

# Creative Data Mining

Lecture 03 : Concepts of python and programming

5 | 03 | 2018

Dr. Varun Ojha, [ojha@arch.ethz.ch](mailto:ojha@arch.ethz.ch)

Danielle Griego, [griego@arch.ethz.ch](mailto:griego@arch.ethz.ch)

# What we'll cover today

- Introduction to programming and python
- Hands-on python tutorial
  - Get familiar with the environment
  - Data types, Operators, Data structures, conditional statements, functions, packages...

# First, did everyone get python and spyder installed?

1. Install Python and Spyder through Anaconda  
<https://www.anaconda.com/download/>
2. Can also install Python and Spyder from <https://www.python.org/downloads/>  
and <https://pythonhosted.org/spyder/>



# What is programming?

- Programming is about solving problems and puzzles. You describe a precise set of instructions which the computer follows exactly.

For example if we want to sort a list:

```
list=[23, 44, 5, 17, 8, 90, 102]  
list.sort()  
print(list)
```

```
output: [5, 8, 17, 23, 44, 90, 102]
```

# Types and Values

- *Values* are for example: 5, 12.6, True or “Hello world”
- Each value is of a certain *type*
  - Numbers are: → numeric (integer or float)
  - True is: → boolean (only two values or states)
  - “Hello world” is: → a string (list of characters)

# Operators

- Operators represent a value manipulation (+, -, /, %, ....)
- It is important to check that the data type is correct, otherwise you might get an unexpected result.

Output :

```
p=int(17/9)      1
print(p)

p=round(17/9)   2
print(p)

p=float(17/9)   1.888888888888
print(p)
```

# Data Structures

There are three basic data structures in Python:

1. Lists- A sequence of values of any type
2. Lists provide several functionalities:

Example:

```
cdm=[ 'There are', [20, 17, 11, 7], 'students  
registered and', 2, 'instructors' ]
```

```
cdm[2]           #Accesses the third value  
cdm.remove(2)   #Removes the first occurrence of 2  
cdm.append(2)   #Adds 2 to the end of the list
```

# Data Structures

There are three basic data structures in Python:

2. Tuples- a sequence of values of any type, however cannot be altered after it is initiated.

Example:

```
#tuple
cdm_t=('There are', [20, 17, 11, 7], 'students
registered and', 2, 'instructors')
```

Vs.

```
#list
cdm_l=['There are', [20, 17, 11, 7], 'students
registered and', 2, 'instructors']
```



# Data Structures

There are three basic data structures in Python:

## Example

3. Dictionaries- store data as key-value pairs. Each key has an associated value.

```
CDM={'Number of students so far': [12, 15, 17, 8]}
```

```
CDM['Number of students so far']
```

```
# Returns [12,15,17,8]
```

```
CDM['final number']='usually 3/4 of first day'
```

```
#Adds a new element with key 'final number' and value 'usually 3/4 of the first day'
```

# Logical Operators

Logical Operators are expressions that evaluate to either *True* or *False*.

They are normally used in conditional statements.

Relational operators: `==`, `!=`, `<`, `>`, `<=`, `>=`, `and`, `or`, `not`

They stand for: equal, not equal, smaller, greater, smaller equal, greater equal, and, or, not

Examples:

`x < 2 or x >= 0`

`x == 0 or x != 0`

`not (x == 0 or x > 100)`

# Functions

If you do not want to retype the same code over and over again, you can define a function that you can call over and over again

Structure:

```
def functionName(parameters):  
    instructions  
    return variable
```

Example:

```
def square(a):  
    a = a * a  
    return a
```

# To get the square of 6, now just write:

```
square(6)
```

# Built-in python Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

Built-in Functions				
<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	
<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>	

<https://docs.python.org/3/library/functions.html>

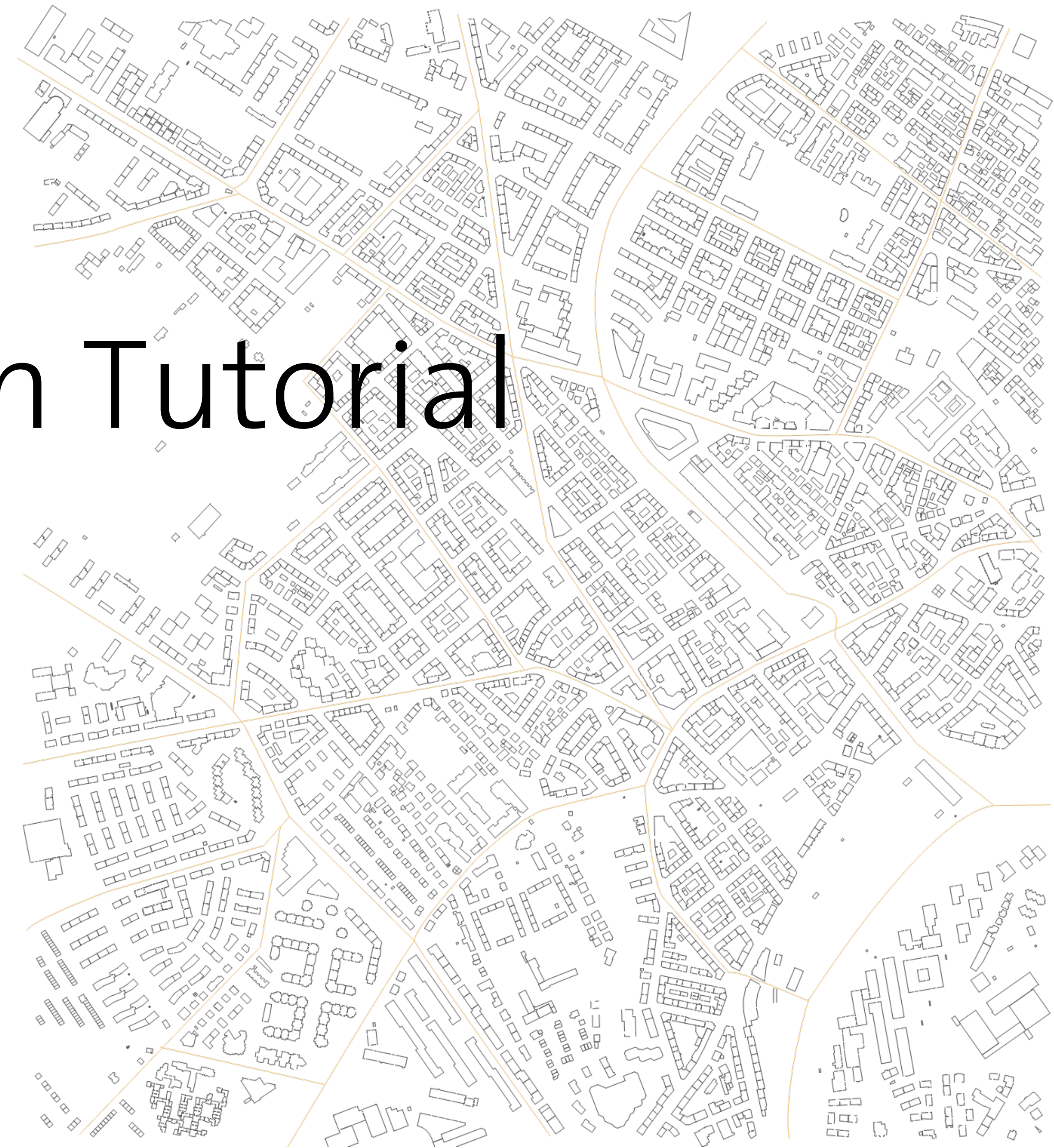
# Packages

- Often, more complicated code is already implemented and can be used as Packages. They help to be more efficient, because the programmer does not need to implement everything from scratch.
- When installed, packages can be integrated in the code using the keyword *import*.

## Example:

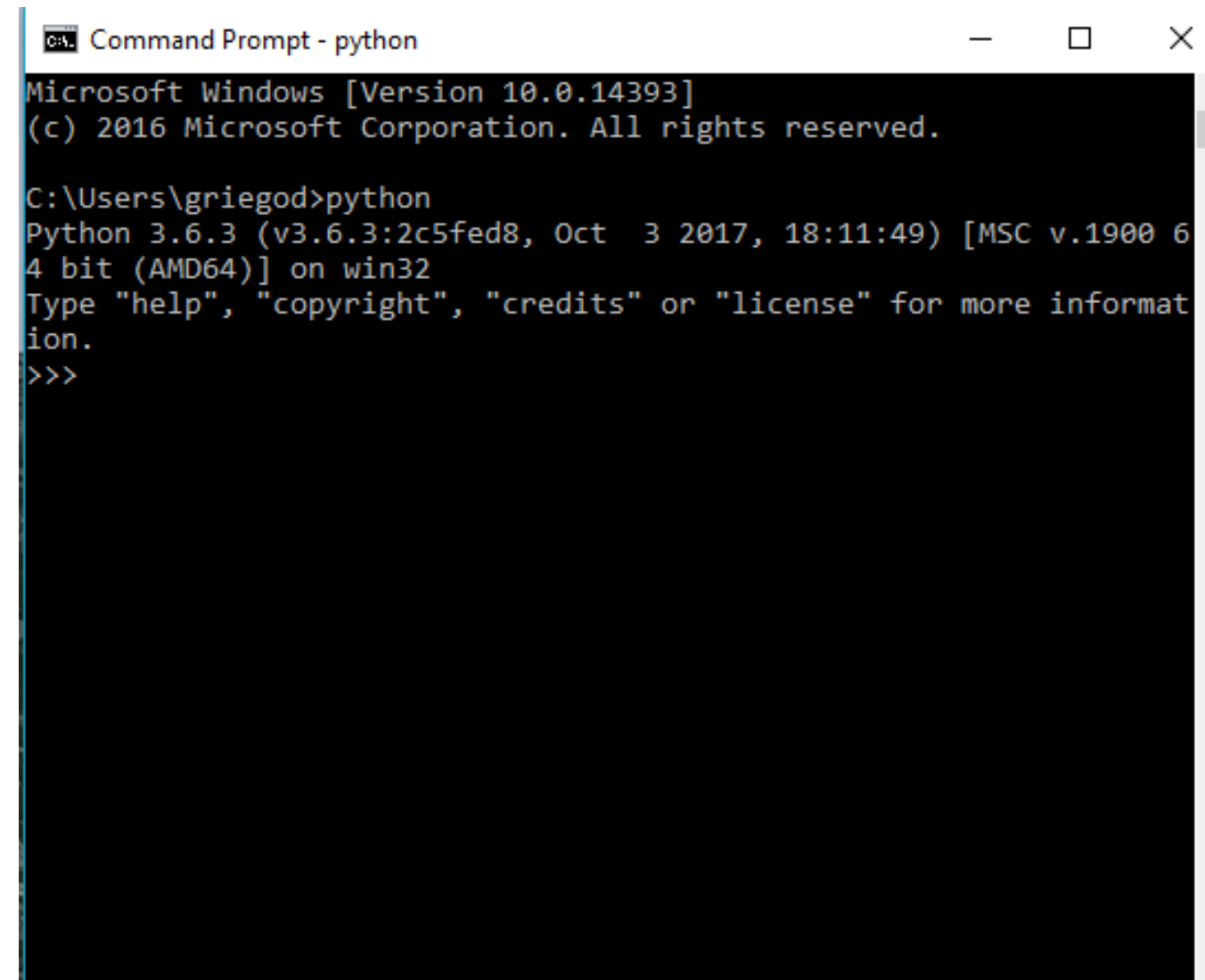
```
import pandas as pd
# imports the Pandas package, and now you can access the built in functions
```

# Hands-on Tutorial



# Python is the programming language

We can simply access it from the command prompt/terminal



```
Command Prompt - python
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\griegod>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## Python Interpreter version 2.x or 3.x

# Spyder is the IDE (interactive development environment)

There are many IDE's to choose from, we will show all tutorials in Spyder in the course

The screenshot displays the Spyder IDE interface. The main window is divided into three panes:

- Editor:** Contains a Python script named 'In-class tutorial.py'. The code includes comments and various operations like printing, list creation, and loops.
- Variable explorer:** A table showing the current state of variables in the workspace.
- IPython console:** Shows the execution of the code from the editor, including a traceback for a 'TypeError: 'str' object is not callable'.

Name /	Type	Size	Value
j	str	1	milk
list	str	1	I have a shopping list of 4 items
n	list	5	[1, 50, 39, 22, 20]
shoplist	list	5	['bread', 'carrots', 'cherries', 'coconut', 'milk']
year	list	8759	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...]

```
In [7]: print(239+785)
...: print(7 % 5) #modulo, operator that gives you the remainder when dividing one number by another
...: print(9**3) #power
...: print(9*3) #multiply
...: print(45.0/14) #divide, write one of the numbers as a decimal to get output as a float
...: print(2<5) #less than/greater than returns Ture/False
1024
2
729
27
3.2142857142857144
True

In [8]: list('I have a shopping list of 4 items') #string
...: shoplist=['carrots', 'coconut', 'cherries', 'bread'] #list of characters
...: print(shoplist)
...: print(list, shoplist)
['carrots', 'coconut', 'cherries', 'bread']
I have a shopping list of 4 items ['carrots', 'coconut', 'cherries', 'bread']

In [9]: shoplist.append('milk') #add an additional item to the given shopping list
...: print(shoplist)
...: print('I now have a shopping list of', len(shoplist), 'items') #using the function len to automate the item count value
['carrots', 'coconut', 'cherries', 'bread', 'milk']
I now have a shopping list of 5 items

In [10]: shoplist.sort() #sort the list by alphabetical order
...: print(shoplist)
['bread', 'carrots', 'cherries', 'coconut', 'milk']

In [11]: print('I also have to buy rice.', 'Dont forget to add it to the list')
I also have to buy rice. Dont forget to add it to the list

In [12]: print('I also have to buy rice.', '\nDont forget to add it to the list')
I also have to buy rice.
Dont forget to add it to the list


In [13]: n=[1,50,39,22,20]
...: print(n)
[1, 50, 39, 22, 20]

In [14]: year=list(range(1,8760)) #using range functions to produce a list of numbers
...: print(year)
Traceback (most recent call last):

  File <ipython-input-14-fd948cd38a9f>, line 1, in <module>
    year=list(range(1,8760)) #using range functions to produce a list of numbers
TypeError: 'str' object is not callable

In [15]: for j in shoplist:
...:     print(j)
bread
```





# Questions?

Lecture 3: Python I

5 | 03 | 2018

Dr. Varun Ojha, [ojha@arch.ethz.ch](mailto:ojha@arch.ethz.ch)

Danielle Griego, [griego@arch.ethz.ch](mailto:griego@arch.ethz.ch)