Python Cheatsheed

Contents

Foundamentals

Data types

Name	Data type	Description
string	'str'	sequence of characters
integer	1	integers numbers
float	0.1	decimals numbers
complex	1j	imaginary numbers
boolean	True/False	logical values
list	[a, b, ...]	mutable ordered sequence
tuple	(a, b, ...)	immutable ordered sequence
range	range(init, fin, step)	immutable ordered sequence
dictionary	{key=value, ...}	mutable unordered mapping
set	{x, ...}	mutable unordered set
frozenset	frozenset({x, ...})	immutable unordered set
byte		
bytearray		
memoryview		

Operators

Loops/Conditions

Functions

```
# function definition
def fun_name(param, *args, param=default, **kwargs):
    # function body
    ...
    return x    # value to return

# function call
a = fun_name(arg_1, ..., arg_n))
```

Input/Output

Filesystem

Error Handling

Common Modules

Argumens Parser

```
import argparse

# create object
arg = argparse.ArgumentParser()

# add argument
arg.add_argument('-strink_arg_name', 'full_arg_name',
↪  required=True, help='description')

args = vars(arg.parse_args())
```

Filesystem

```
from pathlib import Path

# creating path
address = Path('home', '...', 'file')
actual_path = Path.cwd()
home_path = Path.home()

# absolute path
address.is_absolute()

# get path parts
address.parent      # previous folder
address.name        # final position
address.stem        # home
address.drive
```

File Management

csv

```
import csv

# write
f = open('file.csv', 'w', newline = '')
f_csv = csv.writer(f, delimiter='\t',
↪  lineterminator='\n\n')
f_csv.writerow(['elem_0', ..., 'elem_n'])

# write as dictionary
f = open('file.csv', 'w', newline = '')
f_csv = csv.DictWriter(f, ['key_0', ..., 'key_n'])
f_csv.writeheader()
f_csv.writerow({'key_0': value_0, ..., 'key_n':
↪  value_n})

# read
f = open('file.csv', 'r')
f_csv = csv.reader(f)
for row in f_csv:
    print('Row #' + str(f_csv.line_num) + ' ' +
    ↪  str(row))

# read dictionary
f = open('file.csv')

f_csv = csv.DictReader(f, ['key_0', ..., 'key_n'])
for row in f_csv:
    print(row['key_0'], ..., row['key_n'])

# close
f.close()
```

json

```
import json

# write
json_write = {'key_0': value_0, ..., 'key_n':
↪  value_n}
stringOfJsonData = json.dumps(json_write)

# read
json_read = '{'key_0': value_0, ..., 'key_n':
↪  value_n}'
jsonDataAsPythonValue = json.loads(json_read)
print(jsonDataAsPythonValue)
```

Regular Expression

```
# object
regex_obj = re.compile(r'string_to_match')

# find the first result
match = regex_obj.search(Text)
regex_obj.search.group()
regex_obj.search.groups()

# find all results and replace
match = regex_obj.findall(Text)

# censoring data
Censored_Text = regex_obj.sub(r'CENSORED', Text)
```

string_to_match:

`string`	specific string
`(\d\D{1})(\w{2}\W)`	
`(...)`	grouping
`[]`	create a character class
`+`	match one or more characters
`*`	match zero or more characters
`?`	match zero or one character
`.`	match any character except for newline
`^string`	start with
`string&`	end with

Modules Management

Virtual environment

`python3 -m venv ws_name`	create virtual environment
`source ws_name/bin/activate`	activate virtual environment
`deactivate`	deactivate virtual environment

PIP

Basic commands

`pip3 list`	list of installed packages
`pip3 install`	install packages
`pip3 unistall`	unistall packages
`pip3 show`	info about package
`pip3 check`	verify dependencies
`pip3 search`	search PyPI for packages

Python dependancies

`pip3 freeze > deps.txt`	create list of dependencies
`pip3 install -r deps.txt`	install dependencies from list
`pip3 uninstall -r deps.txt`	unistall dependencies from list

Object Oriented Programming

Introduction

Principles

- Abstraction
- Encapsulation
- Inheritance
- Polimorphism

Rules

- S
- O
- L
- I
- D

Class

```
class ClassName:

    # Constructor
    def __init__(self, args):
        self.field = value

    # Method definition
    def method(self, args):
        ...


# Child class (Inheritance)
class ChildClassName(ClassName):

    # Static attribute (class attributes definition)
    class_attribute = value

    def __init__(self, args):
        super().__init__(self, args)
        self.field = value

        # modify class attribute
        ClassName.class_attribute = value

        # call class method
        ClassName.classmethod()

    # Static method (class method definition)
    @classmethod
    def class_method(cls):
        cls.class_attribute = value
```

Data Plotting

```
import matplotlib.pyplot as plt
```

Plot

```
fig, axes = plt.subplots(figsize=(12,6))

axes.plot(
    x, y,
    color='value',
    linewidth=value,
    marker='value',
    markersize=value,
    label='string'
)

plt.show()
```

Subplot

```
plot_obj = plt.subplots(nrows=nx, ncols=ny, figsize=(12,6))
fig, (axes_0, ..., axes_n) = plot_obj

# subplot 0
axes_0.plot(
    x0, y0, ...
)

...

# subplot n
axes_n.plot(
    xn, yn, ...
)

plt.show()
```

Scatter

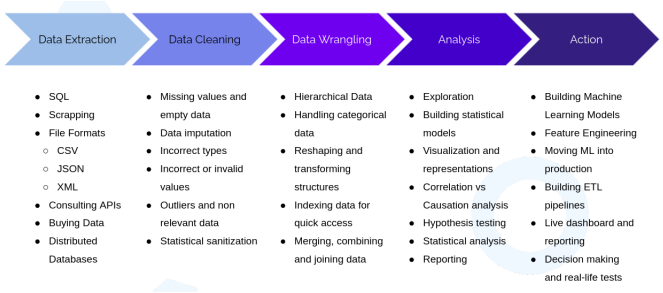
Bar

Histogram

Kernel density estimator

Data Analysis

Introduction



Useful libraries:

- pandas
- numpy
- matplotlib
- scipy

Read/Write files

```
df = pd.read_csv(csv_file_name)
df.to_csv('file_name.csv')
```

DataFrame

```
# Create dataframe
df = pd.DataFrame({
    'col_1': value.index,
    'col_2': value,
    ...
})

# Analysis
df.info()            # info about dataset
df.description()     # statistical info about dataset
df.columns()         # dataframe columns

# Indexing
x = df.column_title                 # single column tar
x = df[['column_title_1'], ...]     # multiple columns
```