Control Flow

conditions

*all conditions are Boolean expressions that must evaluate to True or False
*Boolean expressions can be combined using: and, or, not

examples:
< less than
> greater than
!= not equal to
<= less than or equal
>= greater than or equal

statements

*any valid Python code can make up the statement block of a loop or if-else statement
*loops and if-else statements can be nested
(ex. the statement block of a while loop can include another while loop)

examples:
function calls, print statements, assignment statements, for/while loops, if-else statements

if statement structure

if statement

if *condition*:
  statement(s)

*will only execute if condition evaluates to True

if-else statement

if *condition*:
  statement(s)
else:
  statement(s)

*executes initial statement block if condition is True, evaluates second statement block if False

elif statement

if *condition:*
  statement(s)
elif condition:
  statement(s)
else:
  statement(s)

*allows for one step to check for more than one condition

control flow example

```
if color == “green”:
  print(“GO”)
elif color == “yellow”:
  print(“SLOW”)
else:
  print(“STOP”)
```

color = “green” → “GO”
color = “yellow” → “SLOW”
color = “red” → “STOP”

while loops

*very similar to for loops; any for loop can be rewritten as a while loop

*allow for a loop method to be used without knowing how many times to loop in advance

*the condition for a while loop is far less limited than the parameters needed for a for loop

while loop structure

initialization (so condition is true)
while *condition*:
  statement(s)
  advance (so loop continues)

```
limit = 10
i = 1
while i <= limit:
  print(i)
  i += 1
```
→ 1 2 3 4 5 6 7 8 9 10

infinite loop

*happens when your while loop gets stuck because the condition never becomes false

*to get out of an infinite loop, press Ctrl + C

*to debug, alter the advance statement so that the loop executes appropriately