Science & Technology Facilities Council | UK Research and Innovation | NERC SCIENCE OF THE ENVIRONMENT

Python

More on Functions

Centre for Environmental Data Analysis
SCIENCE AND TECHNOLOGY FACILITIES COUNCIL
NATURAL ENVIRONMENT RESEARCH COUNCIL

software carpentry

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

National Centre for Earth Observation
NATURAL ENVIRONMENT RESEARCH COUNCIL

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]))
1.4285714285714286
```

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 and sequences (1)

List Comprehensions come in handy:

1. Send a sequence to a function to create a new sequence of only positive numbers:

```
def positive(x):
    return x >= 0

print([x for x in [-3, -2, 0, 1, 2] if positive(x)])
[0, 1, 2]
```

Connecting functions and sequences (2)

2. Send a sequence of numbers to a function that will return the negative value of each item:

```
def negate(x):
    return -x

print([negate(x) for x in [-3, -2, 0, 1, 2]])
[3, 2, 0, -1, -2]
```