

# Lambda functions and errorhandling

# Lambda functions

#### Lambda functions

```
raise_to_power = lambda x, y: x ** y
raise_to_power(2, 3)
->8
```

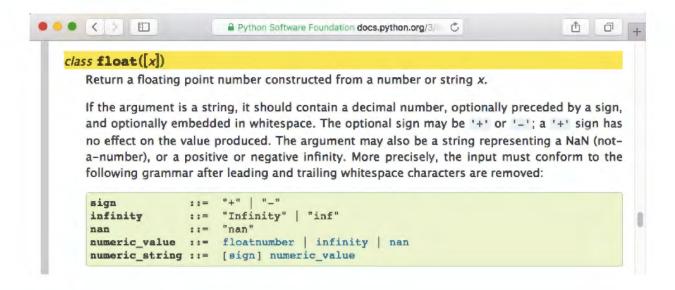
# **Anonymous functions**

- Function map takes two arguments: map(func, seq)
- map() applies the function to ALL elements in the sequence

```
nums = [48, 6, 9, 21, 1]
square_all = map(lambda num: num ** 2, nums)
print(square_all)
-><map object at 0x103e065c0>
print(list(square_all))
->[2304, 36, 81, 441, 1]
```

# Introduction to error handling

## The float() function



# Passing an incorrect argument

#### **Passing valid arguments**

```
def sqrt(x):
    """Returns the square root of a number."""
    return x ** (0.5)
sqrt(4)
->2.0
```

```
sqrt(10)
->3.1622776601683795
```

# Passing invalid arguments

```
sqrt('hello')
->------
TypeError Traceback (most recent call last)
<ipython-input-4-cfb99c64761f> in <module>()
----> 1 sqrt('hello')
<ipython-input-1-939b1a60b413> in sqrt(x)
1 def sqrt(x):
----> 2 return x**(0.5)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'float'
```

## **Errors and exceptions**

- Exceptions caught during execution
- Catch exceptions with try-except clause
   a)Runs the code following try
   b)If there's an exception, run the code following except

# **Errors and exceptions**

```
def sqrt(x):
    """Returns the square root of a number."""
    try:
        return x ** 0.5
    except:
        print('x must be an int or float')
sqrt(4)
->2.0
sqrt(10.0)
->3.1622776601683795
sqrt('hi')
->x must be an int or float
```

## **Errors and exceptions**

```
def sqrt(x):
  """Returns the square root of a number."""
    return x ** 0.5
  except TypeError:
    print('x must be an int or float')
Python Software Foundation docs.python.org/3/
                                                                                                    Ů, d
    exception TypeError
       Raised when an operation or function is applied to an object of inappropriate type. The associated value is a string
       giving details about the type mismatch.
    exception UnboundLocalError
       Raised when a reference is made to a local variable in a function or method, but no value has been bound to that
       variable. This is a subclass of NameError.
   exception UnicodeError
       Raised when a Unicode-related encoding or decoding error occurs. It is a subclass of ValueError.
       UnicodeError has attributes that describe the encoding or decoding error. For example,
       err.object[err.start:err.end] gives the particular invalid input that the codec failed on.
```

#### **Errors and exceptions**

```
sqrt(-9)
->(1.8369701987210297e-16+3j)
def sqrt(x):
 """Returns the square root of a number."""
 if x < 0:
    raise ValueError('x must be non-negative')
 try:
   return x ** 0.5
  except TypeError:
    print('x must be an int or float')
sqrt(-2)
  ValueError Traceback (most recent call last)
 <ipython-input-2-4cf32322fa95> in <module>()
  ----> 1 sqrt(-2)
 <ipython-input-1-a7b8126942e3> in sqrt(x)
 1 def sqrt(x):
 2 if x < 0:
 ----> 3 raise ValueError('x must be non-negative')
 4 try:
  5 return x^{**}(0.5)
 ValueError: x must be non-negative
```