Sisällys

[Pycharm key-shortcuts 6](#_Toc99301750)

[Links: 6](#_Toc99301751)

[Apps: 6](#_Toc99301752)

[Section 3 6](#_Toc99301753)

[Strings in Python 6](#_Toc99301754)

[Variables (value binding) 6](#_Toc99301755)

[Bytecode with Python - how to compile python to exe 6](#_Toc99301756)

[Numeric Data Types (int, float, decimal, complex - Python3) 7](#_Toc99301757)

[Numeric operators 7](#_Toc99301758)

[Operator Precedence 7](#_Toc99301759)

[Slicing (needed at least one column :) 7](#_Toc99301760)

[Using Step in a Slice 7](#_Toc99301761)

[Slicing backwards 8](#_Toc99301762)

[Python String Types (f format in use Python >= 3.6) 8](#_Toc99301763)

[Section 4 8](#_Toc99301764)

[Using debugger 8](#_Toc99301765)

[Conditional Operators 8](#_Toc99301766)

[Simplify Chained Comparison 8](#_Toc99301767)

[Boolean expressions (True or False with capital) 8](#_Toc99301768)

[Operator Precedence Table 8](#_Toc99301769)

[String methods 8](#_Toc99301770)

[For loops 9](#_Toc99301771)

[Python Build-in Functions 9](#_Toc99301772)

[Continue 9](#_Toc99301773)

[Break 9](#_Toc99301774)

[While 9](#_Toc99301775)

[Binary Search 9](#_Toc99301776)

[Augmented Assignment 9](#_Toc99301777)

[Pep 8 9](#_Toc99301778)

[Section 5 – List and Tuples (sequence types) 9](#_Toc99301779)

[Iterables have either: 9](#_Toc99301780)

[Mutable objects 10](#_Toc99301781)

[Methods and Function 10](#_Toc99301782)

[Enumerate 11](#_Toc99301783)

[Sorting 11](#_Toc99301784)

[Built-in-Functions 11](#_Toc99301785)

[Sorting 11](#_Toc99301786)

[Case sensitive sorting 12](#_Toc99301787)

[Creating Lists (about 20 ways to do it) 12](#_Toc99301788)

[Replacing a slice 12](#_Toc99301789)

[Deleting items from a list 13](#_Toc99301790)

[Safely removing items from a list 13](#_Toc99301791)

[Testing 13](#_Toc99301792)

[Removing items from list – backwards (Lesson 108) 13](#_Toc99301793)

[The Reversed Function https://docs.python.org/3/library/functions.html 14](#_Toc99301794)

[Algorithms Performance 14](#_Toc99301795)

[Pep8 – style guide 14](#_Toc99301796)

[Trailing commas 14](#_Toc99301797)

[Processing Nested Lists 14](#_Toc99301798)

[No spam with menu-list (2 different approaches) 14](#_Toc99301799)

[Print revisited 15](#_Toc99301800)

[The join method (lists) 15](#_Toc99301801)

[The split method 16](#_Toc99301802)

[Tuples are immutable 16](#_Toc99301803)

[Unpacking Tuples 16](#_Toc99301804)

[Nesting Further 17](#_Toc99301805)

[Constants 17](#_Toc99301806)

[Section 6 - Functions – introduction 17](#_Toc99301807)

[Palindromes 17](#_Toc99301808)

[String functions 17](#_Toc99301809)

[Range exercise 16 - sum(range(start, n, 2)) 17](#_Toc99301810)

[Handling invalid arguments (positional-or-keyword) 18](#_Toc99301811)

[Docstrings (using reStructuredText) 18](#_Toc99301812)

[Fibonacci numbers 19](#_Toc99301813)

[Documenting 19](#_Toc99301814)

[Function annotations and type hints 19](#_Toc99301815)

[Function annotations with default value 19](#_Toc99301816)

[Running program on terminal (msdos / powershell) 19](#_Toc99301817)

[Playing Fizz Buzz -game 20](#_Toc99301818)

[Defining different argument types 20](#_Toc99301819)

[Section 7 20](#_Toc99301820)

[What is a dictionary 20](#_Toc99301821)

[Zen of Python 21](#_Toc99301822)

[Using in with dictionary 21](#_Toc99301823)

[Checking quantities – tuple or dict 22](#_Toc99301824)

[setdefault -method 22](#_Toc99301825)

[Exercise 20 – char counter 23](#_Toc99301826)

[APIs and mobile phone demo 23](#_Toc99301827)

[- https://docs.python.org/3/library/getpass.html 23](#_Toc99301828)

[The dict documentation 23](#_Toc99301829)

[Python Branches 23](#_Toc99301830)

[dict objects 24](#_Toc99301831)

[Shallow copy Lesson 205 copies only reference 24](#_Toc99301832)

[Hash functions 24](#_Toc99301833)

[Tools – Python Console (console woindow) 24](#_Toc99301834)

[Hash tables 24](#_Toc99301835)

[hashlib 24](#_Toc99301836)

[Introduction to sets 25](#_Toc99301837)

[set membership 26](#_Toc99301838)

[Better performance 26](#_Toc99301839)

[Unique data in sets 27](#_Toc99301840)

[Set documentation 27](#_Toc99301841)

[The `pop` method 27](#_Toc99301842)

[Set union 27](#_Toc99301843)

[Symmetric Difference (you could use list) 28](#_Toc99301844)

[Subsets and supersets 28](#_Toc99301845)

[isdisjoint(other) 29](#_Toc99301846)

[Summary 29](#_Toc99301847)

[Section 8 Input and Output I/O 30](#_Toc99301848)

[Pickle 30](#_Toc99301849)

[Shelve 30](#_Toc99301850)

[Section 9 Modules and Functions 30](#_Toc99301851)

[Modules and import 30](#_Toc99301852)

[The Standard Python Library 30](#_Toc99301853)

[Webrowser Module 31](#_Toc99301854)

[Time and Datetime in Python 31](#_Toc99301855)

[Installing pytz module (FAQ) 31](#_Toc99301856)

[Installing pytz 31](#_Toc99301857)

[Using Timezones 32](#_Toc99301858)

[Tkinter 32](#_Toc99301859)

[Import System 32](#_Toc99301860)

[Lesson 288 recursive or iterative 32](#_Toc99301861)

[Factorial and Fibonacci 32](#_Toc99301862)

[OS module 33](#_Toc99301863)

[Nonlocal keyword, FREE and LEGB 33](#_Toc99301864)

[Section 10 – Object Oriented Programming 33](#_Toc99301865)

[Capsulation in Classes check from material 33](#_Toc99301866)

[Static Methods: 34](#_Toc99301867)

[Namespaces: 34](#_Toc99301868)

[DocString and Raw Literals 35](#_Toc99301869)

[Artist Class and import Albums (with circular reference) 35](#_Toc99301870)

[Compare Files and Algorithm Flowcharts 35](#_Toc99301871)

[If using Python 2 in song.py 35](#_Toc99301872)

[Challenge – remove circular references (aim to top diagram) 38](#_Toc99301873)

[Getters and Setters 38](#_Toc99301874)

[Inheritance (Python allows multiple inheritance but be aware that you should master it) 38](#_Toc99301875)

[Python no method overloading 39](#_Toc99301876)

[Python overriding 39](#_Toc99301877)

[tab Regex + (\{\d\.) + $1\_ Replace All 39](#_Toc99301878)

[Inheritance Challenge Lesson 314 40](#_Toc99301879)

[Polymorphism (Java in statically typed – Python dynamically typed) 41](#_Toc99301880)

[Inheritance isn’t the only way to implement polymorphism Lesson316 41](#_Toc99301881)

[Composition Lesson 317 & 319 HTML document demo 41](#_Toc99301882)

[Relationship described with 41](#_Toc99301883)

[Aggregation (weak form of composition) Lesson 320 41](#_Toc99301884)

[Section 11 – Using Databases with Python 42](#_Toc99301885)

[Introduction to SQLite 42](#_Toc99301886)

[Querying data with SQLite 42](#_Toc99301887)

[Autoincrement in SQLite 43](#_Toc99301888)

[More Complex Joins 44](#_Toc99301889)

[Wildcards and Views 44](#_Toc99301890)

[Views (for customer security plus to easy job) 45](#_Toc99301891)

[Housekeeping and Challenge 46](#_Toc99301892)

[SQL Challenge 47](#_Toc99301893)

[SQL in Python 49](#_Toc99301894)

[SQL Injection Attacks (parameter substitution with placeholders, sanitizing the input) 49](#_Toc99301895)

[Placeholders and Parameter Substitution 49](#_Toc99301896)

[Exceptions 49](#_Toc99301897)

[Hints (CTRL + Q) 49](#_Toc99301898)

[Adding Database code to Accounts Class 49](#_Toc99301899)

[Displaying Time in Different Timezones 49](#_Toc99301900)

[SQLite3 strftime Function 50](#_Toc99301901)

[Challenge and Problems Storing Time zones 50](#_Toc99301902)

[- https://www.sqlite.org/lang\_corefunc.html 50](#_Toc99301903)

[- https://www.sqlite.org/lang\_aggfunc.html 50](#_Toc99301904)

[Jukebox (Music Browser) 50](#_Toc99301905)

[Star args 50](#_Toc99301906)

[Jukebox final 51](#_Toc99301907)

[Questions 53](#_Toc99301908)

# Pycharm key-shortcuts

* ALT + SHIFT + arrow up
* CTRL + ALT + L (reformat code)
* CTRL + Q python docs

# Links:

* <https://www.javatpoint.com/python-tutorial>
* <https://www.timbuchalka.com/>

C:\Users\lauri\PycharmProjects\PythonMasterClass\

* <https://www.tutorialspoint.com/sqlite/sqlite_distinct_keyword.htm>

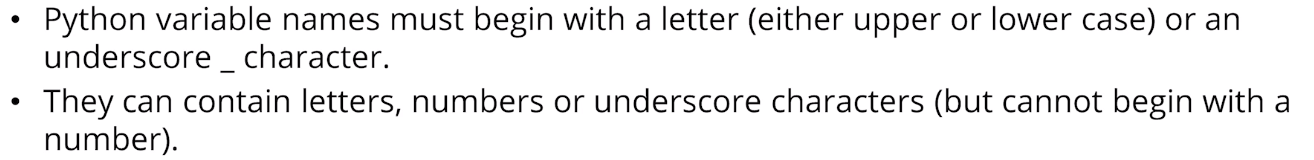
# Apps:

* Diffmerge
* C:\Users\lauri\Koodivertailu

# Section 3

## Strings in Python

## Variables (value binding)



## Bytecode with Python - how to compile python to exe

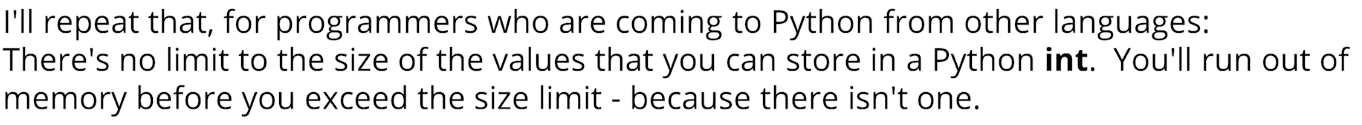
* <https://www.youtube.com/watch?v=NJB6RT0tsLI> (simple)
* <https://www.youtube.com/watch?v=UZX5kH72Yx4>
* pip list
* pip install pyinstaller
* pyinstaller --oneFile --w main.py
* NSIS Wiki, download NSIS

## Numeric Data Types (int, float, decimal, complex - Python3)

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

OBS! Int



Float 52 digits precision

## Numeric operators

a = 12  
b = 3  
  
print(a + b) *# 15*print(a - b) *# 9*print(a \* b) *# 36*print(a / b) *# 4.0*print(a // b) *# 4 integer division, rounded down towards minus infinity*print( a % b) *# modulo 0: the remaider after integer division*

## Operator Precedence

## Slicing (needed at least one column :)

print(parrot[10:])

### Using Step in a Slice

number = **"9,223;372:036 854,775;807"***# print(number[1::4])*separators = number[1::4]  
print(separators)  
values = **""**.join(char **if** char **not in** separators **else " " for** char **in** number).split()  
print([int(val) **for** val **in** values])

### Slicing backwards

letters = **"abcdefghijklmnopqrstuvwxyz"**backwards = letters[::-1]

## Python String Types (f format in use Python >= 3.6)

*print(f”{value} is greater”) – print(“{} is greater”.format(value))*

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

# Section 4

## Using debugger

## Conditional Operators

## Simplify Chained Comparison

*if age >= 16 and age <= 65:***if** 16 <= age <= 65:

## Boolean expressions (True or False with capital)

### Operator Precedence Table

* <https://docs.python.org/3/reference/expressions.html#operator-precedence>

## String methods

* <https://docs.python.org/3/library/stdtypes.html#string-methods>

## For loops

## Python Build-in Functions

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

## Continue

* <https://stackoverflow.com/questions/8420705/example-use-of-continue-statement-in-python>

## Break

## While

## Binary Search

low + (high – low)/2

## Augmented Assignment

* guessed += 1
* <https://docs.python.org/3/reference/expressions.html>

## Pep 8

* <https://www.python.org/dev/peps/pep-0008/>
* <https://www.python.org/dev/peps/pep-0013/>

# Section 5 – List and Tuples (sequence types)

* <https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range>

## Iterables have either:

\_\_iter\_\_ method

\_\_getitem\_\_ method

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

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

## Mutable objects

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

* <https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types>

### Methods and Function

* <https://docs.python.org/3/library/stdtypes.html#string-methods>

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

#### dot notation

s.append(x)

## Enumerate

**for** index, character **in** enumerate(**"abcdefgh"**):  
 print(index, character)

## Sorting

even = [2,4,6,8]  
odd = [1,3,5,7,9]  
  
even.extend(odd)  
print(even)  
another\_even = even  
print(another\_even)  
  
even.sort(reverse=**True**)  
print(even)  
print(another\_even)

## Built-in-Functions

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

## Sorting

## Case sensitive sorting

* word.casefold()

## Creating Lists (about 20 ways to do it)

* <https://stackoverflow.com/questions/2612802/list-changes-unexpectedly-after-assignment-why-is-this-and-how-can-i-prevent-it/43220129#43220129>

## Replacing a slice

* <https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types>

## Deleting items from a list

This doesn’t work in Python as in Java or C, you can’t handle loop control variable

data = [4,5,104,105,110,120,130,130,150,160,170,183,185,187,188,191,350,360]min\_valid = 100  
max\_valid = 200  
  
**for** index, value **in** enumerate(data):  
 **if** (value < min\_valid) **or** (value > max\_valid):  
 **del** data[index]  
  
print(data)

## Safely removing items from a list

## Testing

* <https://en.wikipedia.org/wiki/Edge_case>
* <https://en.wikipedia.org/wiki/Corner_case>

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

## Removing items from list – backwards (Lesson 108)

data = [104,101,4,105,308,103,5,107,100,306,106,102,108]  
  
min\_valid = 100  
max\_valid = 200  
  
**for** index **in** range(len(data) -1, -1, -1): *# backward, step -1* **if** data[index] < min\_valid **or** data[index] > max\_valid:  
 print(index,data)  
 **del** data[index]  
  
print(**'-'**\* 30)  
print(data)

## The Reversed Function <https://docs.python.org/3/library/functions.html>

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

## Algorithms Performance

## Pep8 – style guide

* <https://www.python.org/dev/peps/pep-0008/>

### Trailing commas

my\_list = [

1, 2, 3,

4, 5, 6,

]

## Processing Nested Lists

### No spam with menu-list (2 different approaches)

*# First workin example***for** meal **in** menu:  
 **for** index **in** range(len(meal) -1, -1, -1):  
 **if** meal[index] == **"spam"**:  
 **del** meal[index]  
  
 print(meal)  
  
print(**'-'**\*30)  
  
*# Second workin example***for** meal **in** menu:  
 **for** item **in** meal:  
 **if** item != **"spam"**:  
 print(item,end=**" "**)  
 print()

## Print revisited

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

Kuva, joka sisältää kohteen teksti, sisä, näyttökuva

Kuvaus luotu automaattisesti

## The join method (lists)

* https://docs.python.org/3/tutorial/datastructures.html?highlight=lists
* flowers = [  
   **"Daffodil"**,  
   **"Evening Primose"**,  
   **"Hydrangea"**,  
   **"Iris"**,  
   **"Lavender"**,  
   **"Sunflower"**,  
   **"Tiger Lily"**,  
  ]  
    
  *# for flower in flowers:  
  # print(flower)*separator = **", "**output = separator.join(flowers)  
  print(output)  
    
  print(**","**.join(flowers))

## The split method

generated\_list = [**'9'**,**' '**,  
 **'2'**,**'2'**,**'3'**,**' '**,  
 **'3'**,**'7'**,**'2'**,**' '**,  
 **'0'**,**'3'**,**'6'**,**' '**,  
 **'8'**,**'5'**,**'4'**,**' '**,  
 **'7'**,**'7'**,**'5'**,**' '**,  
 **'8'**,**'0'**,**'7'**]  
values = **""**.join(generated\_list)  
print(values)  
  
values\_list = values.split()  
print(values\_list)

*# replace the values in place***for** index **in** range(len(values\_list)):  
 values\_list[index] = int(values\_list[index])  
print(values\_list)  
  
  
*# create a new list*integer\_values = []  
**for** value **in** values\_list:  
 integer\_values.append(int(value))  
print(integer\_values)

## Tuples are immutable

* <https://docs.python.org/3/library/stdtypes.html#common-sequence-operations>
* <https://docs.python.org/3/library/functions.html>

## Unpacking Tuples

albums = [(**"Welcome to my Nightmare"**, **"Alice Cooper"**, 1975),  
 (**"Bad Company"**, **"Bad Company"**, 1974),  
 (**"Nightflight"**, **"Budgie"**, 1981),  
 (**"More Mayhem"**, **"Emilda May"**, 2011),  
 (**"Ride the Lightning"**, **"Metallica"**, 1984),  
 ]  
  
print(len(albums))  
  
**for** album **in** albums:  
 print(**"Album: {}, Artist: {}, Year: {}"** .format(album[0], album[1], album[2]))  
  
print(**'-'** \* 30)  
  
**for** name, artist, year **in** albums:  
 print(**"Album: {}, Artist: {}, Year: {}"** .format(name, artist, year))  
  
print(**'-'** \* 30)  
  
**for** album **in** albums:  
 print(**f"Album: {album[0]}, Artist: {album[1]}, Year: {album[2]}"**)

## Nesting Further

## Constants

SONGS\_LIST = 3

**from** nested\_data **import** albums

# Section 6 - Functions – introduction

## Palindromes

Do gees see god? - true

desnes not far, rafton sensed – true

### String functions

* <https://docs.python.org/3/library/stdtypes.html#string-methods>

### Range exercise 16 - sum(range(start, n, 2))

**def** sum\_eo\_Tim(n, t):  
 *"""* **if** t == **"e"**:  
 start = 2  
 **elif** t == **'o'**:  
 start = 1  
 **else**:  
 **return** -1  
  
 **return** sum(range(start, n, 2))  
  
  
x = sum\_eo\_Tim(10, **'e'**)  
print(x)

## Handling invalid arguments (positional-or-keyword)

* <https://docs.python.org/3/library/exceptions.html>
* <https://docs.python.org/3/glossary.html#term-parameter>

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

## Docstrings (using reStructuredText)

* <https://www.python.org/dev/peps/pep-0257/>

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

help(get\_integer)  
  
**'''  
print(input.\_\_doc\_\_)  
print("\*" \* 80)  
print(get\_integer.\_\_doc\_\_)  
print("\*" \* 80)  
'''**

## Fibonacci numbers

0,1,1,2,3,5,8,13,21,34,55

* <https://en.wikipedia.org/wiki/Fibonacci_number>

### Documenting

* <https://devguide.python.org/documenting/>

## Function annotations and type hints

**def** is\_palindrome(string: str) -> bool:

* <https://docs.python.org/3/library/typing.html>
* <https://www.python.org/dev/peps/pep-3107/>

## Function annotations with default value

* <https://www.python.org/dev/peps/pep-0008/> (CTRL + F on the page)

When combining an argument annotation with a default value, however, do use spaces around the = sign

## Running program on terminal (msdos / powershell)

* file open in terminal (mouse right click) and get the path
* python -V

installing colorama with pip to PyCharm: (from the path where you saved the python wheel file)

C:\Users\lauri\PycharmProjects\PythonMasterClass\Section6>pip install C:\Users\lauri\PycharmProjects\colorama\_lpa-0.4.4b1.0-py2.py3-none-any.whl

Processing c:\users\lauri\pycharmprojects\colorama\_lpa-0.4.4b1.0-py2.py3-none-any.whl

Installing collected packages: colorama-lpa

Successfully installed colorama-lpa-0.4.4b1.

Settings/Project/Project Interpreter

see the link below and lesson 168 startin at 4 minutes forward (we already intalled colorama with pip on PyCharm / Intellij)

* <https://www.udemy.com/course/python-the-complete-python-developer-course/learn/lecture/21763350#questions/16025702>
* C:\Users\lauri\PycharmProjects\PythonMasterClass\Section6
* C:\Users\lauri\AppData\Local\Programs\Python\Python36-32\

## Playing Fizz Buzz -game

## Defining different argument types

**def** sum\_numbers(\*num: float) -> float:  
 *"""  
 Count the sum of numbers* **:param** *num: integer/float* **:return***: sum of numbers  
 """* sum = 0  
  
 **for** n **in** num:  
 sum = sum + n  
  
 **return** sum  
  
  
print(sum\_numbers(12.5,3.147,98.1))

# Section 7

## What is a dictionary

vehicles = {  
 **'dream'**: **'Honda 250T'**,  
 **'er5'**: **'Kawasaki ER5'**,  
 **'can-am'**: **'Bombardier Can-Am 250'**,  
 **'virago'**: **'Yamaha XV250'**,  
 **'tenere'**: **'Yamaha XT650'**,  
 **'jimny'**: **'Suzuki Jimny 1.5'**,  
 **'fiesta'**: **'Ford Fiesta Ghia 1.4'**,  
 **'roadster'**: **'Triuph Street Triple'**}  
**'''  
my\_car = vehicles['fiesta']  
print(my\_car)  
  
learner = vehicles.get("er5")  
print(learner)  
'''**vehicles[**"starfighter"**] = **"Lockheed F-104"**vehicles[**"learjet"**] = **"Bombardier Learjet 75"**vehicles[**"toy"**] = **"glider"***# Upgrade the Virago*vehicles[**"virago"**] = **"Yamaha XV535"  
  
del** vehicles[**"starfighter"**]  
  
*# for key in vehicles:  
# print(key, vehicles[key], sep=", ")***for** key, value **in** vehicles.items():  
 print(key, value, sep=**", "**)

## Zen of Python

* <https://zen-of-python.info/>

## Using in with dictionary

available\_parts = {**"1"**:**"computer"**,  
 **"2"**:**"monitor"**,  
 **"3"**:**"keyboard"**,  
 **"4"**:**"mouse"**,  
 **"5"**:**"hdmi cable"**,  
 **"6"**:**"dvd drive"**,  
 }  
  
  
current\_choice = **None  
while** current\_choice != **'0'**:  
 **if** current\_choice **in** available\_parts:  
 chosen\_part = available\_parts[current\_choice]  
 print(**f"Adding {chosen\_part}"**)  
  
 current\_choice = input(**"> "**)

## Checking quantities – tuple or dict

recipes\_tuple = {  
 **"Chicken and chips"**: [  
 (**"chicken"**, 100),  
 (**"potatoes"**, 3),  
 (**"salt"**, 1),  
 (**"malt vinegar"**, 5),  
 ],  
}  
  
recipes\_dict = {  
 **"Chicken and chips"**: {  
 **"chicken"**: 100,  
 **"potatoes"**: 3,  
 **"salt"**: 1,  
 **"malt vinegar"**: 5,  
 },  
}  
  
*# using tuples***for** recipe, ingredients **in** recipes\_tuple.items():  
 print(**f"Ingredients for {recipe}"**)  
 **for** ingredient, quantity **in** ingredients: *# ingredients is a tuple* print(ingredient, quantity, sep=**', '**)  
  
print()  
  
*# using a dictionary***for** recipe, ingredients **in** recipes\_dict.items():  
 print(**f"Ingredients for {recipe}"**)  
 **for** ingredient, quantity **in** ingredients.items(): *# ingredients is a dict* print(ingredient, quantity, sep=**', '**)

## setdefault -method

**from** contents **import** pantry  
  
chicken\_quantity = pantry.setdefault(**"chicken"**, 0) *# vrt get()*print(**f"chicken: {chicken\_quantity}"**)  
  
beans\_quantity = pantry.setdefault(**"beans, 0"**)  
print(**f"beans: {beans\_quantity}"**)  
  
  
ketchup\_quantity = pantry.get(**"ketchup"**, 0) *# is not added*print(**f"ketchup: {ketchup\_quantity}"**)  
  
z\_quantity = pantry.setdefault(**"zucchini"**, **"eight"**) *# is added to dict*print(**f"zucchini: {z\_quantity}"**)  
  
print()  
print(**"`pantry` now contains..."**)  
  
**for** key, value **in** sorted(pantry.items()):  
 print(key, value)

## Exercise 20 – char counter

*# We need an empty dictionary, to store and display the letter frequencies.*word\_count = {}  
  
*# Text string*text = **"Later in the course, you'll see how to use the collections Counter class."***# Iterate over every character in the string.***for** char **in** text.casefold():  
 *# We're only counting letters and digits (no punctuation).* **if** char.isalnum():  
 **if** char **in** word\_count:  
 word\_count[char] += 1  
 **else**:  
 word\_count[char] = 1  
  
*# Printing the dictionary***for** letter, count **in** sorted(word\_count.items()):  
 print(letter, count)

## APIs and mobile phone demo

## - <https://docs.python.org/3/library/getpass.html>

- <https://github.com/googleapis/google-api-python-client/blob/main/docs/oauth-installed.md>

## The dict documentation

* <https://docs.python.org/3/library/stdtypes.html#mapping-types-dict>
* <https://docs.python.org/3/glossary.html#term-hashable>

**>>>** a = dict(one=1, two=2, three=3)

**>>>** b = {'one': 1, 'two': 2, 'three': 3}

**>>>** c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))

**>>>** d = dict([('two', 2), ('one', 1), ('three', 3)])

**>>>** e = dict({'three': 3, 'one': 1, 'two': 2})

**>>>** f = dict({'one': 1, 'three': 3}, two=2)

**>>>** a == b == c == d == e == f

True

## Python Branches

* <https://devguide.python.org/#status-of-python-branches>

## dict objects

* <https://docs.python.org/3/library/stdtypes.html#dict-views>

## Shallow copy Lesson 205 copies only reference

## Hash functions

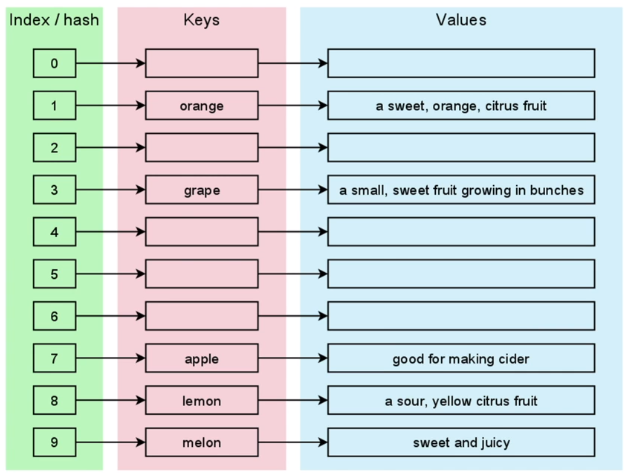
* <https://en.wikipedia.org/wiki/Hash_function>
* <https://docs.python.org/3/glossary.html#term-hashable>

### Tools – Python Console (console woindow)

Tools

Python console

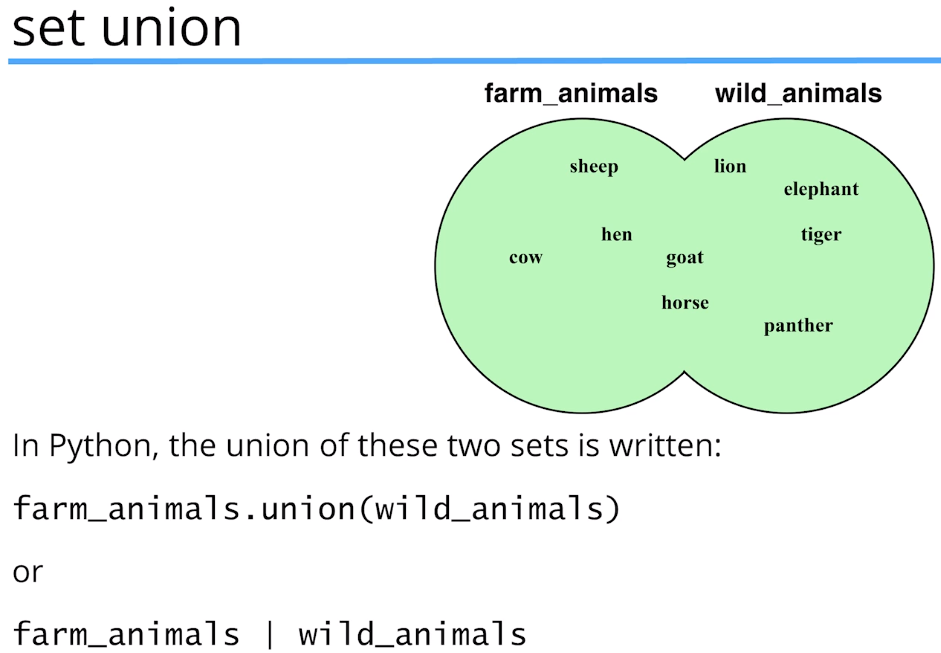
## Hash tables

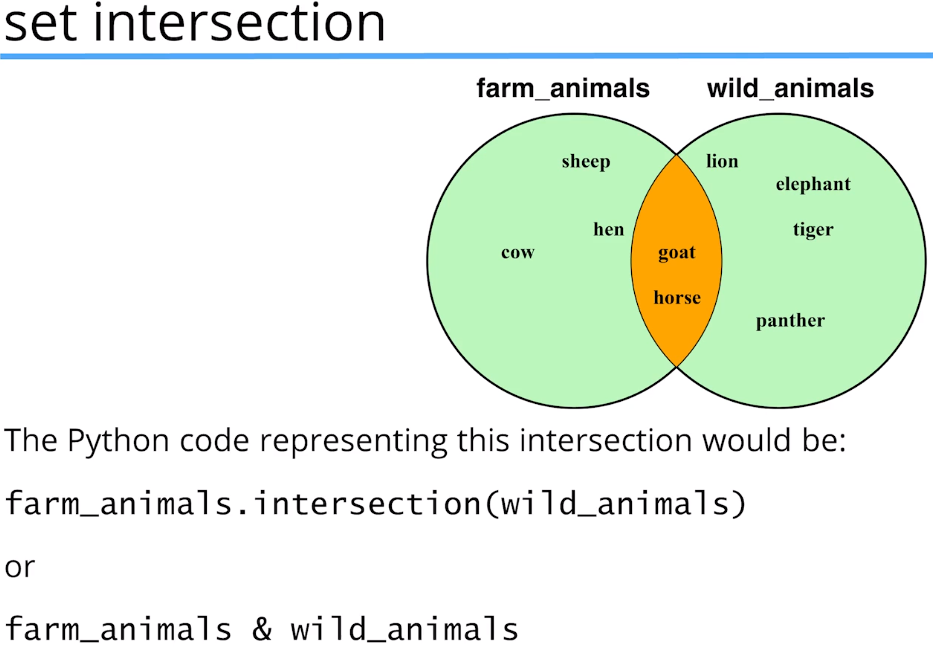


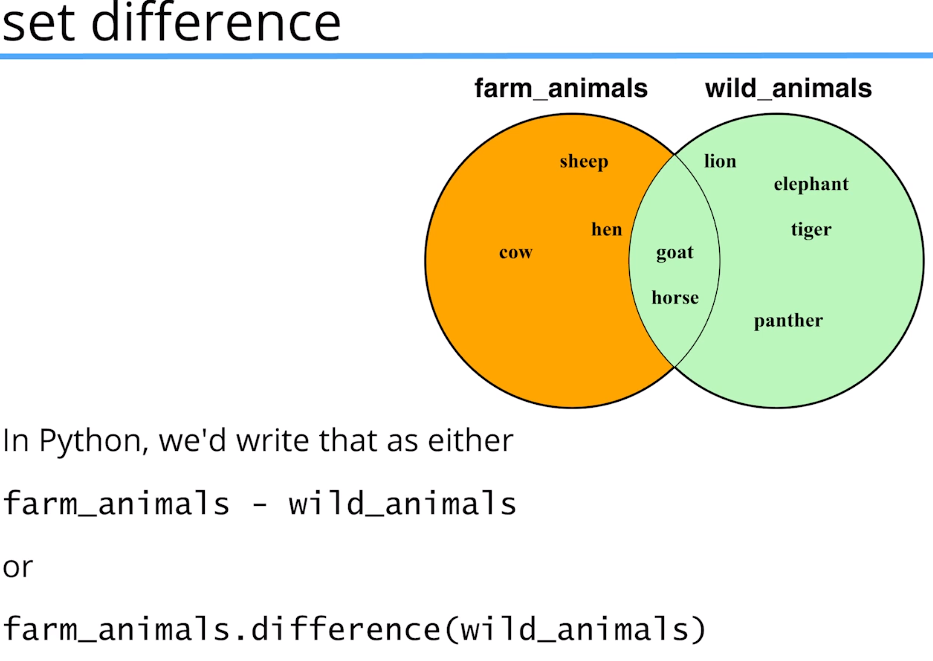
## hashlib

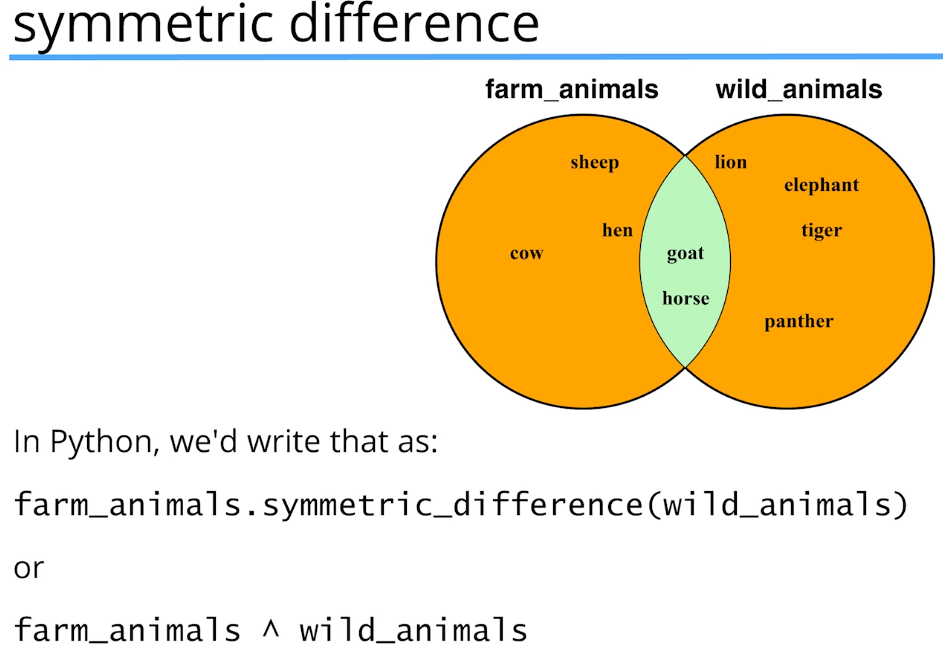
* <https://docs.python.org/3/library/hashlib.html>

## Introduction to sets









Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

## set membership

choice = **"-"** *# initialise choice to something invalid***while** choice != **"0"**:  
 *# if choice in set("12345"):* **if** choice **in** {**"1"**,**"2"**,**"3"**,**"4"**,**"4"**}:  
 print(**"You chose {}"**.format(choice))  
 **else**:  
 print(**"Please choose your option from the list below:"**)  
 print(**"1:\tLearn Python"**)  
 print(**"2:\tLearn Java"**)  
 print(**"3:\tGo swimming"**)  
 print(**"4:\tHave dinner"**)  
 print(**"5:\tGo to bed"**)  
 print(**"0:\tExit"**)  
  
 choice = input()

## Better performance

*# if choice in set("12345"):***if** choice **in** {**"1"**,**"2"**,**"3"**,**"4"**,**"4"**}:

choice = **"-"** *# initialise choice to something invalid***while** choice != **"0"**:  
 *# if choice in "12345": Here's a BUG  
 # if choice in set("12345"):* **if** choice **in** {**"1"**,**"2"**,**"3"**,**"4"**,**"4"**}:  
 print(**"You chose {}"**.format(choice))  
 **else**:  
 print(**"Please choose your option from the list below:"**)  
 print(**"1:\tLearn Python"**)  
 print(**"2:\tLearn Java"**)  
 print(**"3:\tGo swimming"**)  
 print(**"4:\tHave dinner"**)  
 print(**"5:\tGo to bed"**)  
 print(**"0:\tExit"**)  
  
 choice = input()

Still better:

Kuva, joka sisältää kohteen pöytä

Kuvaus luotu automaattisesti

## Unique data in sets

data = [**"blue"**,**"red"**,**"blue"**,**"green"**,**"red"**,**"blue"**,**"white"**]  
  
*# Create a set from the list to remove duplicates*unique\_data = set(data)  
print(unique\_data)

## Set documentation

* <https://docs.python.org/3/library/stdtypes.html#set>

## The `pop` method

* <https://stackoverflow.com/questions/19378718/python-whats-the-use-case-for-set-pop>

## Set union

* <https://docs.python.org/3/library/stdtypes.html#set>

Clear() remove all elements from the set

Note, the non-operator versions of the [update()](https://docs.python.org/3/library/stdtypes.html#frozenset.update), [intersection\_update()](https://docs.python.org/3/library/stdtypes.html#frozenset.intersection_update), [difference\_update()](https://docs.python.org/3/library/stdtypes.html#frozenset.difference_update), and [symmetric\_difference\_update()](https://docs.python.org/3/library/stdtypes.html#frozenset.symmetric_difference_update) methods will accept any iterable as an argument.

**update**(*\*others*)

**set |= other | ...**

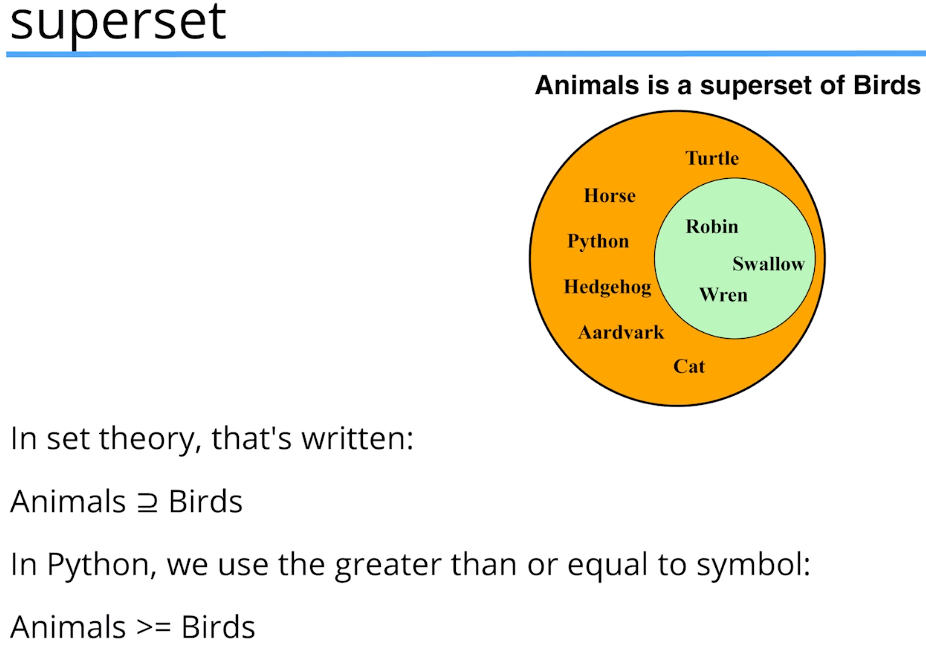
Update the set, adding elements from all others.

**from** prescription\_data **import** adverse\_interactions  
  
meds\_to\_watch = set()  
  
**for** interaction **in** adverse\_interactions:  
 *# meds\_to\_watch = meds\_to\_watch.union(interaction)  
 # meds\_to\_watch = meds\_to\_watch | interaction  
 # meds\_to\_watch.update(interaction)* meds\_to\_watch |= interaction *# update*print(sorted(meds\_to\_watch))

## Symmetric Difference (you could use list)

*'''  
morning = {'Java', 'C', 'Ruby', 'Lisp', 'C#'}  
afternoon = {'Python', 'C#', 'Java', 'C', 'Ruby'}  
  
possible\_courses = morning ^ afternoon  
print(possible\_courses)  
'''*morning = [**'Java'**, **'C'**, **'Ruby'**, **'Lisp'**, **'C#'**]  
afternoon = [**'Python'**, **'C#'**, **'Java'**, **'C'**, **'Ruby'**]  
  
possible\_courses = set(morning).symmetric\_difference(afternoon)  
print(possible\_courses)  
  
possible\_courses = set(afternoon).symmetric\_difference((morning))  
print(possible\_courses)

## Subsets and supersets



## isdisjoint(other)

**isdisjoint**(*other*)

Return True if the set has no elements in common with *other*. Sets are disjoint if and only if their intersection is the empty set.

## Summary

* <https://www.udemy.com/course/python-the-complete-python-developer-course/learn/lecture/27777626#questions>

# Section 8 Input and Output I/O

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

jabber = open(**"sample.txt"**,**"r"**)  
*#jabber = open("C:\\Users\\lauri\\Downloads\\sample.txt","r")***for** line **in** jabber:  
 **if "jabberwock" in** line.lower():  
 print(line,end=**""**)  
  
jabber.close()

## Pickle

* <https://docs.python.org/3/library/pickle.html>
* pickle version 4 with Python 3.4 is *incompatible* downwards
* pickle.dump(imelda, pickle\_file, protocol=0)

## Shelve

# Section 9 Modules and Functions

## Modules and import

**from** turtle **import** forward, right, done, circle  
  
done = **"Well done, you have finished your drawing!"**forward(150)  
right(250)  
circle(75)  
forward(150)  
  
done()  
print(done)

## The Standard Python Library

* <https://docs.python.org/3/library/functions.html#built-in-funcs>
* <https://docs.python.org/3.5/library/shelve.html>

## Webrowser Module

**import** webbrowser  
webbrowser.open(**"https://www.python.org"**)

<https://docs.python.org/3.4/library/webbrowser.html>

example: python -m webbrowser -t "http://www.python.org"

Example:

**import** webbrowser

url = **'http://docs.python.org/'***# Open URL in a new tab, if a browser window is already open.*webbrowser.open\_new\_tab(url)  
  
*# Open URL in new window, raising the window if possible.  
# webbrowser.open\_new(url)*

## Time and Datetime in Python

* <https://docs.python.org/3/library/time.html>
* strftime
* <https://peps.python.org/pep-0418/>
* time.**strftime**(format[, t])
* <https://docs.python.org/3/library/datetime.html#module-datetime>
* *class*datetime.**tzinfo**

An abstract base class for time zone information objects. These are used by the [datetime](https://docs.python.org/3/library/datetime.html#datetime.datetime) and [time](https://docs.python.org/3/library/datetime.html#datetime.time) classes to provide a customizable notion of time adjustment (for example, to account for time zone and/or daylight saving time).

* <https://www.youtube.com/watch?v=-5wpm-gesOY> (The problem with Time & …)
* <https://mail.python.org/pipermail/python-dev/2002-March/020604.html>

## Installing pytz module (FAQ)

* <https://www.udemy.com/course/python-the-complete-python-developer-course/learn/lecture/13986636#questions>

## Installing pytz

c: > pip3 install pytz (as Admin)

C:\Users\lauri\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip

## Using Timezones

* <http://www.iana.org/time-zones>
* <http://pytz.sourceforge.net/>

## Tkinter

* <https://docs.python.org/3/library/tk.html>
* <https://tkdocs.com/>
* <https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html>
* <https://en.wikipedia.org/wiki/Tk_%28software%29>

## Import System

* <https://docs.python.org/3/reference/import.html>

blackjack.py &

import\_test.py

**import** blackjack  
  
*# from command line: python -m blackjack*print(\_\_name\_\_)  
blackjack.play()  
print(blackjack.cards)

## Lesson 288 recursive or iterative

### Factorial and Fibonacci

**def** fact(n):  
 *""" calculate n! iteratively"""* result = 1  
 **if** n > 1:  
 **for** f **in** range(2, n + 1):  
 result \*= f  
 **return** result  
  
**def** factorial(n):  
 *# n! can also be defined as n \* (n-1)!* **if** n <= 1:  
 **return** 1  
 **else**:  
 **return** n \* factorial(n-1)  
  
**def** fib(n):  
 *""" F(n) = F(n - 1) ? F(n - 2) """* **if** n < 2:  
 **return** n  
 **else**:  
 **return** fib(n-1) + fib(n-2)  
  
**def** fibonacci(n):  
 **if** n == 0:  
 result = 0  
 **elif** n == 1:  
 result = 1  
 **else**:  
 n\_minus\_1 = 1  
 n\_minus\_2 = 0  
 **for** f **in** range(1, n):  
 result = n\_minus\_2 + n\_minus\_1  
 n\_minus\_2 = n\_minus\_1  
 n\_minus\_1 = result  
 **return** result  
  
**for** i **in** range(36):  
 print(i, fib(i), **"\t"**, fibonacci(i))

## OS module

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

## Nonlocal keyword, FREE and LEGB

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

# Section 10 – Object Oriented Programming

*Class: template for creating objects. All objects created using the same class will have the same characteristics.*

*Object: an instance of a class.*

*Instantiate: create an instance of a class.*

*Method: a function defined in a class.*

*Attribute: a variable bound to an instance of a class.*

### Capsulation in Classes check from material

Example:

**class** Account:  
 *""" Simple account class with balance """* **def** \_\_init\_\_(self, name, balance):  
 self.name = name  
 self.balance = balance  
 print(**"Account created for "** + self.name)  
  
 **def** deposit(self, amount):  
 **if** amount > 0:  
 self.balance += amount  
 self.show\_balance()  
  
 **def** withdraw(self, amount):  
 **if** amount > 0 **and** amount < self.balance:  
 self.balance -= amount  
 **else**:  
 print(**"Failed to withdraw - balance on your account must be more than the withdraw amount"**)  
 self.show\_balance()  
  
 **def** show\_balance(self):  
 print(**f"Balance is {self.balance} €"**)  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 lasse = Account(**"Lasse"**,0)  
 lasse.show\_balance()  
  
 lasse.deposit(1000)  
 lasse.withdraw(200)  
 lasse.deposit(100)

### Static Methods:

As far as i understand it's used without using "self" and it's being shared on all the class methods?

That's about all there is to it.  The attribute belongs to the class, and you can call it without creating a class instance - in fact, you shouldn't use an instance to call it.

Generally, you'd use a static method for something that doesn't use any of the instance attributes but that may be useful to users of the class.

### Namespaces:

In Python, everything is an object, and names (variable names), also referred to as identifiers, are labels associated with those objects. One of the main ways of accessing an object is via its name.

A namespace is an isolated collection of names mapped to their corresponding objects. Different objects can have the same name and not collide as long as those names are in different namespaces.

Every module creates its own namespace. Each class has its own namespace, and a function creates its own namespace each time it is run.

Namespaces are related to scopes. Each scope has a corresponding namespace, and can be defined as the section of the program where you can access that namespace without any prefixes.

## DocString and Raw Literals

<https://peps.python.org/pep-0257/>

## Artist Class and import Albums (with circular reference)



## Compare Files and Algorithm Flowcharts

1. select files to be compared + CTRL + D or View + Compare Files:
   * opens a new window
2. select one file + CTRL + D opens files directory to select the other file to be combared

### If using Python 2 in song.py

this line wouldn’t work:

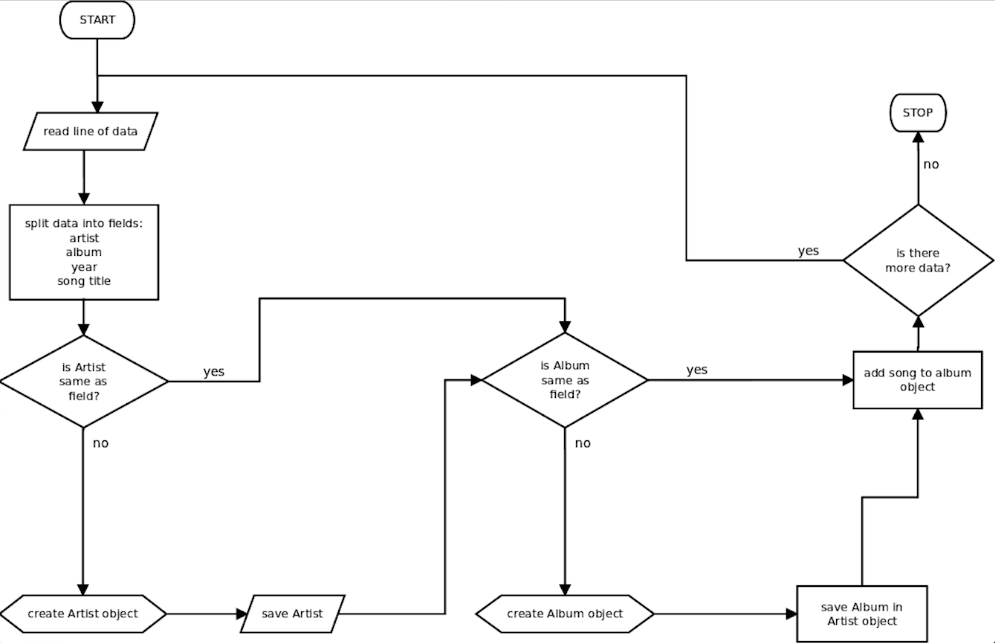
print(**"{0.name}\t{1.name}\t{1.year}\t{2.title}"**.format(new\_artist, new\_album, new\_song),file=checkfile)

But using import as following it should work

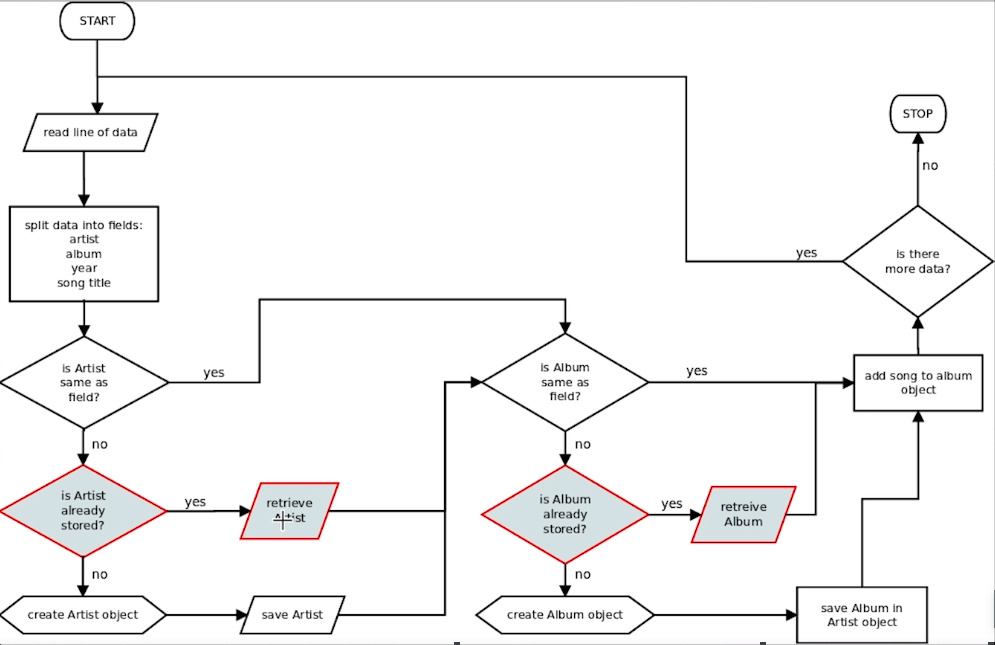
import from \_\_future\_\_ import print\_function



New one



Latest one



## Challenge – remove circular references (aim to top diagram)

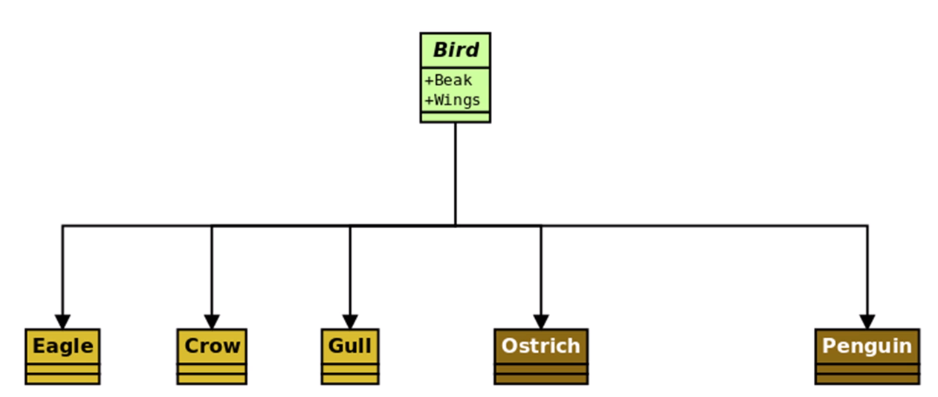
Kuva, joka sisältää kohteen teksti

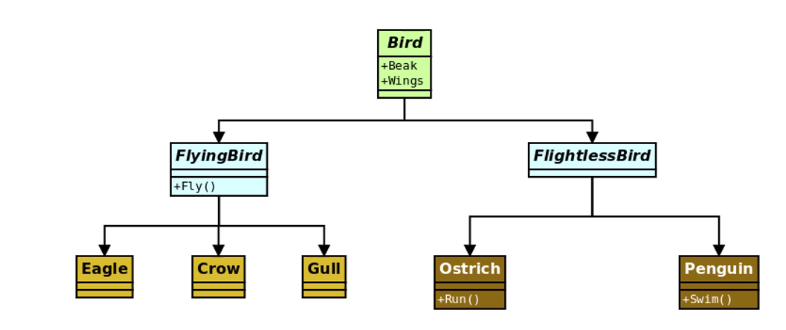
Kuvaus luotu automaattisesti

## Getters and Setters

* <https://docs.python.org/3.5/library/functions.html#property>

## Inheritance (Python allows multiple inheritance but be aware that you should master it)





### Python no method overloading

### Python overriding

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

### tab Regex + (\{\d\.) + $1\_ Replace All

### Inheritance Challenge Lesson 314

**import** random  
  
*# class Enemy:***class** Enemy(object): *# with Python 2, with Python 3 either is Ok* **def** \_\_init\_\_(self, name=**"Enemy"**, hit\_points=0, lives=1):  
 self.\_name = name  
 self.\_hit\_points = hit\_points  
 self.\_lives = lives  
 self.\_alive = **True  
  
 def** take\_damage(self, damage):  
 remaining\_points = self.\_hit\_points - damage  
 **if** remaining\_points >= 0:  
 self.\_hit\_points = remaining\_points  
 print(**"I took {} points damage and have {} left"**.format(damage, self.\_hit\_points))  
 **else**:  
 self.\_lives -= 1  
 **if** self.\_lives > 0:  
 print(**"{0.\_name} lost a life"**.format(self))  
 **else**:  
 print(**"{0.\_name} is dead"**.format(self))  
 self.\_alive = **False  
  
 def** \_\_str\_\_(self):  
 **return "Name: {0.\_name}, Lives: {0.\_lives}, Hit points: {0.\_hit\_points}"**.format(self)  
  
  
**class** Troll(Enemy):  
  
 **def** \_\_init\_\_(self, name):  
 *# Enemy.\_\_init\_\_(self, name=name, hit\_points=23)  
 # super(Troll, self).\_\_init\_\_(name=name, lives=1, hit\_points=23)* super().\_\_init\_\_(name=name, lives=1, hit\_points=23)  
  
 **def** grunt(self):  
 print(**"Me {0.\_name}. {0.\_name} stopm you"**.format(self))  
  
**class** Vampyre(Enemy):  
  
 **def** \_\_init\_\_(self, name):  
 super().\_\_init\_\_(name=name, lives=3, hit\_points=12)  
  
 **def** dodges(self):  
 **if** random.randint(1,3) == 3:  
 print(**"\*\*\*\*\*\*\*\* {0.\_name} dodges \*\*\*\*\*\*\*\*"**.format(self))  
 **return True  
 else**:  
 **return False  
  
 def** take\_damage(self, damage):  
 **if not** self.dodges():  
 super().take\_damage(damage=damage)  
  
**class** VampyreKing(Vampyre):  
  
 **def** \_\_init\_\_(self,name):  
 super().\_\_init\_\_(name=name)  
 self.\_hit\_points = 140  
  
 **def** take\_damage(self, damage):  
 **if not** self.dodges():  
 super().take\_damage(damage//4)

## Polymorphism (Java in statically typed – Python dynamically typed)

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

### Inheritance isn’t the only way to implement polymorphism Lesson316

* <https://en.wikipedia.org/wiki/Duck_test>

## Composition Lesson 317 & 319 HTML document demo

* <https://www.w3.org/TR/html401/struct/global.html>

### Relationship described with

IS a -> inheritance

Has a -> composition

## Aggregation (weak form of composition) Lesson 320

# Section 11 – Using Databases with Python

## Introduction to SQLite

* <https://sqlite.org/index.html>

1. added to path C:\Users\sqlite-tools-win32-x86-3380100
2. c:> sqlite3
3. sqlite> .quit

sqlite3 test.db

.help

.headers on

create table contacts (name text, phone integer, email text);

insert into contacts (name, phone, email) values('Lasse', 05009898989, 'lasse@gmail.com');

insert into contacts(name, phone) values("Steve", 04599888); # also succeeds in sqlite3

if session closed reopen:

*.open test.db* OR just in start: *sqlite3 music.db*

SELECT \* FROM contacts; or use graphical DB browser for sqlite

.backup testbackup

.restore ***How to use?***

update contacts set email="brian@gmail.com" where name ="Brian";

delete from contacts where name="Brian";

.tables

.schema

.dump

## Querying data with SQLite

copied music.db to -> c:/users/lauri

.schema

.headers on

select \* from albums where \_id="367";

### Autoincrement in SQLite

* <https://www.sqlite.org/autoinc.html>

select \* from artists order by name;

select \* from artists order by name collate nocase;

select \* from artists order by name desc; # OR asc

select \* from albums order by artist, name collate nocase;

select \* from songs order by album, track;

select \* from albums where \_id=439;

select \* from artists where \_id=133;

Join:

* select songs.track, songs.title, albums.name FROM songs JOIN albums ON songs.album = albums.\_id;
* select track, title, name FROM songs JOIN albums ON songs.album = albums.\_id;

Kuva, joka sisältää kohteen pöytä

Kuvaus luotu automaattisesti

## More Complex Joins

INNER JOIN

* select songs.track, songs.title, albums.name FROM songs INNER JOIN albums ON songs.album = albums.\_id;
* select songs.track, songs.title, albums.name FROM songs INNER JOIN albums ON songs.album = albums.\_id ORDER BY albums.name, songs.track;

select albums.name, songs.track, songs.title FROM songs INNER JOIN albums ON songs.album = albums.\_id ORDER BY albums.name, songs.track;

select albums.name, artists.name from albums inner join artists on albums.artist=artists.\_id order by artists.name;

select artists.name, albums.name from albums inner join artists on albums.artist=artists.\_id order by artists.name;

select artists.name, albums.name from albums inner join artists on albums.artist=artists.\_id where artists.name = “Alice Cooper”;

select artists.name, albums.name, songs.track, songs.title FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

ORDER BY artists.name, albums.name, songs.track;

select artists.name, albums.name from albums inner join artists on albums.artist=artists.\_id where albums.name = “Doolitle”;

## Wildcards and Views

select artists.name, albums.name, songs.track, songs.title FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

WHERE songs.title LIKE "%doctor%"

ORDER BY artists.name, albums.name, songs.track;

select artists.name, albums.name, songs.track, songs.title FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

WHERE artists.name LIKE "jefferson%"

ORDER BY artists.name, albums.name, songs.track;

### Views (for customer security plus to easy job)

CREATE VIEW artist\_list AS

SELECT artists.name, albums.name, songs.track, songs.title FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

ORDER BY artists.name, albums.name, songs.track;

select \* from artist\_list where name LIKE "jefferson%";

CREATE VIEW album\_list AS

SELECT name FROM albums

ORDER by NAME;

DROP VIEW album\_list;

CREATE VIEW album\_list AS

SELECT name FROM albums

ORDER by NAME COLLATE NOCASE;

select \* from artist\_list where name like "jefferson%";

.headers on

GIVES two name-fields on output:

sqlite> select \* from artist\_list where name like "jefferson%";

name|name:1|track|title

Jefferson Airplane|Surrealistic Pillow|1|She Has Funny Cars

Jefferson Airplane|Surrealistic Pillow|2|Somebody To Love

DROP VIEW album\_list;

CREATE VIEW album\_list AS

SELECT artists.name AS artist, albums.name AS album, songs.track, songs.title from songs

INNER JOIN albums ON songs.album = albums.\_id

ORDER by artists.name, albums.name, songs.track;

select \* from artist\_list where name like "jefferson%";

HEADERS WERE STILL THE SAME

## Housekeeping and Challenge

delete from songs where track < 50;

select \* from songs where track <> 71;

select count(\*) from songs;

## SQL Challenge

Kuva, joka sisältää kohteen teksti, sisä, näyttökuva

Kuvaus luotu automaattisesti

Kuva, joka sisältää kohteen teksti, sisä, näyttökuva, paperi

Kuvaus luotu automaattisesti

1. select songs.track, songs.title, albums.name FROM songs INNER JOIN albums ON songs.album = albums.\_id where albums.name = "Forbidden";
2. select songs.track, songs.title, albums.name FROM songs INNER JOIN albums ON songs.album = albums.\_id where albums.name = "Forbidden" ORDER BY albums.name, songs.track;

select artists.name, songs.track, songs.title FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

WHERE artists.name = "Deep Purple"

ORDER BY artists.name, albums.name, songs.track;

OR

select \* from artist\_list where name = “Deep Purple”;

1. update artists set name = "One Kitten" where name = "Mehitabel";
2. Ok – select \* from artists where artists.name = "One Kitten";

6.

select songs.title FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

WHERE artists.name = "Aerosmith"

ORDER BY songs.title COLLATE NOCASE;

OR

select title from artist\_list where name = “Aerosmith” order by title;

7.

select count(title) FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

WHERE artists.name = "Aerosmith"

ORDER BY songs.title;

8.

select distinct songs.title FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

WHERE artists.name = "Aerosmith"

ORDER BY songs.title COLLATE NOCASE;

9.

SELECT COUNT(DISTINCT songs.title) FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

WHERE artists.name = "Aerosmith"

ORDER BY songs.title COLLATE NOCASE;

10.

SELECT COUNT(DISTINCT artists.name) FROM songs

INNER JOIN albums ON songs.album = albums.\_id

INNER JOIN artists ON albums.artist = artists.\_id

WHERE artists.name = "Aerosmith"

ORDER BY songs.title COLLATE NOCASE;

## SQL in Python

## SQL Injection Attacks (parameter substitution with placeholders, sanitizing the input)

update\_sql = **"UPDATE contacts SET email = ? WHERE phone = ?"**print(update\_sql)

## Placeholders and Parameter Substitution

**import** sqlite3  
  
db = sqlite3.connect(**"contacts.sqlite"**)  
  
name = input(**"Please enter the name: "**)  
  
**for** row **in** db.execute(**"SELECT \* FROM contacts WHERE name LIKE ?"**, (name,)):  
 print(row)  
  
db.close()

## Exceptions

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

### Hints (CTRL + Q)

* <https://peps.python.org/pep-0484/>

## Adding Database code to Accounts Class

* <https://sqlite.org/datatype3.html>

C:\Users\lauri\PycharmProjects\PythonMasterClass\RollingBack\

## Displaying Time in Different Timezones

* <https://docs.python.org/3.5/library/sqlite3.html>
* sqlite3.**PARSE\_DECLTYPES**

## SQLite3 strftime Function

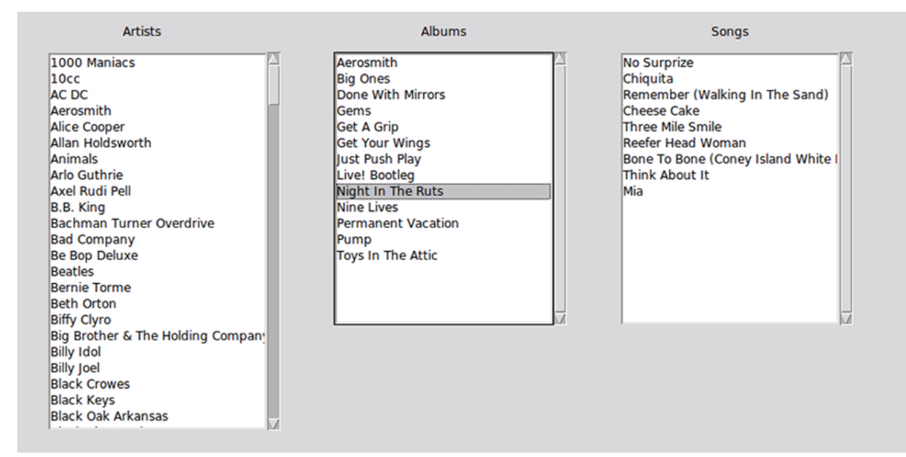
* <https://sqlite.org/lang_datefunc.html>

## Challenge and Problems Storing Time zones

## - <https://www.sqlite.org/lang_corefunc.html>

## - <https://www.sqlite.org/lang_aggfunc.html>

## Jukebox (Music Browser)



## Star args

in case of Python 2:

**from** \_\_future\_\_ **import** print\_function

---------------------------------------------------------------------------

**def** average(\*args):  
 print(type(args))  
 print(**"args is {}"**.format(args))  
 print(**"\*args is:"**, \*args)  
 mean = 0  
 **for** arg **in** args:  
 mean += arg  
 **return** mean / len(args)  
  
print(average(1,2,3,4))

**def** build\_tuple(\*args):  
 **return** (args)  
  
message\_tuple = build\_tuple(**"hello"**, **"planet"**, **"earth"**,**"take"**,**"me"**,**"to"**,**"your"**,**"leader"**)  
print(type(message\_tuple))  
print(message\_tuple)  
  
number\_tuple = build\_tuple(1,2,3,4,5,6)  
print(type(number\_tuple))  
print(number\_tuple)

example:

* reverse a string [::-1]
* <https://docs.python.org/3/tutorial/controlflow.html#more-on-defining-functions>

**def** return\_string(\*args):  
 **return** (args[::-1])  
  
message = return\_string(**"hello"**, **"planet"**, **"earth"**, **"take"**, **"me"**, **"to"**, **"your"**, **"leader"**)  
print(message)

**def** print\_backwards(\*args, file=**None**):  
 **for** word **in** args[::-1]:  
 print(word[::-1], end=**' '**, file=file)  
  
**with** open(**"backwards.txt"**,**"w"**) **as** backwards:  
 print\_backwards(**"hello"**,**"how"**,**"do"**,**"you"**,**"do"**, file=backwards)

# Jukebox final

C:\Users\lauri\PycharmProjects\PythonMasterClass\MusicBrowser

select albums.name, count(albums.name) num\_albums from albums group by albums.name having num\_albums >1;

select artists.\_id, artists.name, albums.name from artists

inner join albums on albums.artist = artists.\_id

where albums.name in

(select albums.name from albums group by albums.name having count(albums.name) > 1)

order by albums.name, artists.name;

SELECT name, \_id FROM albums WHERE name ='Greatest Hits';

SELECT name, \_id, artist FROM albums WHERE name ='Greatest Hits' AND artist = 176;

select \* from songs where album = 399;

Questions:

* <https://www.udemy.com/course/python-the-complete-python-developer-course/learn/lecture/23347020#questions/16829468>