Control flow

Assignment stmt — name = expression

Conditional execution

```
if conditional expr:
    ≡
else:
    ≡
```

Loops

```
while condition expr:
    ≡
```

MULTIWAY CONDITIONAL STMT

```
def sgn(x):
# Return -1,0,1 if x<0, x=0, x>0
    if x < 0:
        return (-1)
    else:
        if x==0:
            return (0)
        else:
            return (1)
```

```
if x < 0:
    return (-1)
elif x == 0:
    return (0)
else:
    return (1)
```

More loops

for each element in some sequence

do something

Find the maximum value in a list

```
def maxlist(l)
    max = l[0]
    for each i in 1 to len(l)-1
        if l[i] > max
            max = l[i]
    return(max)
```

```
def maxlist(l):
    max = l[0]          # Assume l is non-empty!
    for i in range(1, len(l)): # same as [1..len(l)-1]
        if l[i] > max:
            max = l[i]
    return(max)
```

OR

```
for i in l:
    if i > max:
        max = i
```

```
max=l[0]
i=1
while i < len(l):
    if l[i] > max:
        max = l[i]
    i=i+1
return(max)
```

for i in list:

≡

↓

actually, an "iterable type"

range(m,n) → m, m+1, .., n-1

↓

not a list! (in Python 3)

Use list(range(m,n)) to get a real list

range(m,n,d)

m, m+d, ..

d can be negative

range(10, 0, -1)

Loops:

Skip a part of an iteration or abort

Continue, break

- Continue → goes back to top of loop
- break → exits loop immediately

```
for i in range(m,n):
    if i%2 == 0:
        continue
```

≡ } executes only for i%2 != 0

Searching for position of x in l

```
for i in range(0,len(l)):
    if l[i] == x:
        break
```

1. Exit via break, current value of i is location of first x in l

What if last value in l is x?

2. Exit via "for", no x in l, i = len(l) - 1

Loops also have "else:"

else: is executed only if loop ends normally

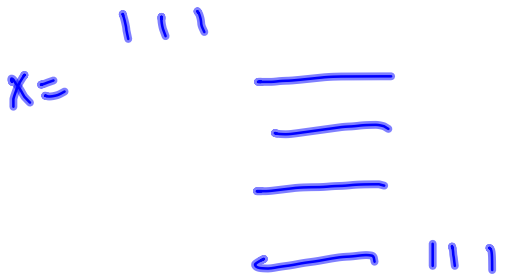
```
for i in range(0, len(l)):
    if l[i] == x:
        break
else:
    i = -1  # if no x in l, loop ends normally,
            # i set to -1
```

```
def firstpos(x,l):
    for i in range(0,len(l)):
        if l[i]==x:
            break
    else:
        i=-1
    return(i)
```

continue,
break
else:

can be used in while also

Strings:



A tuple of names can be assigned in one shot:

$(x, y, z) = (7, 8, 9)$

Deconstruct a tuple

pair = (5, 8)

(x, y) = pair

Interestingly

$(x,y) = (y,x)$ # swaps values

without an intermediate

Passing values to functions

```
def f(a,b,c):
    ==
```

```
w = f(x,y,2)
```

Implicitly

a = x
b = y
c = 2

before f starts

Mutable vs immutable value

x is immutable

```
a = x
:
update a
```

no effect on x

x is mutable

```
a = x
:
update a
```

both refer to same object / value

x is changed also

There is no way to write

```
def swap(a,b):
```

to swap 2 integers

Even for mutable objects

Can change contents

But not what the name points to

```
def f(l):
    l[0] = 7
```

```
v = [2,3,7]
f(v)
```

↪ v = [7,3,7]

```
def g(l):
    l = [7] + l[1:]
```

creates a new list

← v = l

l points to something else

v is unaffected

Functions to modify lists in place

l.append(x) l + [x]

in place new list

l.extend(l2) l + l2