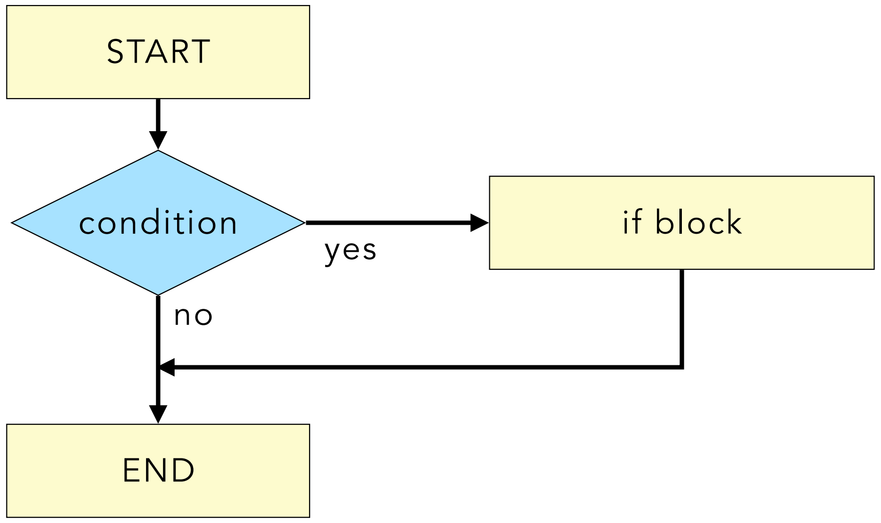
Chapter 6. if Statements

```python
import numpy as np

me = 9.11e-31      # mass of electron
c  = 299792458     # speed of light

u  = 0.1 * c       # particle velocity

gamma = 1 / np.sqrt(1-(u/c)**2)   # gamma factor

KE = (gamma-1) * me * c**2        # relativistic kinetic energy
```

Python for Physicists

if statements: control flow of a program based on conditions

Flow control is most easily visualized with a flow chart



Pseudocode:

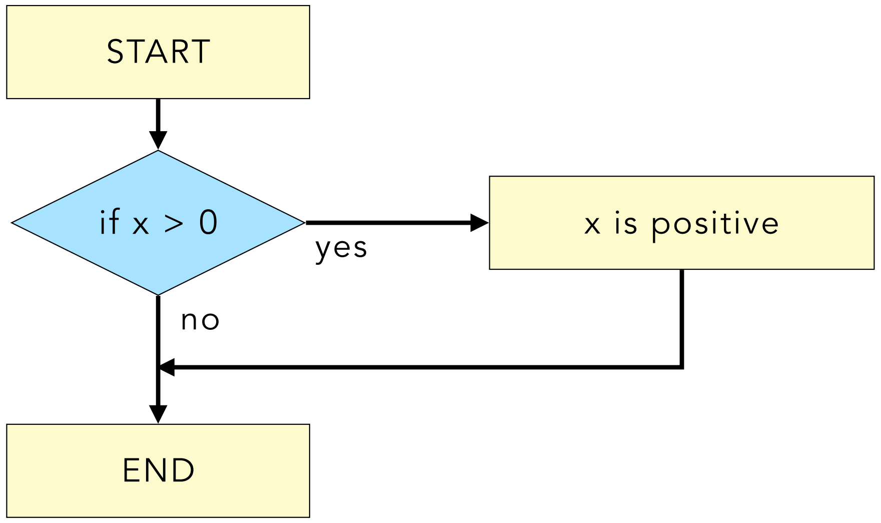
```
if condition:
    code
```

if statements: control flow of a program based on conditions

Example

Python code:

```
if x > 0:
    print(x,'is positive')
```

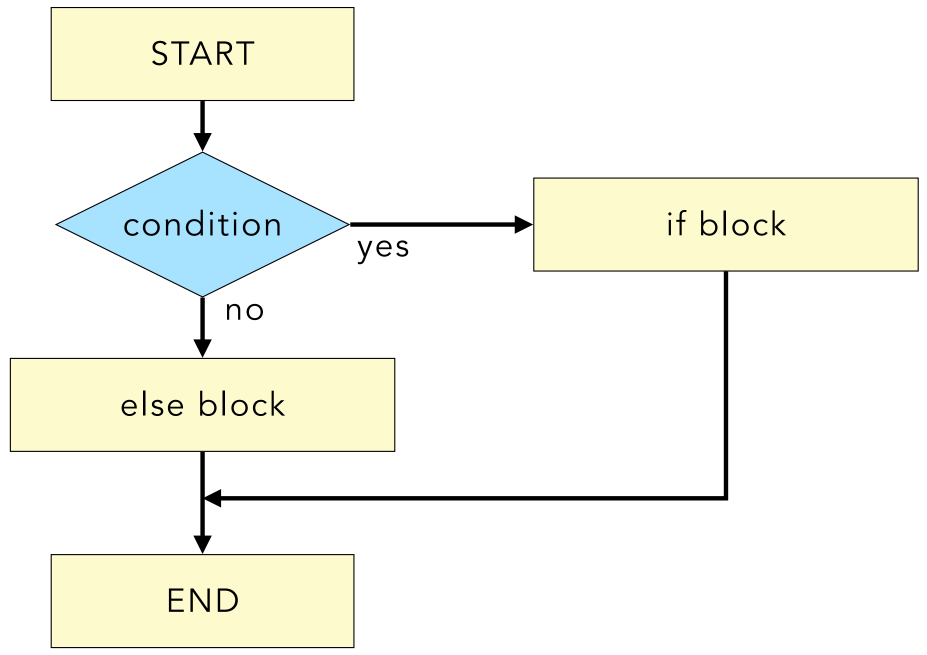


if statements: control flow of a program based on conditions

The **else** block is executed when the if condition is false

Pseudocode:

```
if condition:
    code
else:
     code
```

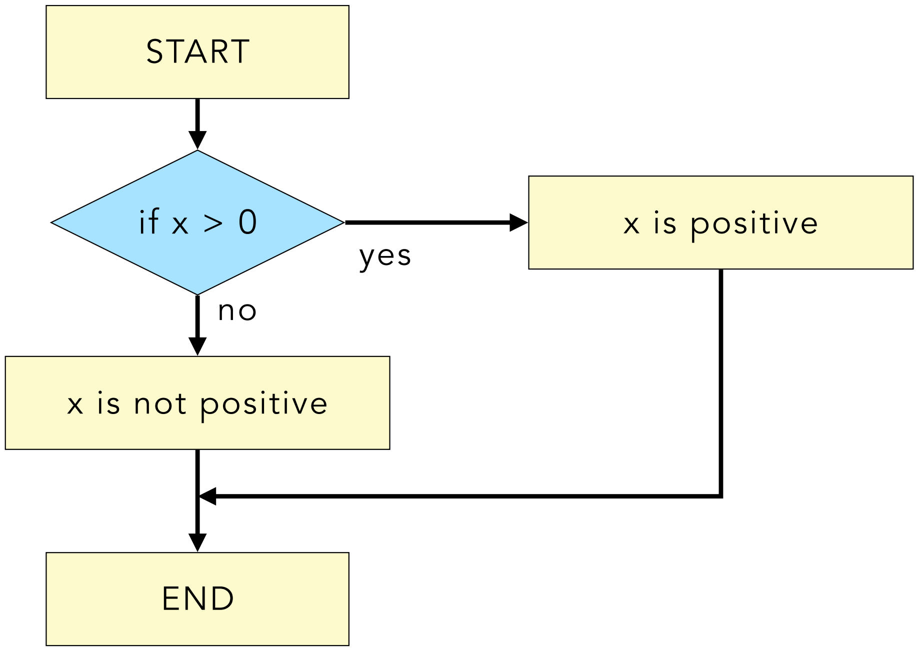


if statements: control flow of a program based on conditions

Example

Python code:

```
x = float(input("Enter a number:  "))

if x > 0:
    print(temp,'is positive’)
else:
    print(temp,'is not positive')
```

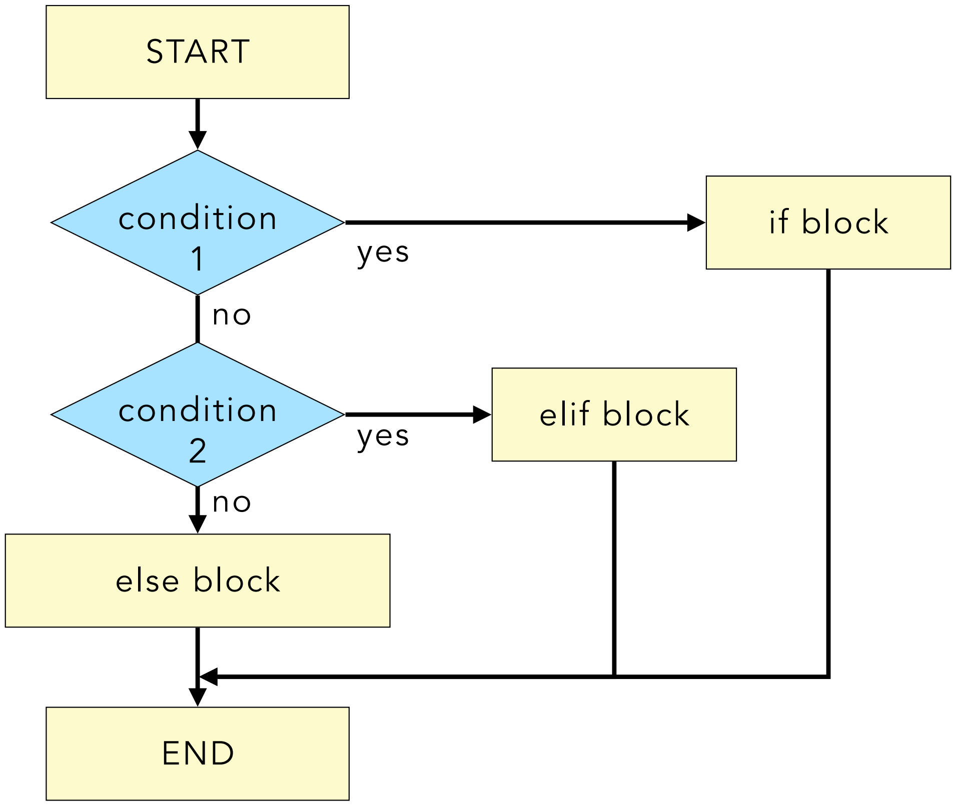


if statements: control flow of a program based on conditions

An **elif** statement can evaluate a second condition if the first condition is false.

Pseudocode:

```
if condition 1:
    code
elif condition 2:
     code
else:
     code
```



Comparison (i.e. relational) operators

operator	meaning	example	
`==`	Equality operator	`x == y`	True if x equal y
`!=`	Not equal	`x != y`	True if x is not equal to y
`>`	Greater than	`x > y`	True if x is greater than y
`<`	Less than	`x < y`	True if x is less than y
`>=`	Greater than or equal to	`x >= y`	True if x is greater than or equal to y
`<=`	Less than or equal to	`x <= y`	True if x is less than or equal to y
`is`	same object (identity)	`a is b`	True if a and b are the same object (not just numerically equal)
`in`	membership	`a in b`	True if a is in b

Boolean (logical) operators

operator	example	meaning
`and`	`x and y`	True if BOTH x and y are True
`or`	`x or y`	True if EITHER x or y are True
`not`	`not x`	True if x is False

Result of Comparison and Boolean Operators is Boolean data type

`x = 4 > 2`	x will be a Boolean data type = True
`y = 1 == 2`	y will be a Boolean data type = False

Examples

```
A = [20, 10, 20, 30]
```

Expression	Boolean Result
`A[0] > A[1]`	`True`
`A[0] > A[1] and A[0] < A[3]`	`True`
`A[0] > A[3]`	`False`
`A[0] == A[2]`	`True`
`30 in A`	`True`
`not(A[0] < A[3])`	`False`

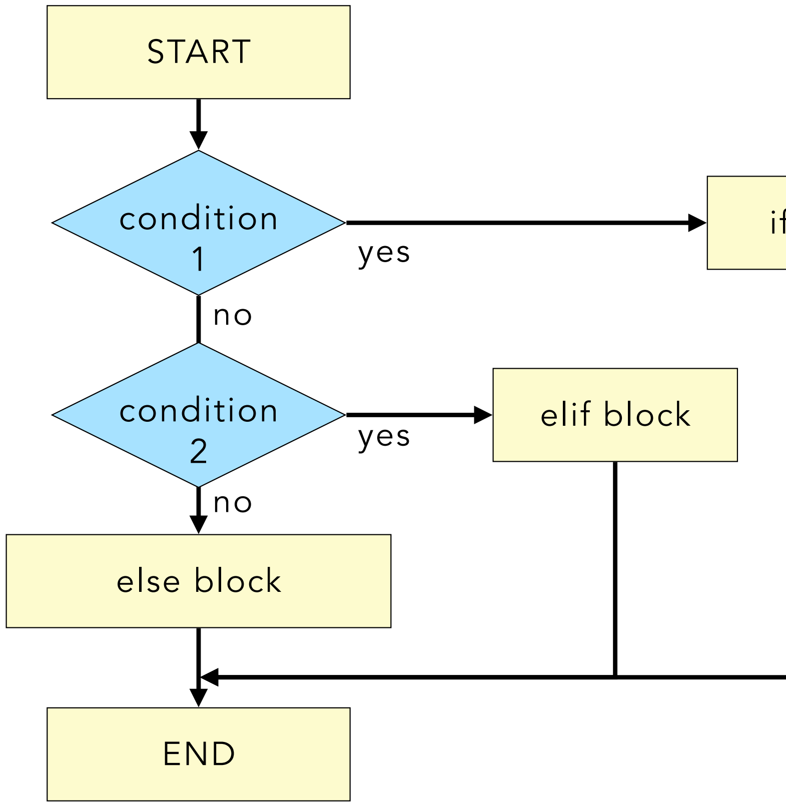
Group Exercise

1. Draw a flow chart to categorize someone's age. Use the following categories to print a message. Hint: you can have as many `elif` statements as you like in an if statement.

   - kid: age < 11
   - tween: 11 ≤ age < 13
   - teen: 13 ≤ age < 20
   - adult: 20 ≤ age

2. Write Python code to do the following:

   - prompt user to enter an age
   - print a message based on their age



Coding Patters

Coding Patterns are commonly-used combinations of loops, if statements, counters, etc. to achieve a particular result. We will discuss the following four examples:

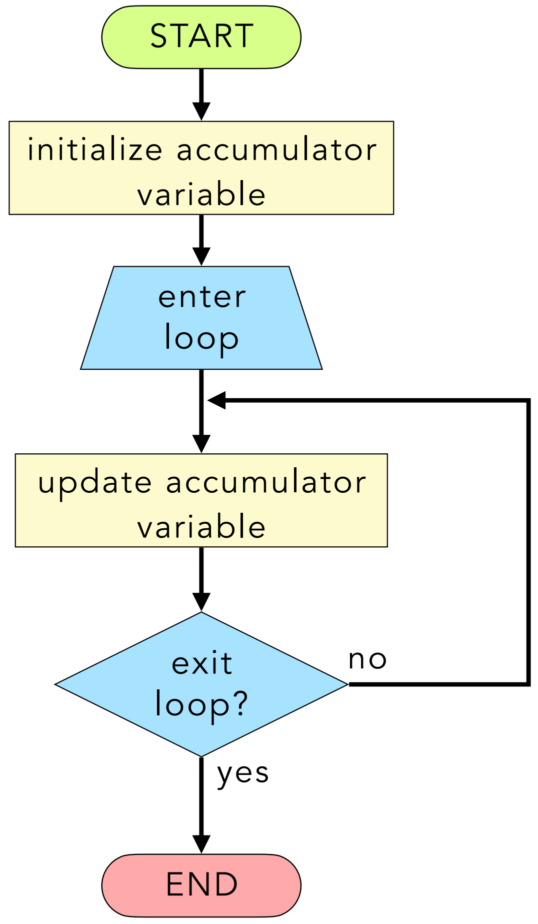
- Accumulator Pattern
- Update (or Replacement) Pattern
- Count Pattern
- Search Pattern

Accumulator Pattern

- The Accumulator Pattern consists of a loop and an accumulator variable.
- On each iteration of the loop, the accumulator variable "accumulates" or "gathers" information.

Uses:

- Summing
- Computing factorial
- Repeatedly extending a list with new elements
- Numerical integration



Accumulator Pattern

Example: Summing integers

Python code:

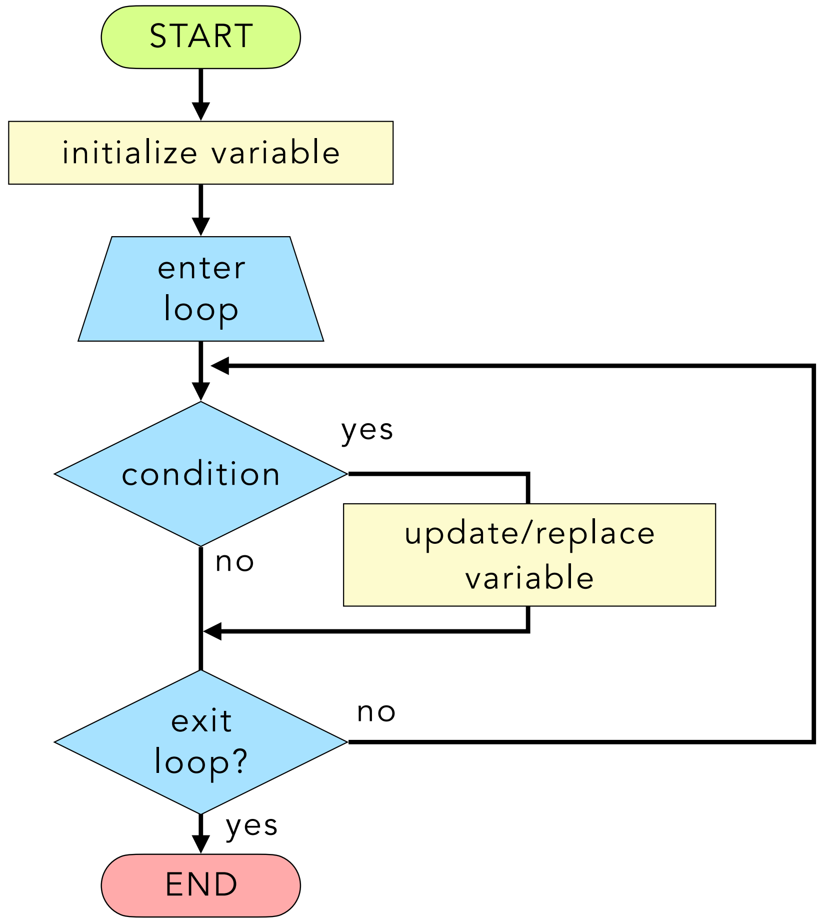
```python
N = 10                            # N = upper limit of sum

total = 0                         # total = accumulator = sum of integers
for i in range(1,N+1):            # loop over integers 1 to N
    total = total + i             # add i to the running total

print("sum of integers from 1 to",N,"is",total)
```

Update (or Replacement) Pattern

- The Update/Replacement Pattern consists of a loop and a variable to be updated/replaced.
- On each iteration of the loop, the variable is replaced with new information when a condition is met.

Uses:

- Calculating min or max of an array
- Detecting events in an array, such as exceeding a threshold



Update (or Replacement) Pattern

Example: Finding maximum value in a list

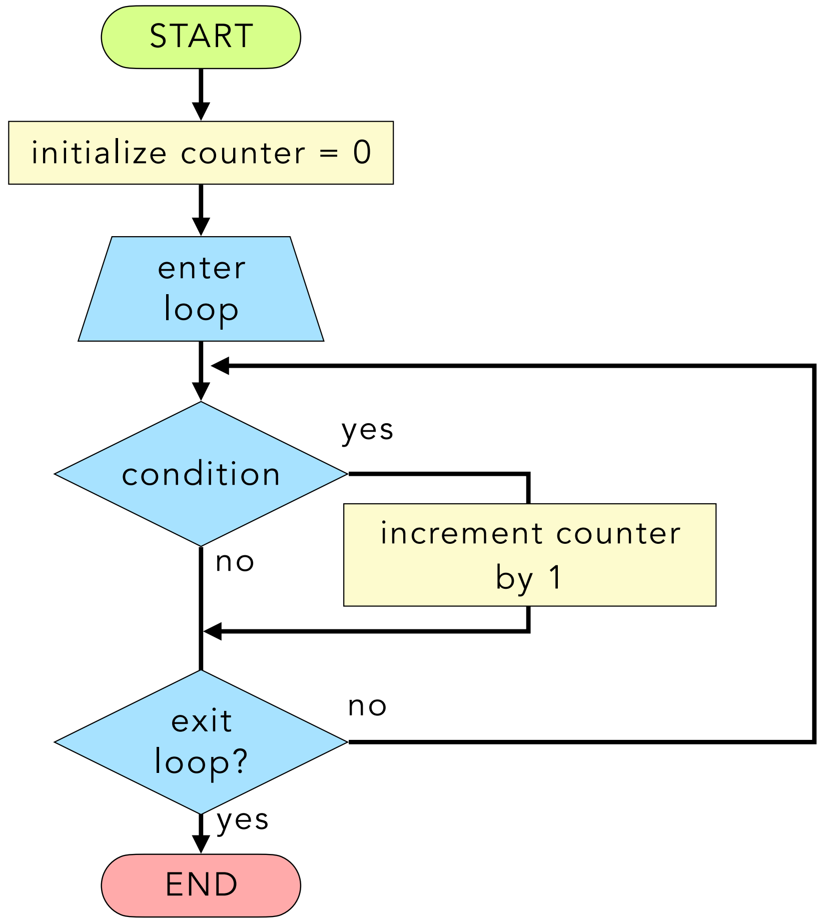
```
vlist = [3, 7, 27, -2, 12]     # define a list of numbers

max_v = vlist[0]               # initialize max_value to first element in list

for v in vlist:                # loop over numbers in the list
    if v > max_v:              # check if the current number > max_val
        max_v = v              # if true, update max_val to current number

print("max value = ", max_v)  # print out the max value in the list
```

Count Pattern

- This pattern is used to count occurances
- A counter variable is initialized to 0
- Loop over an array and increment the counter if some condition is met.

Uses:

- Counting
- Creating histograms



Update (or Replacement) Pattern

Example: Find number of list elements with values >= threshold value

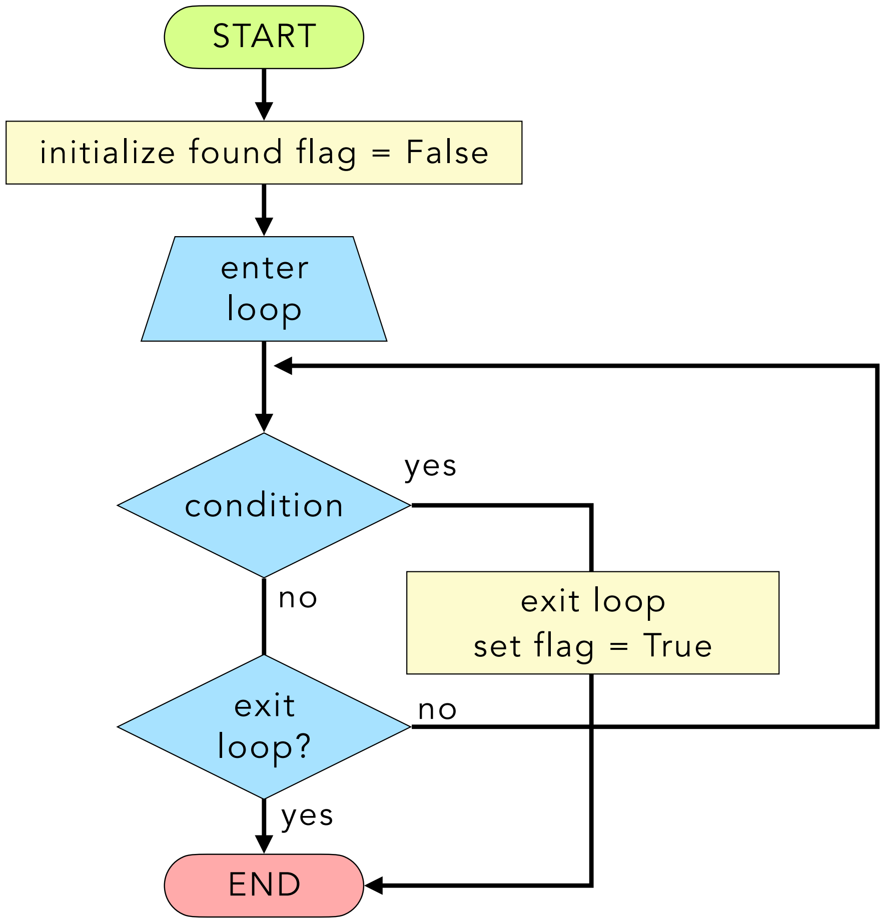
```
n_cats = [0,1,3,4,10]        # number of cats owned

count = 0                    # initialize counter
threshold = 3                # threshold for detection

for cats in n_cats:         # loop over numbers in the list
    if cats >= threshold:   # check if value > threshold
        count += 1          # if true, increment counter

print(count," people in the list own at least",threshold,"cats")
```

Search Pattern

- This pattern searches for a value or pattern in a list.
- If the value is found, a flag is set to true and the loop is exited to save computer resources

Uses:

- Counting
- Creating histograms


START
initialize found flag = False
enter loop
condition
yes
no
exit loop set flag = True
exit loop?
no
yes
END