Course

- 🏠 CS-E4002
- 📖 Course materials
- 📊 Your points
- 💾 Code Vault ⧉

This course has already ended.

Lambda in Python

The name comes from lambda calculus invented by Alonzo Church in 1930s: A theoretic concept to describe what can be calculated (be aware that not everything can be computed - even with unlimited space). Lamda calculus is considered to be as strong as the concept of a Turing Machine (Church-Turing Hypothesis)

In python lambda is a way to anonymously describe a function: Anonymously means that the function does not need to have a name, aka is not described with the usual "def" keyword. The syntax looks like this:

```python
lambda x: x
```

The keyword lambda is followed by the variable x, after the ":" follows the function definition. In this case this definition is equivalent to the identity function:

```python
def identityFun(x):
    return x
```

Pythons lambda can take several arguments - they are separated by commatas.

```python
lambda x,y: x+y
```

This is equivalent to the sum function with two arguments:

```python
def sumFun(x,y):
    return x+y
```

A lambda function in python does not need to stay anonymous: it can be assigned a name.

```python
myLam = lambda a,b : a-b + 3

print(myLam(3,4))
```

Lambda is often used as a lightweight definition of functions. It also proves to be convenient for higher order functions. Higher order functions are functions that contain other functions as parameters or return a function. This is a very powerful and efficient way to program and we will see more of that later on. Commonly used higher order functions in Python are Map, Reduce, Zip and Filter.

Another higher order function you already encountered is the "apply"-method from pandas. The lambda notation becomes very handy for this method. As an example, the conversion from Fahrenheit to Celsius can be done conveniently like this:

```python
df["TAVG"].apply(lambda x: (x-32)*5/9.0)
```