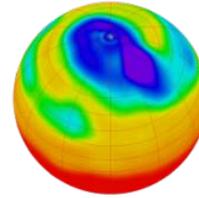




**National Centre for  
Atmospheric Science**  
NATURAL ENVIRONMENT RESEARCH COUNCIL



**Centre for Environmental  
Data Archival**

SCIENCE AND TECHNOLOGY FACILITIES COUNCIL  
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Python: more on functions

Extracted from material by:



You can assign a function to a variable

```
def threshold(signal):  
    return 1.0 / sum(signal)
```

```
t = threshold  
print t([0.1, 0.4, 0.2])  
1.42857
```

Can put (a reference to) the function in a list

```
def area(r):  
    return PI * r * r
```

```
def circumference(r):  
    return 2 * PI * r
```

```
funcs = [area, circumference]
```

```
for f in funcs:  
    print f(1.0)
```

3.14159

6.28318

Can pass (a reference to) the function into a function

```
def call_it(func, value):  
    return func(value)
```

```
print call_it(area, 1.0)  
3.14159
```

```
print call_it(circumference, 1.0)  
6.28318
```

Must need to know *something* about the function  
in order to call it

Must need to know *something* about the function  
in order to call it  
Like number of arguments

Must need to know *something* about the function  
in order to call it

Like ~~number of arguments~~

Must need to know *something* about the function  
in order to call it

Like ~~number of arguments~~

```
def add_all(*args):  
    total = 0  
    for a in args:  
        total += a  
    return total
```

Must need to know *something* about the function  
in order to call it

Like ~~number of arguments~~

```
def add_all(*args):  
    total = 0  
    for a in args:  
        total += a  
    return total
```

Must need to know *something* about the function  
in order to call it

Like ~~number of arguments~~

```
def add_all(*args):  
    total = 0  
    for a in args:  
        total += a  
    return total
```

```
print add_all()  
0
```

Must need to know *something* about the function  
in order to call it

Like ~~number of arguments~~

```
def add_all(*args):  
    total = 0  
    for a in args:  
        total += a  
    return total
```

```
print add_all()
```

0

```
print add_all(1, 2, 3)
```

6

# Connecting functions to sequences

filter(F, S)

select elements of S for which F is  
True

# Connecting functions to sequences

filter(F, S)	select elements of S for which F is True
map(F, S)	apply F to each element of S

# Connecting functions to sequences

filter(F, S)	select elements of S for which F is True
map(F, S)	apply F to each element of S
reduce(F, S)	use F to combine all elements of S

# Connecting functions to sequences

filter(F, S)	select elements of S for which F is True
map(F, S)	apply F to each element of S
reduce(F, S)	use F to combine all elements of S

```
def positive(x): return x >= 0
print filter(positive, [-3, -2, 0, 1, 2])
[0, 1, 2]
```

# Connecting functions to sequences

filter(F, S)	select elements of S for which F is True
map(F, S)	apply F to each element of S
reduce(F, S)	use F to combine all elements of S

```
def positive(x): return x >= 0
print filter(positive, [-3, -2, 0, 1, 2])
[0, 1, 2]
```

```
def negate(x): return -x
print map(negate, [-3, -2, 0, 1, 2])
[3, 2, 0, -1, -2]
```

# Connecting functions to sequences

filter(F, S)	select elements of S for which F is True
map(F, S)	apply F to each element of S
reduce(F, S)	use F to combine all elements of S

```
def positive(x): return x >= 0
print filter(positive, [-3, -2, 0, 1, 2])
[0, 1, 2]
```

```
def negate(x): return -x
print map(negate, [-3, -2, 0, 1, 2])
[3, 2, 0, -1, -2]
```

```
def add(x, y): return x+y
print reduce(add, [-3, -2, 0, 1, 2])
-2
```