Tutorial 1

Computer vision with python
Contact information

- Peleg Harel
- pelegh@post.bgu.ac.il (Please prefix mail title with ‘ICBV:’)
- Reception hours - monday 17:00 - 18:00 (after class)
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
Concepts
Duck typing

- If it walks like a duck and it quacks like a duck, then it must be a duck
- Variable has no type - object has types.

```python
In [2]:
def foo(a, b):
    print(a + b)

foo(1,2)
foo("a", "b")
```

3
ab
duck typing
Functions

- Definition contains function name and parameters name.
- Because of duck typing, names are important!
- No ; - End of statement is end of line.
- And of function body is defined by indentation level.
- Beware of mixing spaces and tabs

```python
In [2]:
    def add_numbers(x, y):
        print("I am going to add {} and {}".format(x,y))
        return x + y
    res = add_numbers(1, 2)
    print(res)
```

```
I am going to add 1 and 2
3
```
Collections

- List - Elements have order.
- Set - A collection of unique elements
- Dictionary - A Mapping from keys to values. Both can have any type
List comprehension

- A clear and readable way to transform lists
- Prefer this over raw for loops
- Can be used for all types of collections
For loops

- Iterates over a collection or an iterator (no c-style loops)
- Can have any number of instructions in loop body
- Same as functions - end of definition is defined by indentation level.
- **When transforming or filtering a list, prefer list comprehension.**
Indexing

1. You can select a single element (same as C arrays)
2. You can select a range of elements
3. You can use a negative index to select from the end of the list

```
In [6]: l = list(range(1,100))
print(l[12])
print(l[1:10])
print(l[:10])
print(l[80:])
print(l[80:-2])
print(l[10:20:3])
```

13
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
[81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97]
[11, 14, 17, 20]
Complete Python Tutorial

https://docs.python.org/3/tutorial/
Opencv and numpy
Libraries

- **OpenCV** - The largest open source computer vision library. Written in C++ and has an official Python binding.

- **NumPy/SciPy** - Python libraries for linear algebra and other mathematical operations. Operations are implemented in a low-level language and are extremely efficient.

- Both of the libraries work together.

- Some operators have both an OpenCV and a NumPy implementation (e.g. convolution)
Read and display an image

- An image will be read as a numpy array
- We can use matplotlib to display the image on screen

```python
In [12]: image = cv2.imread('duck.jpg', cv2.IMREAD_GRAYSCALE).astype(np.float)
   ...: imshow(image,
   ...:     cmap='gray',
   ...:     figure=figure(figsize=(6, 6)))

Out[12]: <matplotlib.image.AxesImage at 0x7fdba20c6588>
Numpy array indexing

1. Works the same as list indexing
2. You can use range selection to select subset or a whole dimension of the matrix
Matrix operations

- Numpy supports any typical matrix operations such as Add, subtract multiply by a scalar and Matrix multiplication
- SciPy contains Algorithms such as eigenvalue, and equation solver.