1. Introduction

Topics

- Self-Introduction
- Course structure (async home work, schedule)
- Links
    - https://vk.com/wall-54530371_1075
    - http://www.ibm.com/developerworks/ru/library/l-python_part_1/index.html
    - http://www.shashkovs.ru/_prog/Lutc_M._-_Izuchaem_Python_(4-e_izdanie)-_2011.pdf
    - http://pythontutor.com/visualize.html
    - https://www.checkio.org/
    -
- Python is everywhere: BitTorrent, Dropbox, World of Tanks, Instagram, etc. Heavy usage in statistics and scientific world.
- Why popular
    – Minimalism
    – Laconic
    – Integration
    – Universal
    – Slow, but who cares :)
    –
- Dynamic vs static languages. Pros and cons.
- Installing
    – versions
    – virtualenv
- Sublime
    – Package Manager (Ctrl+Shift+P), plugins, console, settings, build
    – PyCharm
- Start
    – hello world
    – import this, antigravity

Home work

- install python, sublime, virtualenv
- print “Hello, <Name>”

2. Types and expressions

Topics

- Idea of types as a set of associated operations:
    – Strings (‘, “, “”“): multiplication, concatenation, slicing, indexing.
    – Numbers, integers, real: power, different types of division //, /, %.
    – Arrays (string as an array of chars): length, map, filter, reverse.
    – Boolean: only 2 values with negation relation.
- vars
- vars of different types: sum of 2 numbers (strings, but not string and number)

- implicit conversions, rules
- how to read errors
- Python naming conventions: underscore-based for funcs and vars, camelazed for classes
- expressions
- number manipulation
    – modulo, sqrt, power, sin, cos
- string manipulation
    – multiplication, concatenation, slicing
    – create char pattern using string multiplication (i.e. “/\/\/\/\/”)
    – print dotted square with 2 patterns
    – *get the last/first/3rd/Nth element of a string
    – *swap the 1st letters in name and surname
    – *uppercase, lowercase all string/1st/last/3rd
    – *find the index of the 1st ‘a’ in your name
- module `math`

Class work

- arithmetic expressions
- string manipulations

Home work

- 1-6 from h/w

3. Functions & modules

Topics

- Functions
    – syntax, positioned parameters, named parameters
    – passing params by ref and by value
    – pass
    – name scope, collisions
    – builtin functions
    – functions composition
    – recursion
    – implicit return value None
- Modules
    – modules as a way to handle complexity
    – py files
    – importing system modules
    – writing user modules, importing them (with vars example)

Class work

- Square of number
- Power of number

- Fahrenheit to/from Celsius
- Radians to degrees

Home work

- Create different modules (~libs) for different topics
- Create function that sums 2 numbers
- Create function that adds tags to the passed string
- http://codingbat.com/python/String-1, 2
- 7-10 from h/w

4. Conditions

Topics

- Conditions
  – simple, chaining, example of difference between `elif` and 2 `if`s
  – check range inclusion (Python way!)
- Negation
  – boolean type
  – (not) empty list, >0

Class work

- isEven
- Find the largest number from 2
- Find average
- Fibonacci (compare to library one)
- Greatest Common Denominator
- Coding bat (http://codingbat.com/python/Logic-1, 2)

Home work

- Coding bat (http://codingbat.com/python/Logic-1, 2)
- 11-16 from h/w

5. Loops

Topics:

- Loops and iterations
  – for/while loop
  – nested loops
  – continue/break
  – varargs
  – no lists yet!

Class work:

- Print all squares from 1 to 100
- Print pyramid of ‘a’

```
a
```

aa
aaa
aaaa

- Create Pythagoras table
- Sum of passed numbers (*kvargs)
- Find largest, smallest
- Closest to X between n and m
- Prime numbers
- *Print all squares from 1 to 100 w/o multiplication

Home work:

- Print full pyramid of ‘a’

  a
  aa
  aaa
  aa
  a
  Hint: use function composition
- Parameterize with a symbol to print
- *Create collection of pyramids of increasing sizes (like in Giza!)
- 17-21 from h/w

6. Lists and arrays

*Topic*s:

- Collection of elements of homo- and heterogeneous types
- String as a list of chars
- List as a default container. When to use arrays.
- Iteration over the list
  – read-only
  – modification (index)
  – enumeration
- reverse()

Class work:

- Sum all elems in the list
- Sum all elems that are >N
- Sum all elems that are >N and <M
- Calc diff of all positive and negative
- Calc diff of all >N and <=N (when learn lists show how it is also possible with filters)
- Given a list of strings, print all elements which length is >N
- *Create own reverse() function to reverse list

Home work:

- http://codingbat.com/python/List-1, 2
- 22-26 from h/w

7. Dictionaries

Topics:

- how to create
- adding/removing values
- replacing, accumulating side effect, values can repeat, but not keys!
- iterating over keys/values
- list of dics
- nested dics
- switch for poor ppl
- tuples (swap, return 2+ values)

Class work:

- Given the following dictionary:

```
inventory = {
   'gold' : 500,
   'pouch' : ['flint', 'twine', 'gemstone'],
   'backpack' : ['xylophone','dagger', 'bedroll','bread 
loaf']
}
```

Try to do the followings:

Add a key to inventory called 'pocket'.
Set the value of 'pocket' to be a list consisting of the strings 'seashell', 'strange berry', and 'lint'.
.sort()the items in the list stored under the 'backpack' key.
Then .remove('dagger') from the list of items stored under the 'backpack' key.
Add 50 to the number stored under the 'gold' key.

- Create a new dictionary called prices.

Put these values in your `prices` dictionary:

```
  "banana": 4,
  "apple": 2,
  "orange": 1.5,
  "pear": 3
```

Create a new dictionary called stocks.

Put these values in your `stocks` dictionary:

```
  "banana": 6,
  "apple": 0,
  "orange": 32,
  "pear": 15
```

Loop through each key in `prices`. For each key, print out the key along with its price and stock information. Print the answer in the following format:

apple
price: 2
stock: 0
banana
price: 4
stock: 6
...

Home work:

– Calculate total sum for the given food from the class work dic
– Calculate total sum for all foods
– Add dic `discounts` with food items and %% of discount for them
– Loop through each key in `prices` and recalculate final price according to the discount
– <tdb>

8. Sort, ?search, ?group by

*Topic*s:

- types of sort: stable, not stable, bubble, quick
- sort list of strings and numbers
- sort list of dics: by key, value
- ?binary search
- ?group by value (number of repetitions)

Class work:

- writing bubble sort, test on large input
- idea of quick sort, test on large input, compare quick and bubble sort
- extract comparator: sort by string length, by value in dic, by user function
- idea of hash
- ?binary search
- ?calc statistical distribution of number in large input (group by + sort)

Home work:

- shaker sort (semi-inverted bubble)
- sort, ?search list by
    – number of 'a'
    – distance from X
- Given the list of dics with the following keys: `country;city;population`
    – Sort countries by name
    – Sort cities by name
    – Sort cities by population
    – *Sort countries by population (requires grouping!)

- 27-28 from h/w

9. Multi-dimensional arrays + test

Topics:

- multi-dimensional arrays
- matrix x scalar
- matrix x vector
- submit to github (web+sublime plugin, no console!)

Class work:

- vector x vector
- submit to github

Home work:

- 29-30 from h/w

10. Classes (basics+encapsulation)

Topics:

- test aftermath
- motivation:
    - complexity is handled with abstraction (main idea w/o details. example: car)
    - abstraction has its behavior and state, so class is an aggregation of data and behavior. Similar to dic, but with internal functions.
- when to use dics and classes?
- syntax
- difference between class and instance: blueprint vs real item
- Professor class example: {name, age, salary, dep}
- self
- constructor
- private/public
- dynamic class extension

Class work

- create class Student
- add methods:
    – Student::full_name()
    – Student::age()
    – Student::avg_gpa()
    – Student::supervisor()
    – Student::is_group_head()
    – Student::classes() → List<String>

Home work

- create class Class
- add methods:
    – Class::name()
    – Class::full_description()

    – Class::lecturer() → Professor
    – Class::lecturer_2nd() → Professor
- extend Professor with methods:
    – Professor::classes() → List<Class>
- modify Student with methods:
    – Student::classes() → List<Class>
- 31-32 from h/w

11. Classes (inheritance+polymorphism)

*Topic*s:

- motivation:
    - the same interface for different behaviors and why this is important.
- example with Dog, Cat : Animal
- name resolving
- simple UML notation to depict classes relations
- ?LSP, OCP
- derive Student, Professor from Person and move name, age to Person

Class work:

- for the given UML diagram create graphic shapes hierarchy: Circle. Rectangle, Triangle, Ellipse: Shape
- draw() and size() are base and polymorphic
- Triangle::base, Triangle::height
- Circle::center
- Rectangle::side1, Rectangle::side2

Home work

- 33 from h/w

12. Classes (practice)

*Topic*s:

- real life app!

Class work:

- create list of phone book entries {name, age, phone_number} using menu lib
- add commands
    – print phonebook
    – add entry
    – delete by id
    – search by name
    – search by phone number
    – restore/load from disk
- implement entry as class. Show benefits (encapsulated logic).

Home work

- add `is_family_member` flag in entry class
- add print family members command
- update the rest of commands (print, add)
- <tbd>: OOP
- 35 from h/w

13. Files

Topics

- binary vs text files
- open modes (r, w, r+, w+)
- read, seek
- write

Class work

- Read file content and print it on screen.
- Write in the file numbers and its cubes from 1 to 10. 2 numbers in one line. Numbers are delimited by tabular symbol.

Example:
```
1	1
2	8
...
```
- Add to the existing file from p.1 squares between a number and it’s cubes for all numbers.

Example:
```
1	1	1
2	4	8
```
- *Print out dir content (recursively)

Home work

- Create a file and fill it with 20 symbols 'a', so that the result is like:
  aaaaaaaaaaaaaaaaaaaa
- Open file from p.1 and modify it by inseritng symbols 'b', so that the result is like:
  abababababababab...ab
- Open file from p.1 and modify it by inseritng symbols 'b', so that the result is like:
  abbabbabbabbabbabbabb...abb
- Open file from p.1 and modify it by inseritng symbols 'b', so that the result is like:
  ababbabbbabbbbabbbbbabbbbbbab...abbbbbbbbbbbbbbbbbbbb

14. Files+dictionaries+simple statistics

Topics

- what is the most used word or letter in English? Lets analyze!

Class work

- read big book and calculate

    – word distribution
    – letter distribution
    – word length distribution
    – sentence length distribution
- compare Russian vs English books
- add command line interface
- create horizontal histogram

Home work

- read big book and calculate
    – avr. sentence length
    – avr. word length
- *find big project on github
    – pull
    – recursively traverse and collect statistics
    – compare for Java vs Python projects!

15-16. ?UI for Phonebook/Game/Buffer

Topics:

- *<tbd>*

Advanced Python topics

- lambdas, map/filter/zip
- metaprogramming
- python idioms
- decorators
- NumPy
- Flask, Django

Different CS topics

- FSM (draw diagram)
- github, plugins for sublime
- hex, regexp, json, csv

Teaching hacks:

- When introduce new syntax constructions (if, loop, func) do it with formal BNF rules!
- There are tips to write program: explain in natural language => pseudocode => code
- from small to big
- debugging: start with working core and iteratively add functionality until it breaks
- seek is like cursor when edit document
- when you give theory, they should practice immediately. This will recall gained knowledge (considered the best learning hack ever).
- create modules that collect typical operations under lists and dics and use them to demonstrate how we can export functions