DATA 520
Lecture 11
Using Loops

Doing things over and over

for, while

break and continue

Loops

Loops repeat things any number of times

Life without Loops:

```
# 4 different speeds to be shown from a list and converted
velocities = [0.0, 9.81, 19.62, 29.43]
print('Metric:', velocities[0], 'm/sec;  Imperial:', velocities[0] * 3.28, 'ft/sec')
print('Metric:', velocities[1], 'm/sec;  Imperial:', velocities[1] * 3.28, 'ft/sec')
print('Metric:', velocities[2], 'm/sec;  Imperial:', velocities[2] * 3.28, 'ft/sec')
print('Metric:', velocities[3], 'm/sec;  Imperial:', velocities[3] * 3.28, 'ft/sec')
```

Life with Loops:

```
for velocity in velocities:   # note: velocity not defined
    print('Metric:', velocity, 'm/sec; Imperial:', velocity * 3.28, 'ft/sec')
```

for *«each item»* **in** *«list»*:

    «code block»

Loops

for loops start with the first item and execute code through to the last item

```
for «each item» in «list»:
    «code block»
# add one more that Ralph did
velocities = [0.0, 9.81, 19.62, 29.43, 'Ralph']
for velocity in velocities:    # note: velocity not defined before, now defined on the fly
    print('Metric:', velocity, 'm/sec;',
    'Imperial:', velocity * 3.28, 'ft/sec')
Metric: 0.0 m/sec; Imperial: 0.0 ft/sec
Metric: 9.81 m/sec; Imperial: 32.1768 ft/sec
Metric: 19.62 m/sec; Imperial: 64.3536 ft/sec
Metric: 29.43 m/sec; Imperial: 96.5304 ft/sec
Traceback (most recent call last):
  File "<pyshell#6>", line 3, in <module>
    'Imperial:', velocity * 3.28, 'ft/sec')
TypeError: can't multiply sequence by non-int of type 'float'
```

Loops

for loops start with the first item and execute code through to the last item

```
# velocities = [0.0, 9.81, 19.62, 29.43, 'Ralph'] # now 5 items
for velocity in velocities:
    print(str(velocity) + ' out of ' + str(velocities) ) # must convert list to string
```

```
0.0 out of [0.0, 9.81, 19.62, 29.43, 'Ralph']
9.81 out of [0.0, 9.81, 19.62, 29.43, 'Ralph']
19.62 out of [0.0, 9.81, 19.62, 29.43, 'Ralph']
29.43 out of [0.0, 9.81, 19.62, 29.43, 'Ralph']
Ralph out of [0.0, 9.81, 19.62, 29.43, 'Ralph']
```

```
# open a new file and paste in
velocities = [0.0, 9.81, 19.62, 29.43, 'Ralph']
loopcount = 1
velolen = len(velocities)
for velocity in velocities:
    print('step ' + str(loopcount) + ' out of ' + str(velolen) + ' steps')
    loopcount += 1
```

Loops

for loops start with the first item and execute code through to the last item

```
step 1 out of 5 steps
step 2 out of 5 steps
step 3 out of 5 steps
step 4 out of 5 steps
step 5 out of 5 steps
```

Each time the code block is executed it is an iteration

```
# now try this code;
velocities = [0.0, 9.81, 19.62, 29.43]
speed = 2
for speed in velocities:   # note: speed IS defined
    print('Metric:', speed, 'm/sec')

print('Final(speed):', speed)
```

```
Metric: 0.0 m/sec
Metric: 9.81 m/sec
Metric: 19.62 m/sec
Metric: 29.43 m/sec
Final(speed): 29.43
```

value of speed at end has changed to last one in list

Loops

for loops can iterate through strings, character by character

```
country = 'United States of America'  # 24 chars
for ch in country:
    if ch.isupper():  # 24 times
        print(ch)     # 3 times
```

How to print out 'USA'?

```
country = 'United States of America'
capstr = ''  # must be defined as (empty) string
for ch in country:
    if ch.isupper():
        capstr = capstr + ch  # can't use += for strings

print(capstr)
```

Loops

for loops can iterate using numbers

uses the range(int) function : all integers from start (low) to <= stop (high)

```
range(10)
range(0,10)

# copy into file editor:
for num in range(10):
    print(num)
0
1
2
3
4
5
6
7
8
9
```

Loops

```
# print 1 through 10:
for num in range(10):
    print(num+1)
```

```
1
2
3
4
5
6
7
8
9
10
```

```
num  # what is the value now?
```

```
9
```

Loops

You can coerce (convert) a range to a list

```
list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
list(range(2))
[0, 1]

# range with 2 numbers: start value, stop value (NOT included)
list(range(1,5))
[1, 2, 3, 4]

# range with 3 numbers: start value, stop value (NOT included), step value
list(range(1,10,2))
[1, 3, 5, 7, 9]
#election years
list(range(1900,2020,4))
[1900, 1904, 1908, 1912, 1916, 1920, 1924, 1928, 1932, 1936, 1940, 1944, 1948, 1952, 1956, 1960,
   1964, 1968, 1972, 1976, 1980, 1984, 1988, 1992, 1996, 2000, 2004, 2008, 2012, 2016]
```

Loops

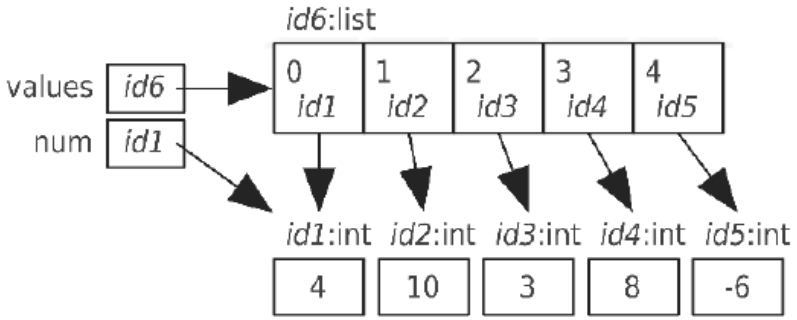
```
# going backwards: larger number, smaller number, negative number
# leap years backwards
list(range(2050, 2000, -4))
[2050, 2046, 2042, 2038, 2034, 2030, 2026, 2022, 2018, 2014, 2010, 2006, 2002]

# pay attention to order of start, stop, and value of increment
list(range(2000, 2050, -4))
[]
list(range(2050, 2000, 4))
[]

# if we want the sum of 100 numbers, do something 100 times, add 1:
total = 0
for i in range(1, 101): # 101 will NOT be included (stop number)
    total = total + i
print(total)
5050
```

Loops

List values are unaffected by simple loops

```
lvalues = [4, 10, 3, 8, -6]
for num in lvalues:
    num = num * 2
print(lvalues)
[4, 10, 3, 8, -6]
num
-12
```

```
for num in values:
```




```
lvalues = [4, 10, 3, 8, -6]
for num in lvalues:
    num = num * 2
    print(num)
8
20
6
16
-12
```

Loops

What if we want to change the values in a list?

We can't loop through numbers we are changing

We can loop through the indices of the list lvalues[0,1,2,3..]

```
list(range(len(lvalues)))  # len(lvalues) = 5; list(range(5)) = [0,1,2,3,4]
[0, 1, 2, 3, 4]
# so use code:
values = [4, 10, 3, 8, -6]
for i in range(len(values)):  # index number 0 to 4
    print(i, values[i])

values = [4, 10, 3, 8, -6]
for i in range(len(values)):   # index number 0 to 4
    values[i] = values[i] * 2  # index number 0 to 4
print(values)
```

Loops

Parallel processing using indices

```
# corresponding data by index: code
metals = ['Li', 'Na', 'K']
awts = [6.941, 22.98976928, 39.0983]
for i in range(len(metals)):
    print(metals[i], awts[i])
    #print(metals[i] + '\t' + str(awts[i]))
```

Note: using metals index means number of metal items <= number of awts items;
if not, IndexError: list index out of range

You might call for item 4 in one list but there are only 3 in the other.

Loops

Nested Loops in Loops (permutations)

How many ways can we pair one of 3 men with one of 3 women?

(speed dating)

```
Men = ['M1','M2','M3']
Women = ['F1','F2','F3']
for man in Men:
    for woman in Women:
        print(man + '-' + woman)
```

```
M1-F1
M1-F2
M1-F3
M2-F1
M2-F2
M2-F3
M3-F1
M3-F2
M3-F3
```

```
# example from text:
outer = ['Li', 'Na', 'K']
inner = ['F', 'Cl', 'Br']
for metal in outer:
    for halogen in inner:
        print(metal + halogen)
```

Loops

Let's print a multiplication table (copy and paste into code editor from text file)

```
def print_mtab(maxn):
    """ (int) -> NoneType (matrix)
    Print the multiplication table for numbers 1 through n inclusive.
    >>> print_table(5)
    1 2 3 4 5
    1 1 2 3 4 5
    2 2 4 6 8 10
    3 3 6 9 12 15
    4 4 8 12 16 20
    5 5 10 15 20 25
    """
    # The numbers to include in the table.
    numbers = list(range(1, maxn + 1))

    # Print the header row.
    for i in numbers:
        print('\t' + str(i), end='')

    # End the header row.
    print()

    # Print each row number and the contents of each row.
    for i in numbers:
        print (i, end='')
        for j in numbers:
            print('\t' + str(i * j), end='')

        # End the current row.
        print()
```

Loops

Let's print a multiplication table *prettier* (copy and paste into code editor from text file)

```
def print_mtab(maxn):
    """ (int) -> NoneType (matrix)
    Print the multiplication table for numbers 1 through n inclusive.
    >>> print_table(5)
    1 2 3 4 5
    1 1 2 3 4 5
    2 2 4 6 8 10
    3 3 6 9 12 15
    4 4 8 12 16 20
    5 5 10 15 20 25
    """
    # The numbers to include in the table.
    numbers = list(range(1, maxn + 1))

    # Print the header row.
    for i in numbers:
        print('\t' + rjust(str(i)), end='')

    # End the header row.
    print()

    # Print each row number and the contents of each row.
    for i in numbers:
        print (i, end='')
        for j in numbers:
            print('\t' + rjust(str(i * j)), end='')

        # End the current row.
        print()

def rjust(num):
    return str.rjust(num,2)
```

Loops

Looping until a condition is reached or something happens: while

```
while «expression»:
«block»
```

In file editor:

```
#rabbits die :-(
rabbitcount = 5
while rabbitcount > 0:
    print(rabbitcount)  # before decrease
    rabbitcount = rabbitcount -1
```

```
5
4
3
2
1
```

Loops

Looping until a condition is reached or something happens: while

```
while «expression»:
«block»
```

In file editor:

```
#rabbits reproduce! :-)
rabbitcount = 2
while rabbitcount > 0:
    print(rabbitcount)
    rabbitcount = rabbitcount * 2
```

```
2
4
8
16
32
64
128...  (press Ctrl-C to stop)
```

Loops

Looping until a condition is reached or something happens: while

```
while «expression»:
«block»
```

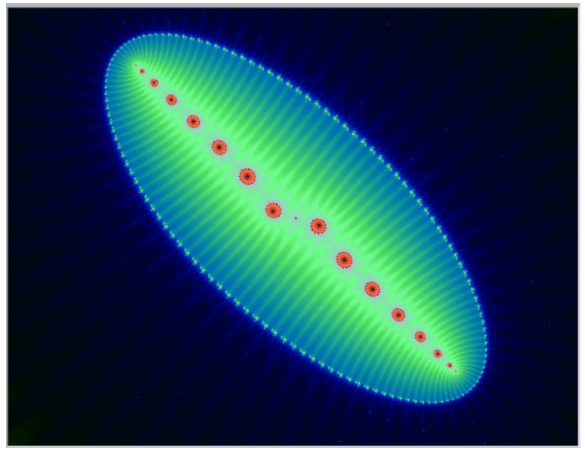
In editor:

```
#rabbits reproduce until there isn't enough food
rabbitcount = 2
while rabbitcount < 2000:
    print(rabbitcount)  # before rabbitcount increase
    rabbitcount = rabbitcount * 2
```

```
2
4
8
16
32
64
128
256
512
1024
```

Loops

Looping until a condition is reached or something happens: while

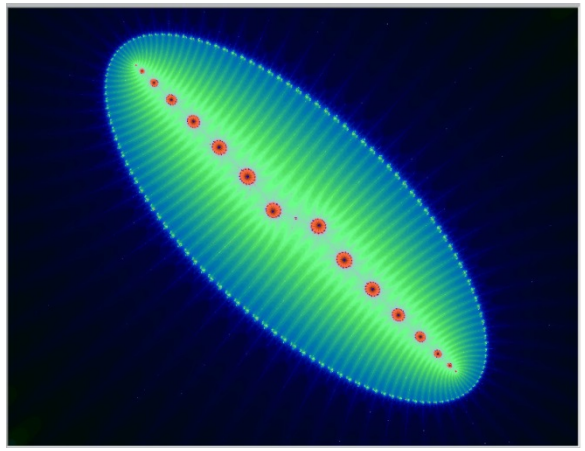


```
# bacteria reproduce until there isn't enough food
time = 0
population = 1000 # 1000 bacteria to start with
growth_rate = 0.21 # 21% growth per minute
while population < 2000:
    population = population + growth_rate * population
    print(round(population))  # after population increase
    time = time + 1
print("It took", time, "minutes for the bacteria to double.")
print("The final population was", round(population), "bacteria.")
```

```
1210
1464
1772
2144
It took 4 minutes for the bacteria to double.
The final population was 2144 bacteria.
```

Loops

Looping until a condition is reached or something happens: while



```
# bacteria reproduce until they are exactly 2000
time = 0
population = 1000 # 1000 bacteria to start with
growth_rate = 0.21 # 21% growth per minute
while population != 2000:
    time = time + 1
    population = population + growth_rate * population
    print(time, round(population) )  # after population increase
print("It took", time, "minutes for the bacteria to double.")
print("The final population was", round(population), "bacteria.")
```

```
1 1210
2 1464
3 1772
4 2144
5 2594 ... oops - an infinite loop again
```

Loops

Looping until a condition is reached or something happens: while

- based on user input (note the indents!)

```
# Similar to earlier code
text = ""
while text != "quit":
    text = input("Please enter a chemical formula H2O,NH3,CH4 (or 'quit' to exit): ")
    if text == "quit":
        print("exiting program...")
    elif text == "H2O":
        print("Water")
    elif text == "NH3":
        print("Ammonia")
    elif text == "CH4":
        print("Methane")
    else:
        print("Unknown compound")
```

Loops

Looping until a condition is reached or something happens: while

- interrupting flow, maybe when errors: break

```
# stop with break
text = ""
while text != "quit":
    text = input("Please enter a chemical formula H2O,NH3,CH4 (or 'quit' to exit): ")
    if text == "quit":
        print("exiting program...")
        break  # new, exit the while loop - go to next code block OUTSIDE loop
    elif text == "H2O":
        print("Water")
    elif text == "NH3":
        print("Ammonia")
    elif text == "CH4":
        print("Methane")
    else:
        print("Unknown compound")
```

An Infinite Loop: while True

Looping forever (or until break): while True

```
# while ... forever?
text = ""
while True:
    text = input("Please enter a chemical formula H2O,NH3,CH4 (or 'quit' to exit): ")
    if text == "quit":
        print("exiting program...")  # "break" in next line needed in order to stop!
    elif text == "H2O":
        print("Water")
    elif text == "NH3":
        print("Ammonia")
    elif text == "CH4":
        print("Methane")
    else:
        print("Unknown compound")
```

You can only exit through pressing Ctrl-C (bad practice) - be sure to provide clear comments!

Loops

Looping until a condition is reached or something happens

```
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
digit_index = -1 # This will be -1 until we find a digit.
# we want to find first digit, then do something else
for i in range(len(text)):
    # If character [i] is a digit
    if text[i].isdigit():
        digit_index = i
        print(digit_index)
```

```
9
13
14
18
19
24
25
```

Loops

Looping until a condition is reached or something happens

```
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
digit_index = -1 # This will be -1 until we find a digit.
# we want to find first digit, then do something else
for i in range(len(text)):
    # If we haven't found a digit, and s[i] is a digit
    if digit_index == -1 and text[i].isdigit():
        digit_index = i
        print(digit_index)
9
```

- BUT how many times did it go through the code block this time?

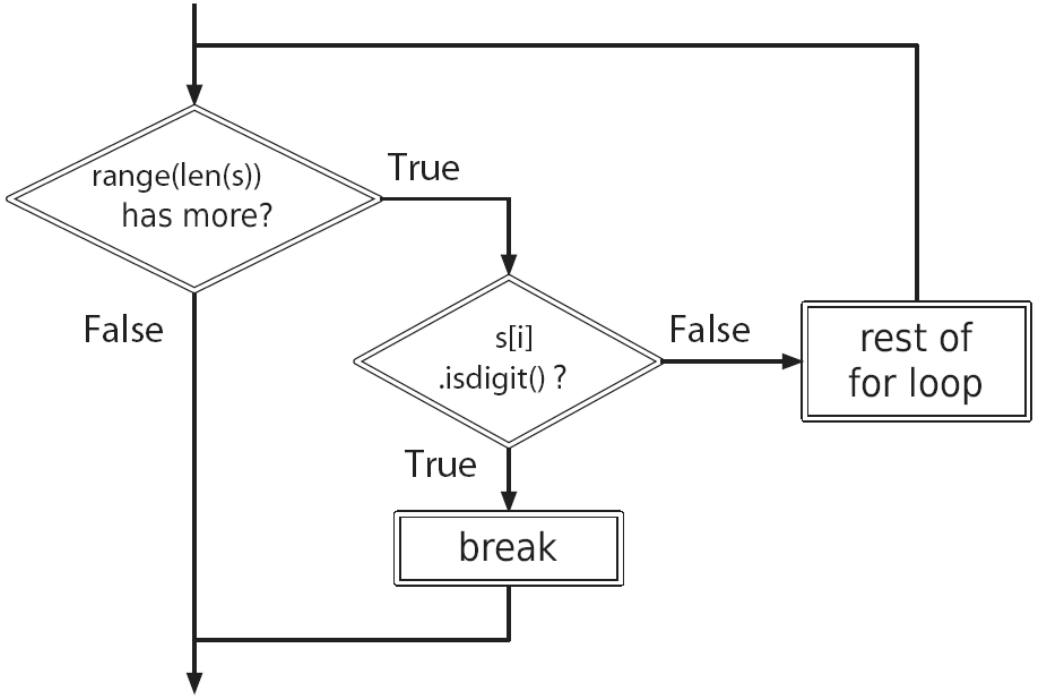
```
>>> print(i)
```

Loops

Looping until a condition is reached or something happens

- interrupting flow, maybe when errors: break

```
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
digit_index = -1 # This will be -1 until we find a digit.
# we want to find first digit, then do something else
for i in range(len(text)):
    # If we find a digit
    if text[i].isdigit():
        digit_index = i
        print(digit_index)
        break
9
>>> print(i)

- it stopped after finding the digit.
```



Loops

Looping until a condition is reached or something happens

- cycle back to top of block, keep going, maybe when errors: continue

```
# first try
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
sumDfound = 0 # The sum of the digits seen so far.
numDfound = 0 # The number of digits seen so far.
for i in range(len(text)):
    if text[i].isalpha():
        continue
        sumDfound = sunDfound + int(text[i])
        numDfound = numDfound + 1


print(sumDfound)
print(numDfound)
0
0
```

```
>>> s = 'C3H7'
>>> total = 0 # The sum of the digits seen so far.
>>> count = 0 # The number of digits seen so far.
>>> for i in range(len(s)):
...     if s[i].isalpha():
...         continue
...     total = total + int(s[i])
...     count = count + 1
...
>>> total
10
>>> count
2
```

Loops

Looping until a condition is reached or something happens: while

- cycle back to top of block, keep going, maybe when errors: continue

```
# second try
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
sumDfound = 0 # The sum of the digits seen so far.
numDfound = 0 # The number of digits seen so far.
for i in range(len(text)):
    if text[i].isalpha():
        continue
    sumDfound = sumDfound + int(text[i])
    numDfound = numDfound + 1

print(sumDfound)
print(numDfound)
```

```
77
19
```

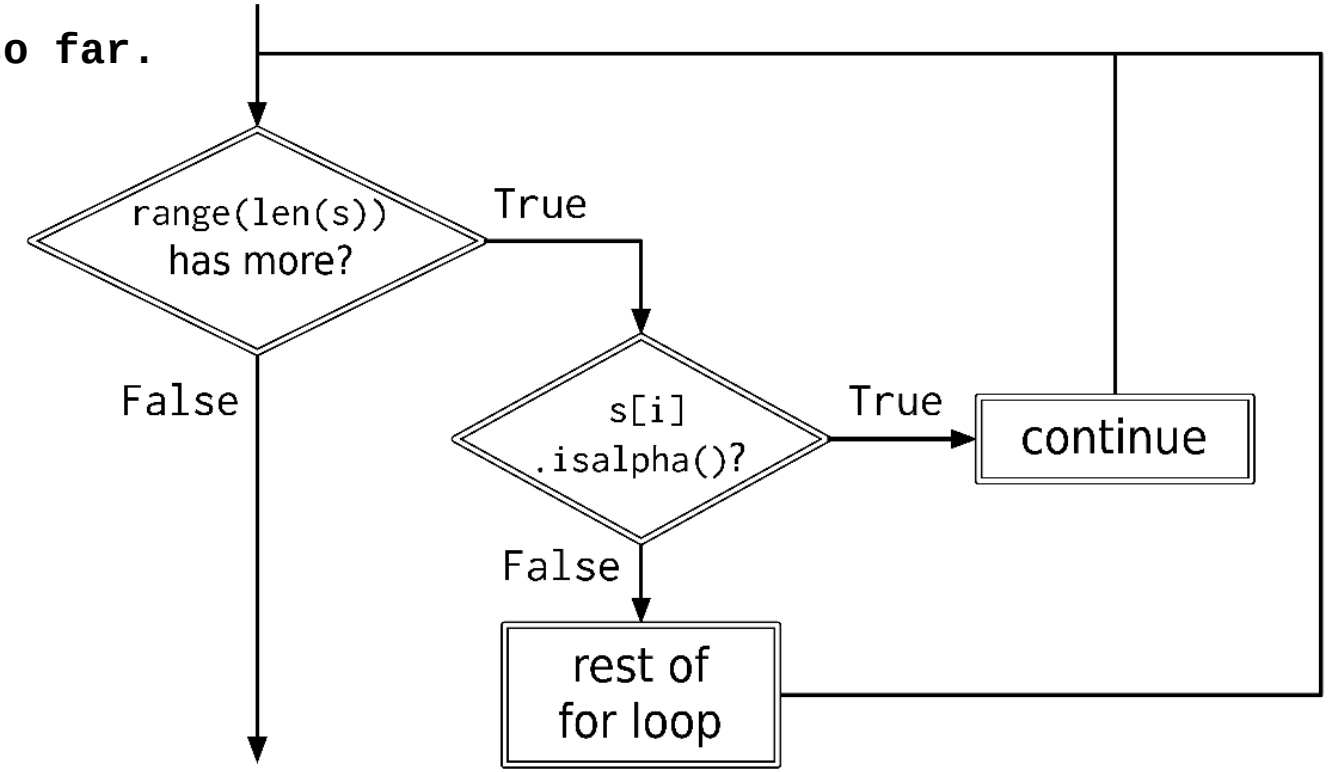
Loops

Looping until a condition is reached or something happens: while

- cycle back to top of block, keep going, maybe when errors: continue

```
# more efficient
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
sumDfound = 0 # The sum of the digits seen so far.
numDfound = 0 # The number of digits seen so far.
for i in range(len(text)):
    if text[i].isalpha():
        continue
    sumDfound += int(text[i])
    numDfound += 1

print(sumDfound)
print(numDfound)
```

```
77
19
```



Loops

Looping until a condition is reached or something happens: while

- cycle back to top of block, keep going, maybe when errors: continue

```
# even more efficient
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
sumDfound = 0 # The sum of the digits seen so far.
numDfound = 0 # The number of digits seen so far.
for i in range(len(text)):
    if not text[i].isalpha():
        sumDfound += int(text[i])
        numDfound += 1

print(sumDfound)
print(numDfound)
```

Loops

Looping until a condition is reached or something happens: while

- cycle back to top of block, keep going, maybe when errors: continue

```
# super efficient and will not choke on '-' or '@'
text = "kjhefkjhe4lkj45jhj43jlkj435444kuj5h34u53h45kj4"
sumDfound = 0 # The sum of the digits seen so far.
numDfound = 0 # The number of digits seen so far.
for i in range(len(text)):
    if text[i].isdigit():
        sumDfound += int(text[i])
        numDfound += 1

print(sumDfound)
print(numDfound)
```

Homework 9 due before Exam

Gries 9.10, page 166

1, 3, 6, 7, 13, 14, 15

For 13, 14, and 15, use a fixed-width font for output.

Times Roman	Courier New
T	T
TT	TT
TTT	TTT
TTTT	TTTT
TTTTT	TTTTT
TTTTTT	TTTTTT
TTTTTTT	TTTTTTT

Then:

A. Write programs to print the numbers from 1 to 10 (inclusive) using a for loop:

a. in descending order, all on one line.

b. in ascending order, all on one line.

c. in descending order, one per line.

d. in ascending order, one per line.

Readings for Wednesday

If Wednesday is **not** Hurst Day, we **will** have class.

Read Gries Chapter 10 for Wednesday.

Exam on Monday, October 11
Sample Exam Questions

What is the result of the following statements? (translate into Python code)

$(3 + 5)^5$

Are these legitimate statements?

```
gray = '324r2j243'234k234'324l324k'
s9 = 93.4455
s10 = 9,4,5,6
10s = 'teness'
```

come up with better variable names:

```
x = oxygen concentration
rab = count of rabbits
```

What is wrong with the following code? (I see 2 coding errors, one logic error)

```
while rabbits >= 200
rabbits = rabbits + 1
```