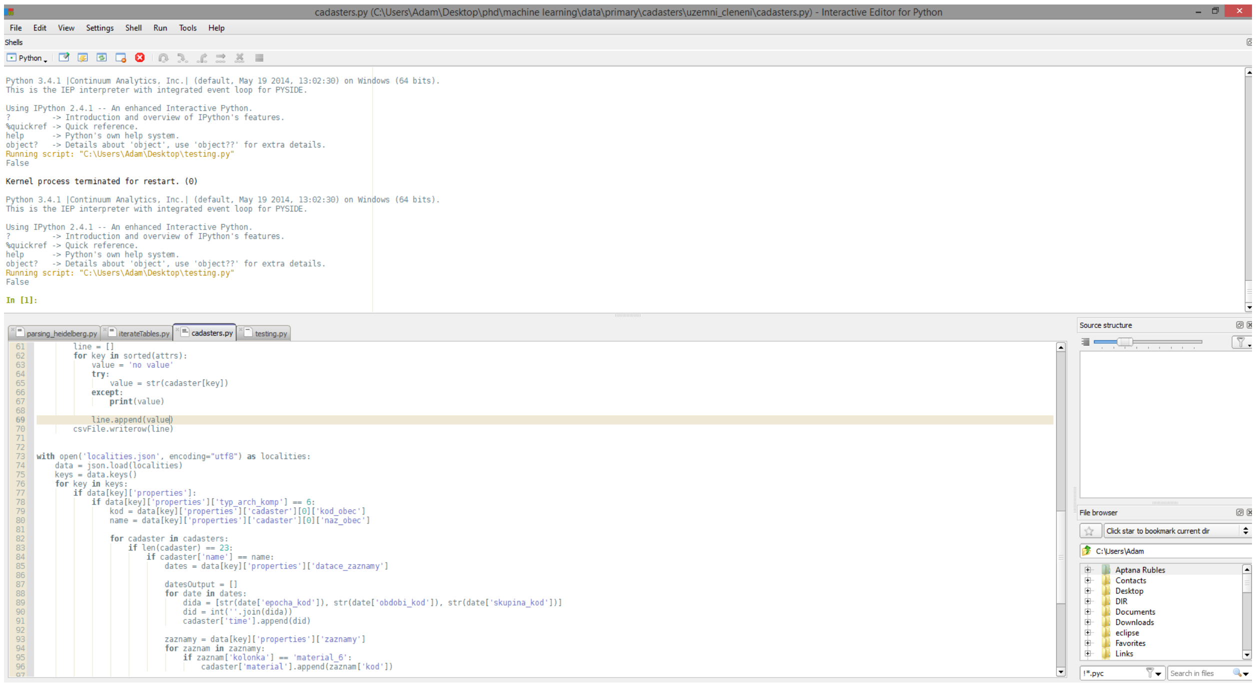
1. PYZO ENVIRONMENT (http://www.pyzo.org/)



```
Python 3.4.1 |Continuum Analytics, Inc.| (default, May 19 2014, 13:02:30) on Windows (64 bits).
This is the IEP interpreter with integrated event loop for PYSIDE.

Using IPython 2.4.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
Running script: "C:\Users\Adam\Desktop\testing.py"
False

Kernel process terminated for restart. (0)

Python 3.4.1 |Continuum Analytics, Inc.| (default, May 19 2014, 13:02:30) on Windows (64 bits).
This is the IEP interpreter with integrated event loop for PYSIDE.

Using IPython 2.4.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
Running script: "C:\Users\Adam\Desktop\testing.py"
False

In [1]:
```

```python
        line = []
        for key in sorted(attrs):
            value = 'no value'
            try:
                value = str(cadaster[key])
            except:
                print(value)

            line.append(value)
        csvFile.writerow(line)


with open('localities.json', encoding="utf8") as localities:
    data = json.load(localities)
    keys = data.keys()
    for key in keys:
        if data[key]['properties']:
            if data[key]['properties']['typ_arch_komp'] == 6:
                kod = data[key]['properties']['cadaster'][0]['kod_obec']
                name = data[key]['properties']['cadaster'][0]['naz_obec']

                for cadaster in cadasters:
                    if len(cadaster) == 23:
                        if cadaster['name'] == name:
                            dates = data[key]['properties']['datace_zaznamy']

                            datesOutput = []
                            for date in dates:
                                dida = [str(date['epocha_kod']), str(date['obdobi_kod']), str(date['skupina_kod'])]
                                did = int(''.join(dida))
                                cadaster['time'].append(did)

                            zaznamy = data[key]['properties']['zaznamy']
                            for zaznam in zaznamy:
                                if zaznam['kolonka'] == 'material_6':
                                    cadaster['material'].append(zaznam['kod'])
```

- interactive shell
- block of code (ctrl + E to compile, ctrl + S to save)

- python 3.4.1 (also has to be installed (to check: run cmd -> python -V))

2. VARIABLES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

2A: Assigning Values to Variables

The equal sign (=) is used to assign values to variables.

```
apples = 11
berries = 45

bus = 20
train = 52.56
```

Textual values are inside quotes:

```
word = 'hello'
sentence = 'Hello world, this is sentence.'
```

Not allowed values:

```
# mixing number with textual value
wrong1 = 'hello'45
wrong2 = 12men
wrong3 = 7'apples'

# …
wrong4 =
wrong5 = 'hello
wrong6 = 158 597
wrong6 = 158,597 # should be 158.597
```

Not allowed names:

```
12train = 'variable starting with number'
train-variable = 'variable name with special symbol'
hi there = 'variable with two words'
```

Allowed names:

```
	train11 = 'number not in the beginning'
	my_variable = 'variable name with underscore (only allowed
symbol)'
	myVariable = 'camelCase naming'
```

3. BASIC COMMANDS

3A: Print

Outputs to command line

- debugging
- informative purposes

without a variable

```
print('hello world')
print('ID485454?asd, aaabbcc')

print ('''
	There's something going on here.
	With the three double-quotes.
	We'll be able to type as much as we like.
	Even 4 lines if we want, or 5, or 6.
''')
```

printing textual variable (more possible ways)

```
fruit = 'banana'
print (fruit)
print ('favourite fruit: ' + fruit)

number_of_apples = 12
print ('Hi, I have', number_of_apples, 'apples.' )

fruit = 'banana'
number_of_apples = 12
print ('my favourite fruit is %s, but I have %s apples.' %(fruit,
number_of_apples))
```

3B: Input

Getting data from user

```
person = input('Enter your name: ')
print('Hello', person)

# print converts values to strings!
number_of_apples = input('How many apples do you have?')
print('Hello', person ,',you have' , number_of_apples ,'apples.')
# number_of_apples
```

3C: Comments

Information purposes, annotating code

```
# this code will be about fruits
fruit = 'banana' # this is my favourite fruit
```

4. NUMBERS *(https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex)*

4A: Numerical types

int (signed integers): They are often called just integers or ints, are positive or negative whole numbers with no decimal point.
float (floating point real values) : Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = 2.5 x 10^2 = 250).

```
litres_of_milk1 = 2.17 # float
litres_of_milk2 = 3.77 # float
number_of_people = 3 # integer
```

Converting numerical types with **int()** and **float()**

```
float(number_of_people)
int(litres_of_milk)
```

4B: Doing math

basic operations:

```
print (2 + 3) # addition
print (2.14 - 0.58) # subtraction
print (2 * 1.78) # multiplication
print (5.14 / 3) # division
```

other operations:

```
# negation
money_yesterday = 120
money_missing = -money_yesterday

# modulo - remainder after division
apples = 27
apples_for_pie = 5
apples_not_used = apples % apples_for_pie

# exponent, square root
square1_edge = 2
square1_area = pow(square1_edge, 3) # also square1_edge ** 2

square2_area = 3
square2_edge = pow(square2_area, 0.5)

# increment, decrement
region_inhabitants = 1532
region_inhabitants += 29 # immigration, region_inhabitants = 1561
region_inhabitants -= 15 # emigration, region_inhabitants = 1546
```

comparing numbers:

```
coef1 = 2.145
coef2 = 1.245
coef3 = 0.475
```

```
print (coef1 > coef2)
print (coef1 <= coef3) # smaller or equal
print (coef3 = coef2) # equal !
print (coef3 != coef2) # not equal
```

4C: Number methods

rounding round()

```
# round to integer
invited_people = 157
inviting_effectivity = 0.43
expected_people = round(invited_people * inviting_effectivity)

# rounding to decimal points
two_decimals = round(12.131321, 2)  # 12.13
```

maximum max(), minimum min()

```
region1 = 156
region2 = 17
region3 = 478
region4 = 69
regions_max = max(region1, region2, region3, region4)
regions_min = min(region1, region2, region3, region4)
```

...

5. STRINGS *(https://docs.python.org/2/library/string.html)*

variables with textual value

5A: String values

setting values, getting substrings with []

```
a_string = 'hello world!'
print (a_string ) # hello world!
print ('value of my string is', hello_string)

# getting substring (index 0 ~ position 1 !)
print (a_string[0]) # h -> character at index 0
print (a_string[-1]) # ! -> first character from the end
print (a_string[0:5]) # hello -> characters at indexes 0 to 5
```

string operators

```
# in returns true if a character exists in the given string
```

```
        a_string = 'hello world!'
        print ('a' in a_string) # False
        print ('hello' in a_string) # True

        # merging strings with +
        a = 'hello'
        b = 'world!'
        print (a + b) # helloworld

        # repetition with *
        a = 'hello'
        print (a * 4) # hellohellohellohello
```

comparing strings:

```
        name = "Bob"
        print (name == "Alex") # False
```

5A: String methods

(https://docs.python.org/3/library/stdtypes.html#string-methods)

find()

```
        # find() returns position of substring or -1
        hello_string = 'hello world!'
        print (hello_string.find('a')) # -1
        print (hello_string.find('l')) # 2
        print (hello_string.find('hello')) # 1

        # second (starting index) and third (ending index) parameter
        print (hello_string.find('l', 4)) # 9
        print (hello_string.find('o', 0, 5)) # 4
```

count()

```
        # count() return number of occurences
        hello_string = 'hello world!'
        print (hello_string.count('l')) # 3
```

isnumeric()

```
        # isnumeric() return true if string contains only numeric
characters
        text_string = 'hello world!'
        number_string = '1547854'
        print (text_string.isnumeric()) # False
        print (number_string.isnumeric()) # True
```

len()

```
# number of characters in string returns len()
example_string = 'a                 c'
print(len(example_string)) # 18
```

replace()

```
# replace changes substrings
example_string = 'hello world!'
print(example_string.replace('world', 'python')) #hello python!
```

6. BOOLEANS

Boolean values are the two constant objects False and True

bool()

```
# bool will evaluate 0, empty string or None value as False
number1 = 0
number2 = 0.0
number3 = 15
print (bool(number1)) # False
print (bool(number2)) # False
print (bool(number3)) # True

text1 = 'hello'
text2 = ''
print (bool(text1)) # True
print (bool(text2)) # False

nothing = None
print (bool(nothing)) # False
```

7. CONVERSION OF DATA TYPES

```
# int() converts to integer number
print (int(21.75)) # 21
print (int('456')) # 456
print (int(False)) # 0

# float()
print (float('1.2456')) # 1.2456
print (float(12)) # 12.0
print (float(True)) # 1.0

# str()
print (str(12)) # '12'
print (str(False)) # 'False'

# bool()
```

```
print (bool(-1)) # True
print (bool('hallo')) # True
print (bool('')) #False
print (bool(0)) #False
```

Motivation:

```
# problems with combination of data types
a_string = "blabla"
a_number = 15
print (a_string + a_number) # TypeError
```