Sisällys

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)

- Python variable names must begin with a letter (either upper or lower case) or an underscore _ character.
- They can contain letters, numbers or underscore characters (but cannot begin with a number).

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

Python Course by Tim Buchalka, started 28.1.2022

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

Python has several **built-in** data types, that can be classed as:

- **numeric**
- **iterator**
- **sequence (which are also iterators)**
- **mapping**
- **file**
- **class**
- **exception**

OBS! Int

I'll repeat that, for programmers who are coming to Python from other languages:
There's no limit to the size of the values that you can store in a Python **int**. You'll run out of memory before you exceed the size limit - because there isn't one.

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))

Python 3 has 5 sequence types built in:

- The string type
- list
- tuple
- range
- bytes and bytearray

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

When an object is described as **immutable**, that means it can't be changed.

The following **immutable** types are built into Python:

- int
- float
- bool (True and False) : a subclass of **int**
- str (string)
- tuple
- frozenset
- bytes

Mutable objects

A **mutable** object is one whose value can be changed.

Python has the following **mutable** objects built in:

- list
- dict
- set
- Bytearray

We'll be looking at dictionaries and sets in the next section.

We can change the value of mutable objects, without the object being destroyed and re-created.

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

Methods and Function

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

Methods and Functions

I used the term **method** there.

A method is the same as a function, except that it's bound to an object.

That means we need an object, in order to call the **method**.

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

Test Cases

So, what are the cases we need to consider?

We should test our code with data that meets the following criteria:

- Outlying values at both the low and high ends.
- Outlying values at the low end only.
- Outlying values at the high end only.
- No outlying values.
- Only outlying values (no valid ones).
- An empty data set.

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

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

The first parameter for the **print** function is defined in a strange way. **objects** has an asterisk before it.

That means you can provide zero or more values. We've usually only provided one value - or none, when we wanted to print a blank line.

I'll demonstrate **print** being used with several values, when we get back to our code.

Next, the **print** function defines the first of several **keyword** arguments.

They're also called **named arguments**, and you'll hear me use that term as well.

We've used keyword arguments, when we reversed the sort order, and again when we provided a key to the **sorted** function.

print defines 4 of these in total. The first one is **sep**.

keyword arguments are useful, because you can give them a default value.

If you don't provide a value for **sep**, it defaults to a space.

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

Ok, there was a lot to take in, in the last few videos.

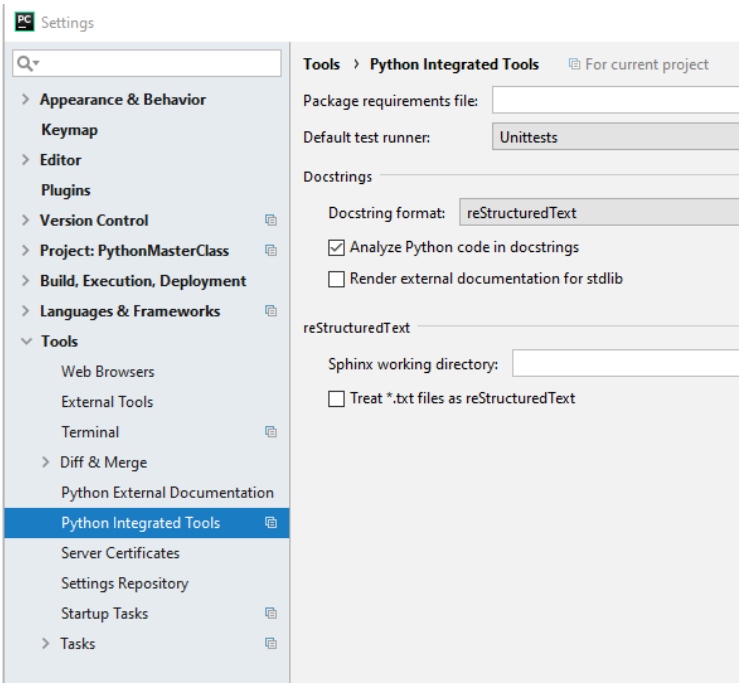
We've seen how to define **parameters**, and pass the corresponding **arguments** when calling our functions.

We've also learnt about **positional** arguments - where the arguments are used to provide a value for the parameter in the same position.

You can also use the parameters as **keyword** parameters, by specifying the parameter name when you pass the argument.

Docstrings (using reStructuredText)

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




```
help(get_integer)

'''
print(input.__doc__)
print("*" * 80)
print(get_integer.__doc__)
print("*" * 80)
'''
```

Python Course by Tim Buchalka, started 28.1.2022

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

Python Course by Tim Buchalka, started 28.1.2022

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

Python Course by Tim Buchalka, started 28.1.2022

dict objects

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

Shallow copy Lesson 205 copies only reference

Hash functions

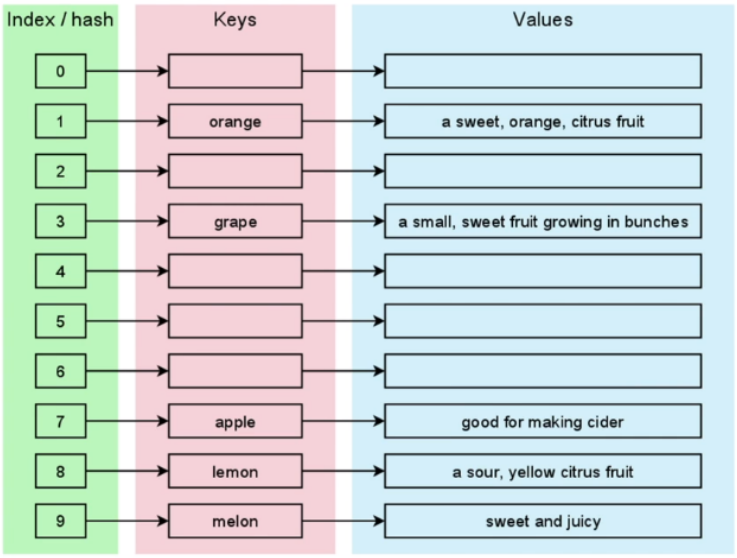
- https://en.wikipedia.org/wiki/Hash_function
- https://docs.python.org/3/glossary.html#term-hashable

Tools – Python Console (console woindow)

Tools

Python console

Hash tables

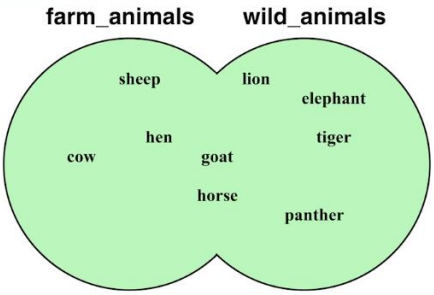


hashlib

- https://docs.python.org/3/library/hashlib.html

Introduction to sets

set union



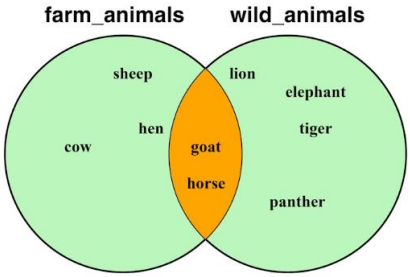
In Python, the union of these two sets is written:

```
farm_animals.union(wild_animals)
```

or

```
farm_animals | wild_animals
```

set intersection



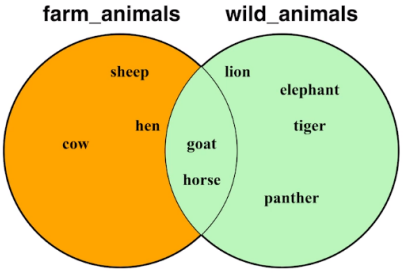
The Python code representing this intersection would be:

```
farm_animals.intersection(wild_animals)
```

or

```
farm_animals & wild_animals
```

set difference



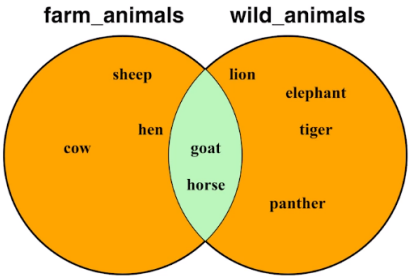
In Python, we'd write that as either

```
farm_animals - wild_animals
```

or

```
farm_animals.difference(wild_animals)
```

symmetric difference



In Python, we'd write that as:

`farm_animals.symmetric_difference(wild_animals)`

or

`farm_animals ^ wild_animals`

subsets and supersets

One set can also be a subset of another set.

We use the normal comparison operators; <, <=, > and >=, to check for subsets.

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:

```
choice = "-"  # initialise choice to something invalid
valid_choices = {"1", "2", "3", "4", "5"}
while choice != "0":
   if choice in valid_choices:
      print("You chose {}".format(choice))
```

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()`, `intersection_update()`, `difference_update()`,
and `symmetric_difference_update()` methods will accept any iterable as an argument.

update(**others*)
set |= other | ...
Update the set, adding elements from all others.

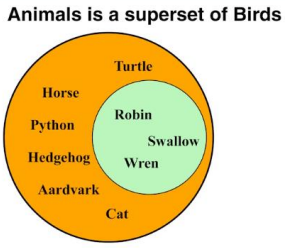
```
from prescription_data import adverse_interactions

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

superset



In set theory, that's written:

Animals ⊇ Birds

In Python, we use the greater than or equal to symbol:

Animals >= Birds

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 and 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

Python Course by Tim Buchalka, started 28.1.2022

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.

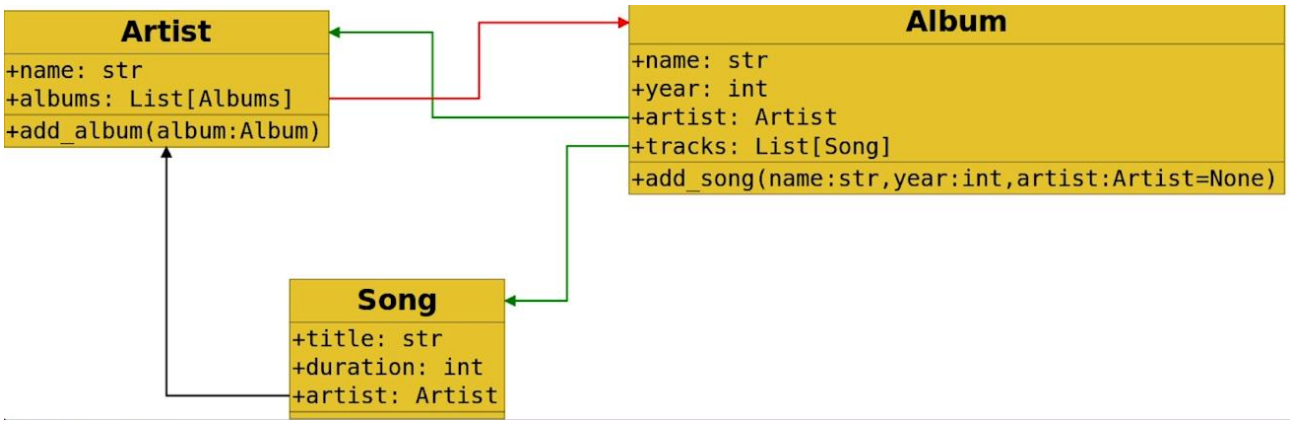
Every module creates its own namespace. Each class has its own namespace, and a function creates its own namespace each time it is run.

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

a) select files to be compared + CTRL + D or View + Compare Files:
   - opens a new window

b) select one file + CTRL + D opens files directory to select the other file to be combared

If using Python 2 in song.py

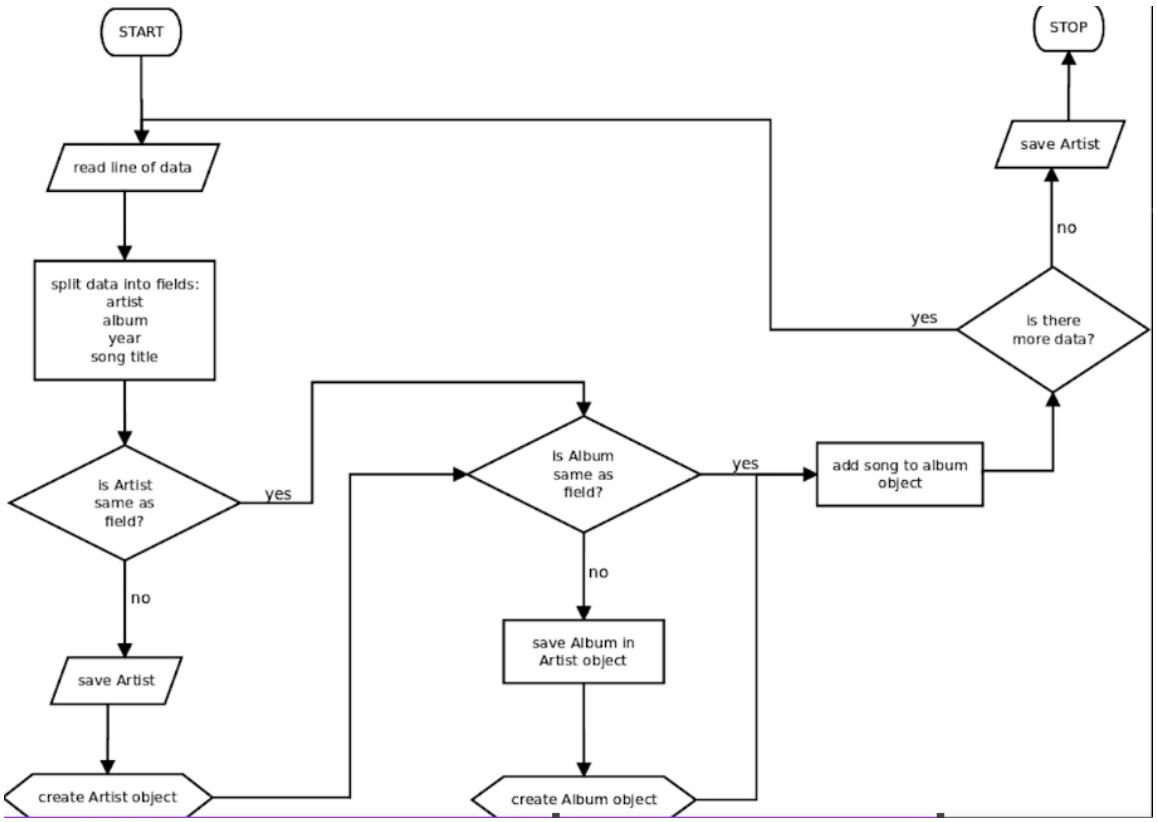
this line wouldn't work:

```
print("{0.name}\t{1.name}\t{1.year}\t{2.title}".format(new_artist, new_album, new_song),file=checkfile)
```

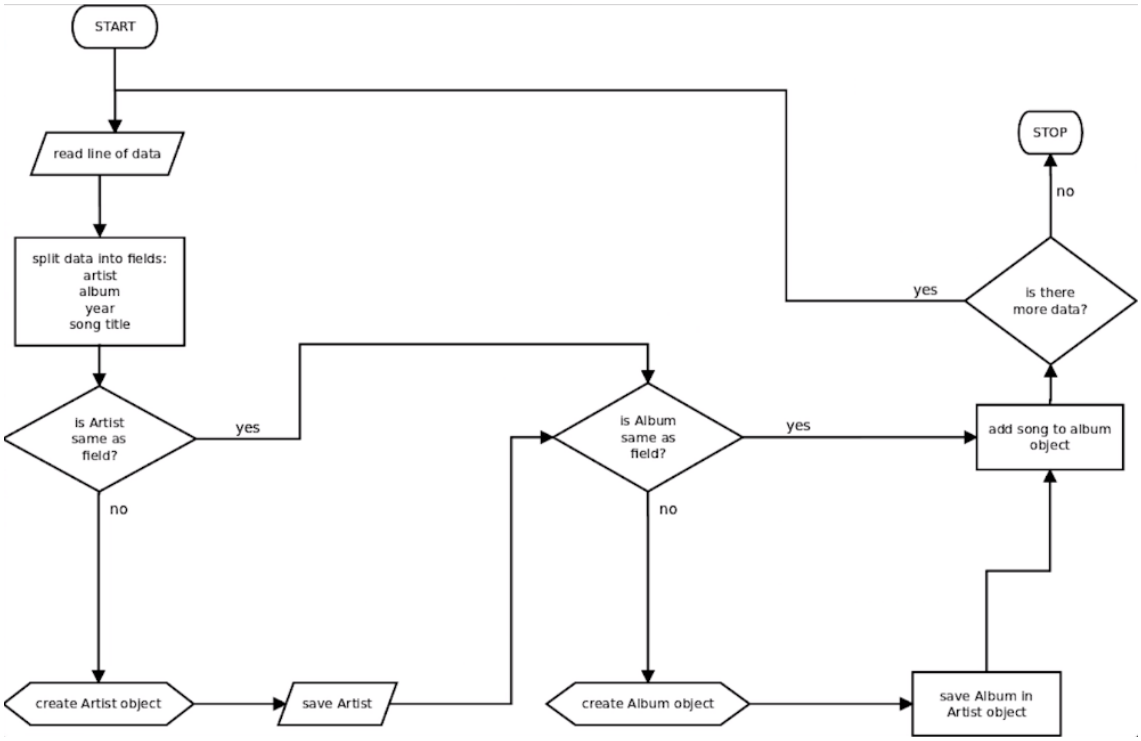
But using import as following it should work

Python Course by Tim Buchalka, started 28.1.2022

import from __future__ import print_function

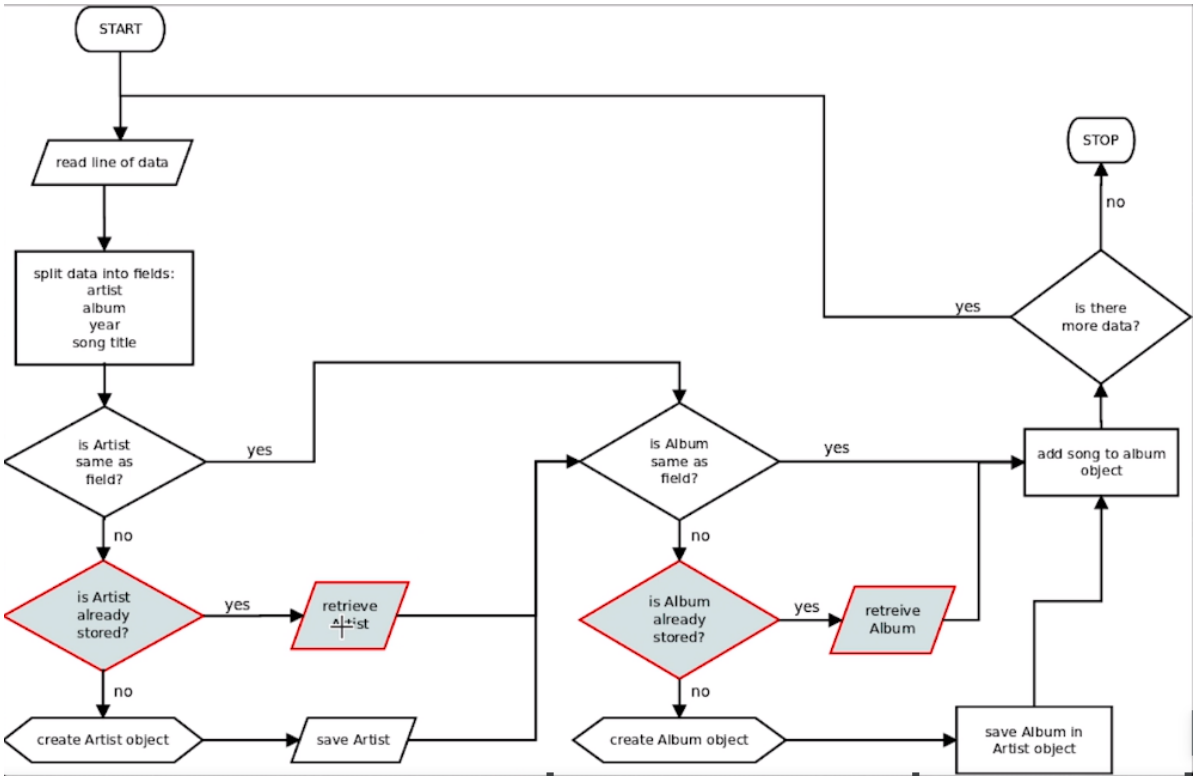


New one

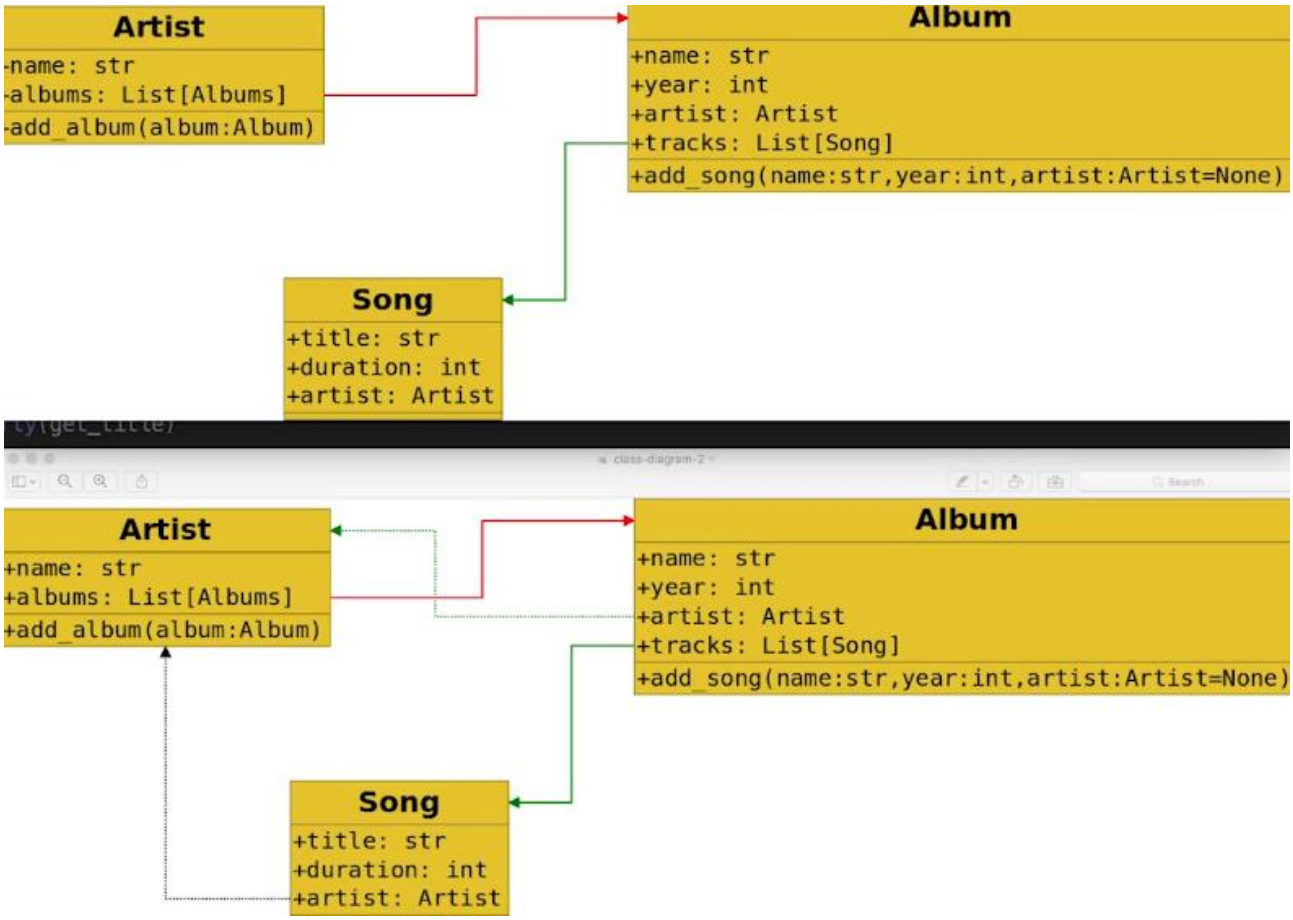


Python Course by Tim Buchalka, started 28.1.2022

Latest one



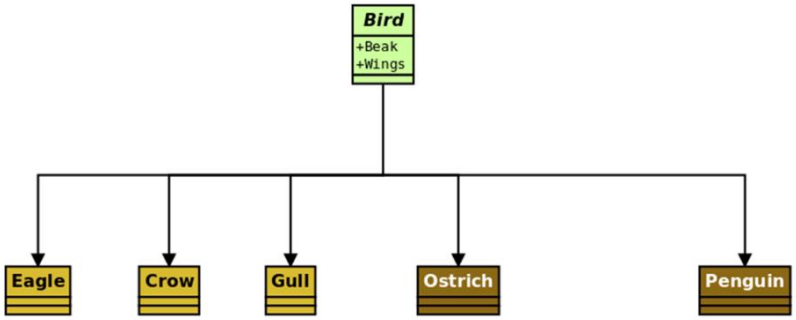
Challenge – remove circular references (aim to top diagram)

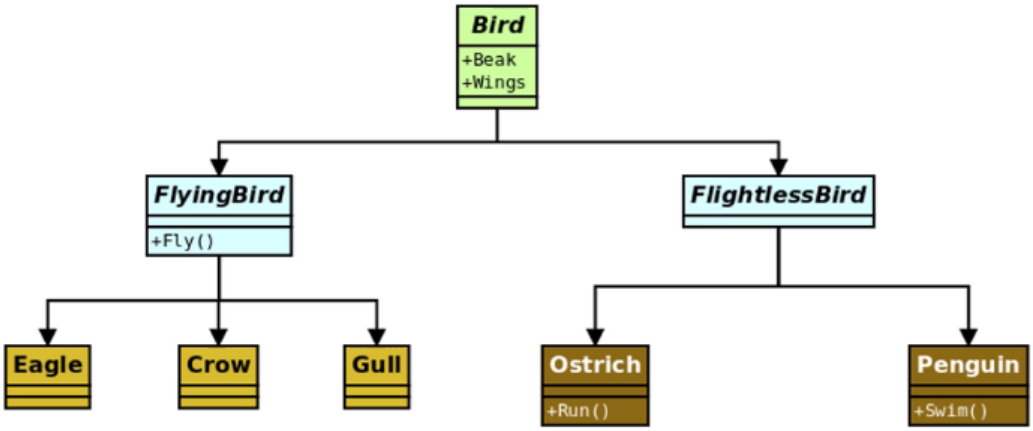


Getters and Setters

- https://docs.python.org/3.5/library/functions.html#property

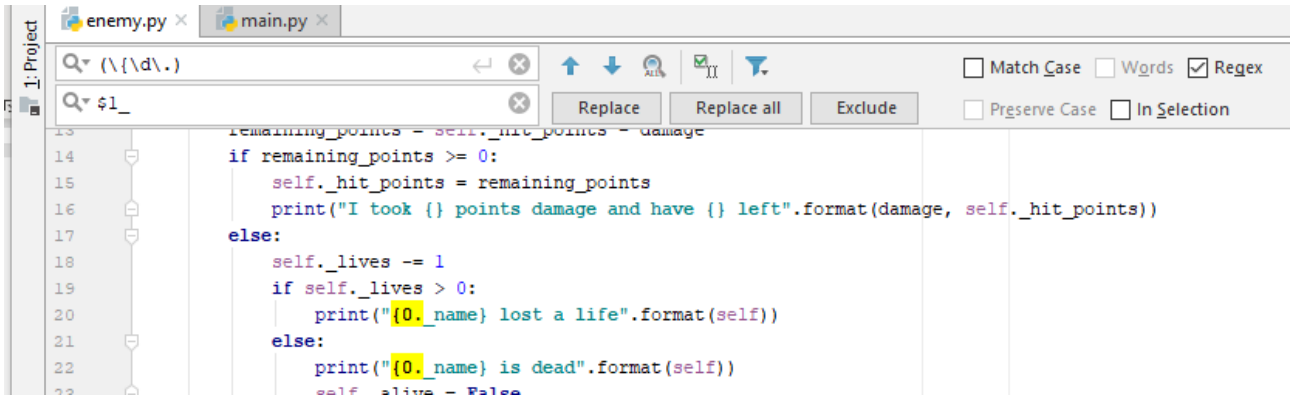
Inheritance (Python allows multiple inheritance but be aware that you should master it)





Python no method overloading

Python overriding



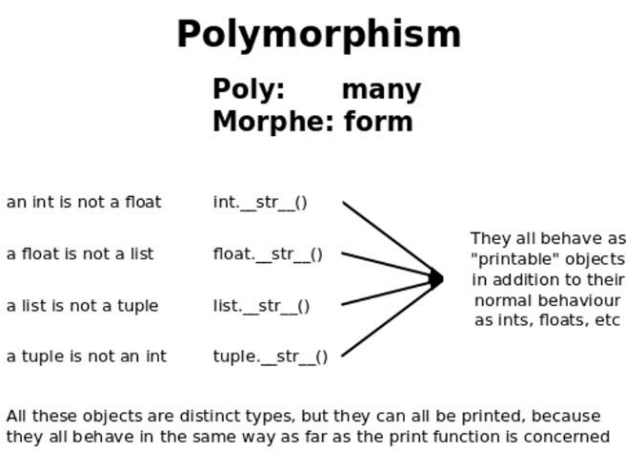
tab Regex + (\{\d\.) + $1_ Replace All

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)

Polymorphism



In this particular example, the polymorphic behaviour of the objects is implemented using inheritance. All Python objects inherit from a base class called **object**, which defines a **__str__** method.

So polymorphism allows the **print** function to accept arguments of any type, and it's able to print them out.

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;

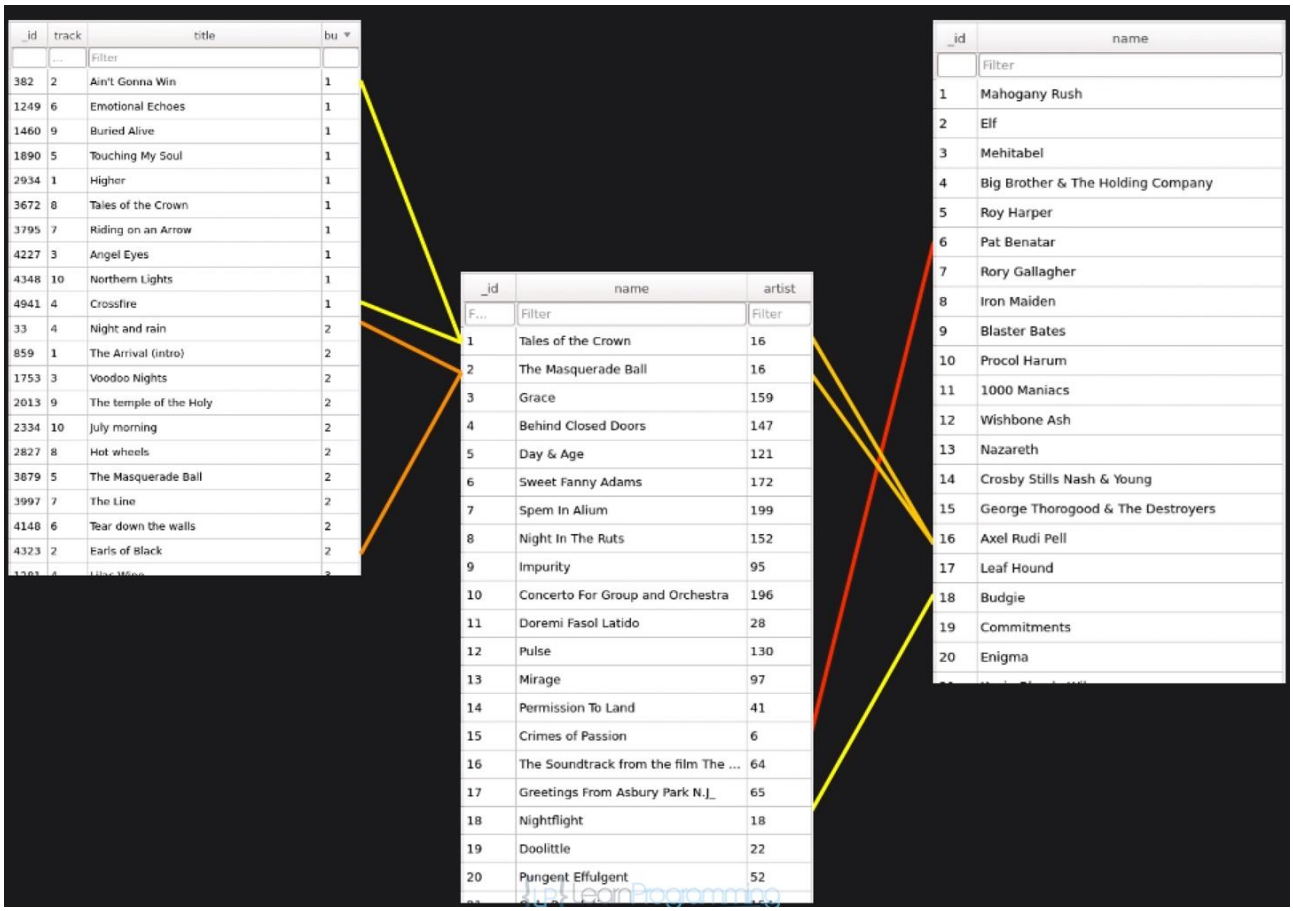
select * from albums where _id=439;

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;



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

1. Select the titles of all the songs on the album "Forbidden".

2. Repeat the previous query but this time display the songs in track order. You may want to include the track number in the output to verify that it worked ok.

3. Display all songs for the band "Deep Purple".

4. Rename the band "Mehitabel" to "One Kitten". Note that this is an exception to the advice to always fully qualify your column names. SET artists.name won't work, you just need to specify name.

5. Check that the record was correctly renamed.

6. Select the titles of all the songs by Aerosmith in alphabetical order. Include only the title in the output.

7. Replace the column that you used in the previous answer with count(title) to get just a count of the number of songs.

8. Search the internet to find out how to get a list of the songs from step 6 without any duplicates.

9. Search the internet again to find out how to get a count of the songs without duplicates. Hint: It uses the same keyword as step 8 but the syntax may not be obvious.

10. Repeat the previous query to find the number of artists (which, obviously, should be one) and the number of albums.

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;
3.

```
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”;
```

4. update artists set name = "One Kitten" where name = "Mehitabel";
5. 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`

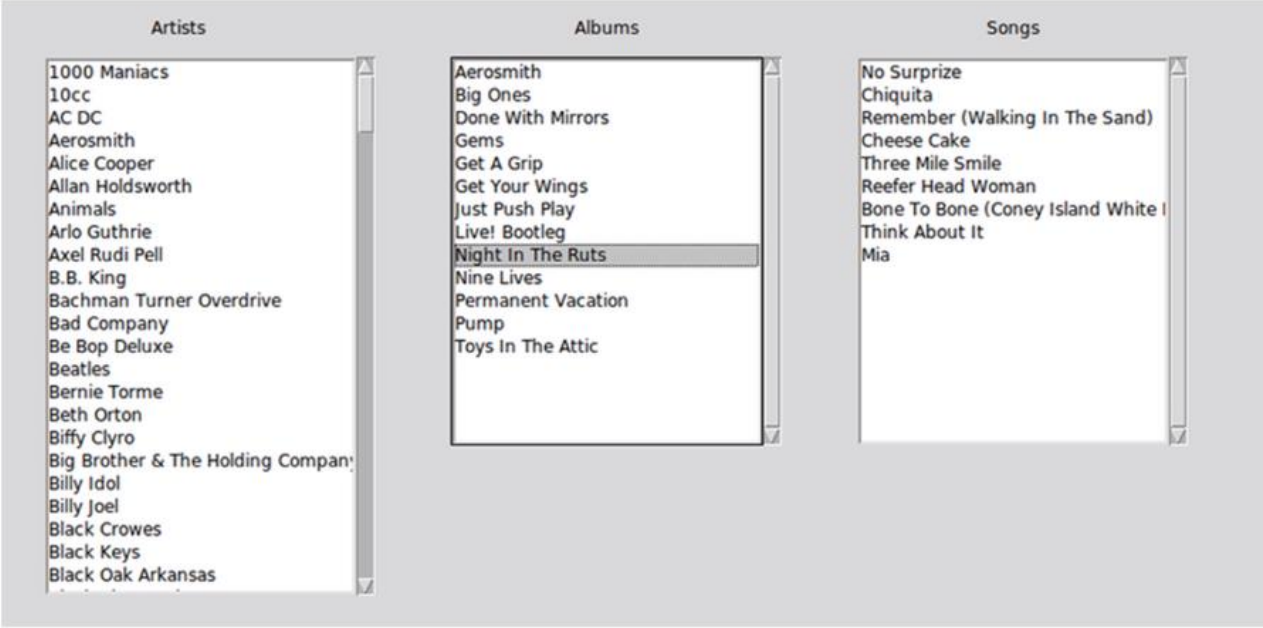
SQLite3 strftime Function

- 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