USER INPUTS — most code is written to solve a type of problem
- you often need data from a user
  eg. Age
  # of days
- But your user doesn't want to have to run your code!

input() → where the data string is saved; PROMPT string the user will see

```
name = input('Tell me your name:')
print(f'Hi {name}, you are cool!')
```

The prompt should be clear and easy to follow.

```
prompt = "Tell me your name and I can add you to the list"
promp += "\n Type in your name: "
name = input(prompt)
```

+= is a really fast way to add to strings or integers or floats.

Input assumes everything is a string

```
age = input('Enter your age')
print(age)
```

say you write: 21
'21'

```
age = int(age)
```
-or-
```
age = float(age)
```

This is where you have to change the data type.

The Modulo Operator – divides and returns the remainder

4 % 3

$$\begin{array}{r} 1 \\ 3\overline{)\,4} \\ -3 \\ \hline 1 \end{array}$$

remainder = 1

Think of a clock.

12am — same as — 0
1am — same as 1
⋮
12 pm — same as 12
1 pm — same as 13

13 % 12 = 1

Can I divide a class of people into groups of 3?

```
num_people = input('Enter Number of people in class:')
num_people = int(num_people)

if num_people % 3 == 0
        Print('Yes you can do groups of 3!')
```

1

While Loops:

Continues doing the loop until a condition is met.

```
# I want 3 hobbies for each user:

hobbies = [ ]
while len(hobbies) < 3
    hob = input('Enter a hobby: ')
    hobbies.append(hob)
```

```
# I want to let the user choose when to quit.
print('Name as many colors as you can!')
prompt = '\n Enter a new color: '
prompt += " If you can't think of one type 'quit'."
colors = [ ]
message = ''
while message != quit:
    message = input(prompt)
    if message != quit:
        if message not in colors:
            colors.append(message)
        else: print(colors)
            print('You already guessed that one')
```

Flags

a flag is a boolean statement

```
active = True
while active:
    :
```

Breaks

you can use the word break in any kind of loop to end it immediately. don't do any more commands.

Continue

keep going - looping - but don't do the rest of commands for that run.

DANGER! Infinite loops ... make sure your loop will end.

```
x = 1
while x < 5
    print(x)
```

} this will run forever because x is always < 5

```
x = 1
while x < 5
    print(x)
    x += 1
```

Using while loops with other data types:

```
my_list = [5, 4, 3, 2, 1]

while my_list:
    print(my_list)
    my_list.pop()
```

→ go until my_list is empty.

```
my_list = [5, 4, 5, 3, 5, 2, 1]

while 5 in my_list:
    my_list.remove(5)
```

You can also do operations w/ Dictionaries.