and	`x == 5 or y == 10 and z != 10` is evaluated as `(x == 5) or ((y == 10) and (z != 10))` because the and operator has precedence over the or operator.
or	Logical OR	

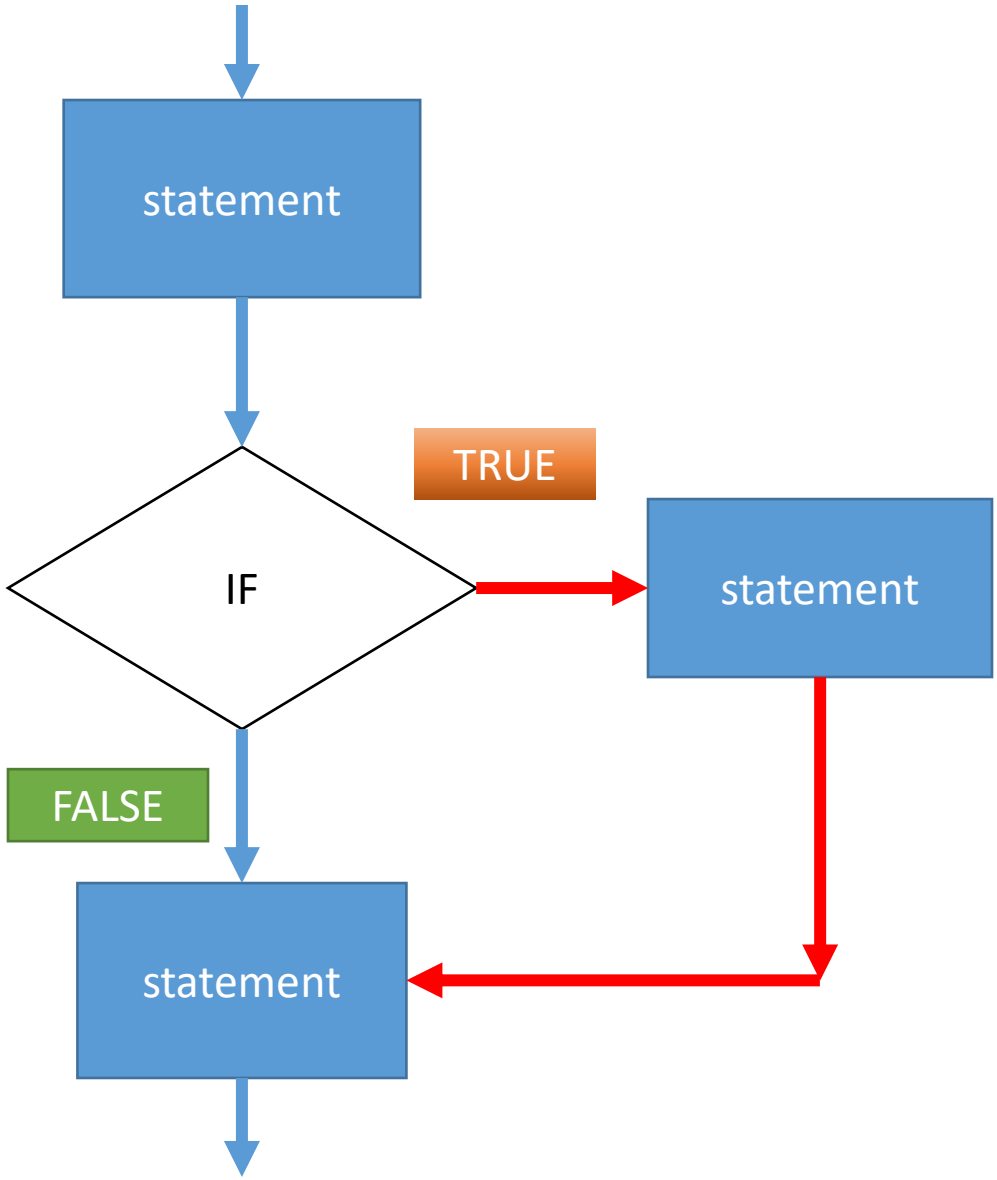
Be careful

Try to keep calculations and function usage out of the conditions and Boolean line

```
x == (y+1)

y = y+1
X == y
```

IF statement


statement
TRUE
IF
statement
FALSE
statement

If statement – simple form

```
if (condition):
    statements
```

Note: colon

Note: Indent

Statements executed if condition is TRUE

If simple - example

```
if amount > 0:
    payment = amount * .25
```

NOTE

```
def main():
    a = 5
    b = 3
    c = a + b
    if (c > 5):
        print('C is greater than 5')
    print('---------')
main()
```

condition

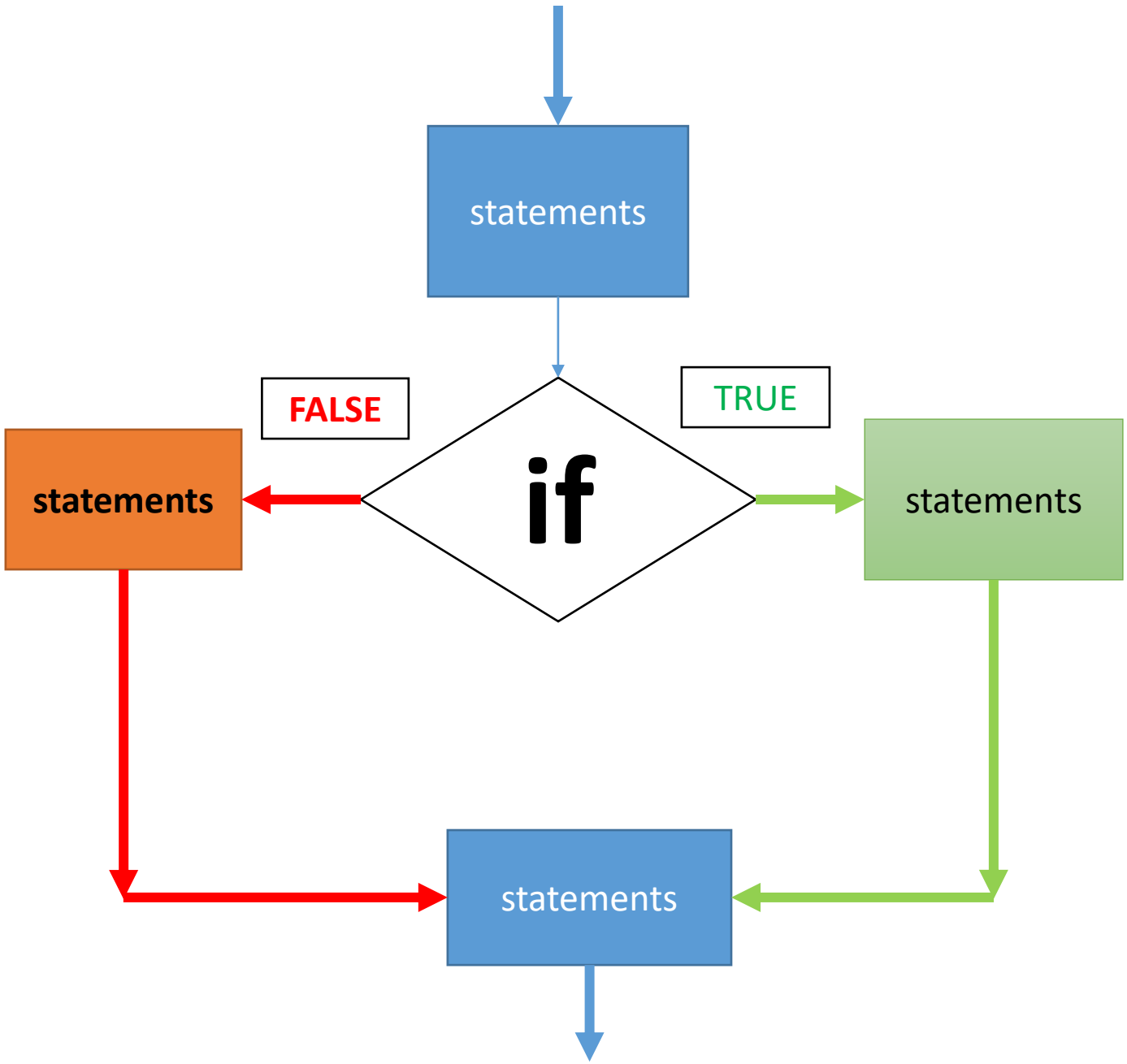
TRUE

Try this

```
def main():
    a = 5
    b = 3
    c = a + b
    if (c > 5):
        print('C is greater than 5')
    print('---------')
main()
```

Try This

```
def main():
  w = ''
  w = input('enter a letter --> ')
  if (w == 'b'):
     print('w contains a b')
main()
```


statements
FALSE
TRUE
if
statements
statements
statements

If true/false form

```
if (condition):
    statements
else:
    statements
```

TRUE

FALSE

NOTE the COLON

If true/false example

```
if amount > 0:
    payment = amount * .25
else:
    print(‘ No payment required’)
```

If with Boolean

```
if (a > b) and (c==d):
    statements
else:
    statements
```

NOTE

```
def main():
    a=2
    m=3
    b=int(input('enter a number -> '))
    k=int(input('enter a number -> '))
    if (a > b) and (m==k):
        print(' --TRUE --')
    else:
        print(' false ‘)
    print(‘—done—’)
main()
```

condition

Always executed

Try this

```
def main():
    a=2
    m=3
    b=int(input('enter a number -> '))
    k=int(input('enter a number -> '))
    if (a > b) and (m==k):
        print(' --TRUE --')
    else:
        print(' false ')
main()
```

Nested if

```
if (condition):
    statements
else:
    if (conditions)
        statements
    else:
        statements
```

If and the else must line up and watch the indent

Try this

```
def main():
    m=int(input('enter a number -> '))
    if(m>=10):
        print('Greater than 10')
    else:
        if (m>=5):
            print('greater than 5')
        else:
            print('less than 5')
main()
```

Alternative to nested if

Use elif

Similar to case statement In other programming languages

IF / elif
else-if

```
if (condition1):
   statement(s)
elif (condition2):
   statement(s)
elif (condition3):
   statement(s)
else:
   statement(s)
```

MUST HAVE

IF / elif (else-if) example

```
If (level == 1):
    level1()
elif (level==2):
    level2()
elif (level==3):
    level3()
else:
    levele()
```

Similar to case statements

Try this

```
def main():
    x = int(input("enter an integer: "))
    if x < 0:
        x = 0
        print('Negative changed to zero')
    elif x == 0:
        print('Zero')
    else:
        print('positive number')
main()
```

if05

Try this

```
def main():
    m=int(input('enter a number -> '))
    if(m>=10):
        print('Greater than 10')
    elif  (m>=5):
        print('greater than 5')
    else:
        print('less than 5')
main()
```

Nested vs elif

```
def main():
    m=int(input('enter a number -> '))
    if(m>=10):
        print('Great-r than 10')
    else:
        if (m>=5):
            print('greater than 5')
        else:
            print('less than 5')
main()
```

```
def main():
    m=int(input('enter a number -> '))
    if(m>=10):
        print('Greater than 10')
    elif  (m>=5):
        print('greater than 5')
    else:
        print('less than 5')
main()
```

Notes

A condition always has 2 parts, a left side and a right side.

When comparing a variable with multiple condition, each condition is a pair, that the variable has to be repeated.

Example

CANNOT

(pymt > 0 or < 100)

SHOULD……

(pymt > 0) or (pymt < 100)

Note: variable is stated twice

More examples

if07

DONE