PYTHON DATA SCIENCE TOOLBOX I

Lambda functions

Lambda functions

```
In [1]: raise_to_power = lambda x, y: x ** y

In [2]: raise_to_power(2, 3)
Out[2]: 8
```

Anonymous functions

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

```
In [1]: nums = [48, 6, 9, 21, 1]

In [2]: square_all = map(lambda num: num ** 2, nums)

In [3]: print(square_all)
<map object at 0x103e065c0>

In [4]: print(list(square_all))
[2304, 36, 81, 441, 1]
```

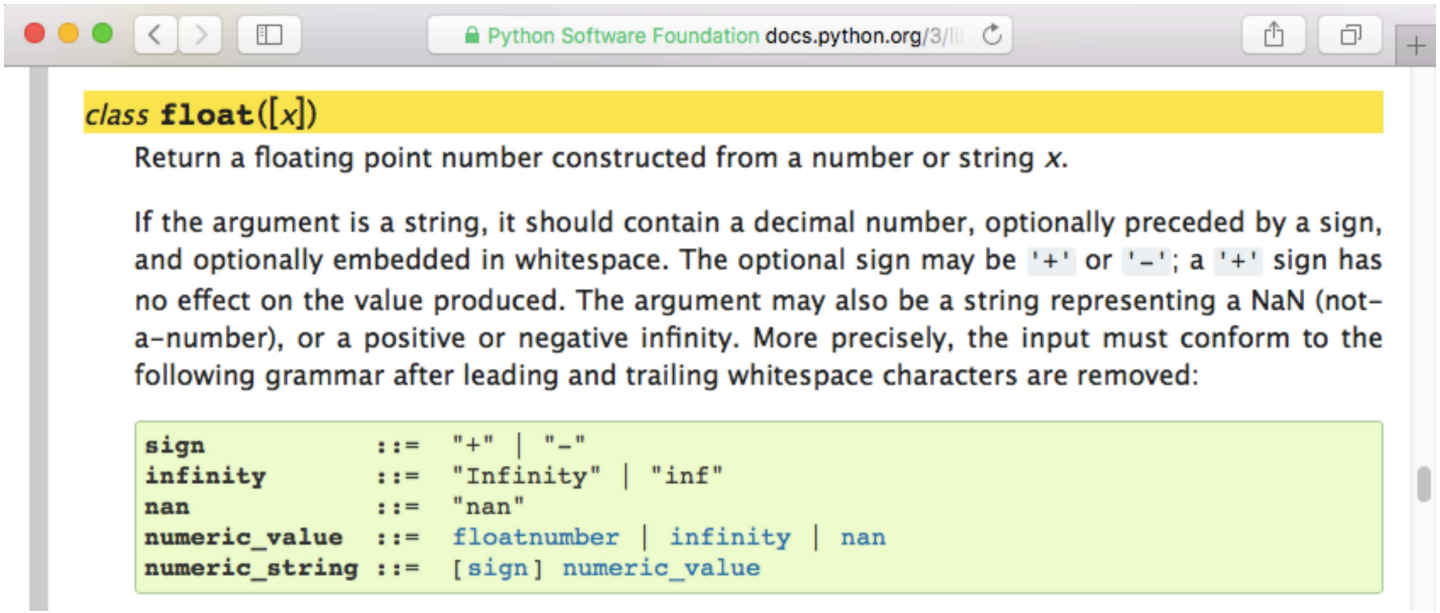
PYTHON DATA SCIENCE TOOLBOX I

Let's practice!

PYTHON DATA SCIENCE TOOLBOX I

Introduction to error handling

The float() function



Passing an incorrect argument

```
In [1]: float(2)
Out[1]: 2.0

In [2]: float('2.3')
Out[2]: 2.3

In [3]: float('hello')
---------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-d0ce8bccc8b2> in <module>()
----> 1 float('hi')
ValueError: could not convert string to float: 'hello'
```

Passing valid arguments

```
In [1]: def sqrt(x):
   ...:     """Returns the square root of a number."""
   ...:     return x ** (0.5)

In [2]: sqrt(4)
Out[2]: 2.0

In [3]: sqrt(10)
Out[3]: 3.1622776601683795
```

Passing invalid arguments

```
In [4]: 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

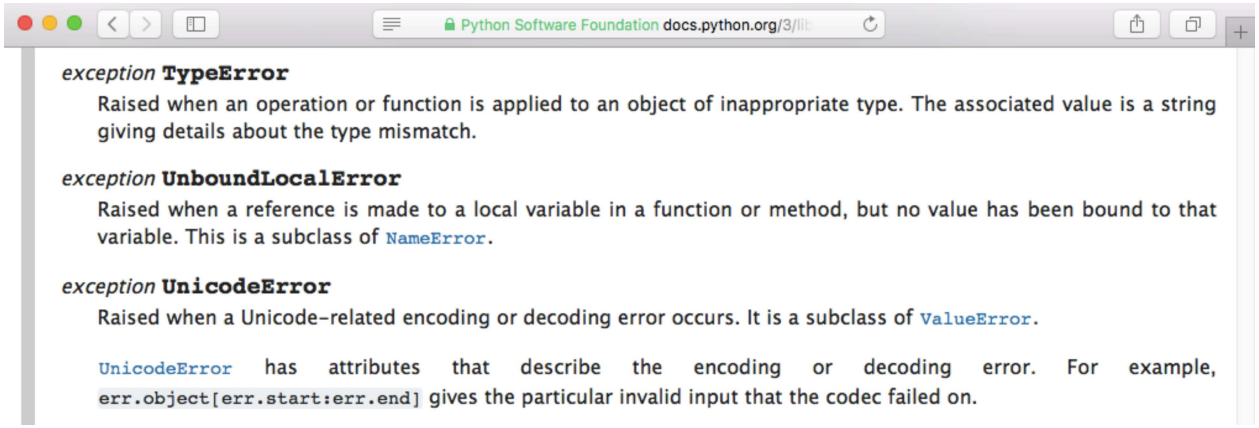
- Exceptions - caught during execution
- Catch exceptions with `try-except` clause
  - Runs the code following `try`
  - If there's an exception, run the code following `except`

Errors and exceptions

```
In [1]: def sqrt(x):
   ...:     """Returns the square root of a number."""
   ...:     try:
   ...:         return x ** 0.5
   ...:     except:
   ...:         print('x must be an int or float')

In [2]: sqrt(4)
Out[2]: 2.0

In [3]: sqrt(10.0)
Out[3]: 3.1622776601683795

In [4]: sqrt('hi')
x must be an int or float
```

Errors and exceptions

```
In [1]: def sqrt(x):
   ...:     """Returns the square root of a number."""
   ...:     try:
   ...:         return x ** 0.5
   ...:     except TypeError:
   ...:         print('x must be an int or float')
```



Errors and exceptions

```
In [2]: sqrt(-9)
Out[2]: (1.8369701987210297e-16+3j)
```

```
In [3]: 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')
```

Errors and exceptions

```
In [4]: 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
```

PYTHON DATA SCIENCE TOOLBOX I

Let's practice!

PYTHON DATA SCIENCE TOOLBOX I

Bringing it all together

Errors and exceptions

sqrt.py

```
def sqrt(x):
    try:
        return x ** 0.5
    except:
        print('x must be an int or float')
```

```
In [1]: sqrt(4)
Out[1]: 2.0

In [2]: sqrt('hi')
x must be an int or float
```

Errors and exceptions

sqrt.py

```
def sqrt(x):
    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')
```

PYTHON DATA SCIENCE TOOLBOX I

Let's practice!

PYTHON DATA SCIENCE TOOLBOX I

Congratulations!

What you've learned:

- Write functions that accept single and multiple arguments
- Write functions that return one or many values
- Use default, flexible, and keyword arguments
- Global and local scope in functions
- Write lambda functions
- Handle errors

There's more to learn!

- Create lists with list comprehensions
- Iterators - you've seen them before!
- Case studies to apply these techniques to Data Science

PYTHON DATA SCIENCE TOOLBOX I

Congratulations!