Cheatography

My Python Modules Reference Cheat Sheet

by LoiVyen via cheatography.com/163788/cs/34340/

Pandas

```
import pandas as pd
```

```
mydataset = { 'cars': ["BMW", "Volvo", "Ford"],
   'passings': [3, 7, 2]
}
myvar = pd.DataFrame(mydataset) print(myvar)
```

```
             car s    pa ssings
0            BMW                  3
1           Volvo           7
2          Ford                2
```

```
df = pd.read_csv('data.csv')
print(df)
```

```
df = pd.read_csv(filepath_or_buffer="./testdata.csv",sep=",",dtype=str)
```

Pandas is a library used for working with data sets.
functions: analyzing, cleaning, exploring, and manipulating data.
https://pandas.pydata.org/docs/reference/
https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

PyPDF2

```
from PyPDF2 import PdfFil eReader
```

```
def text_e xtr act or( path):
    with open(path, 'rb') as f:
        pdf = PdfFil eRe ader(f) # get the first page
        page = pdf.ge tPa ge(1)
        print("page : \n ", page)
        print('\n Page type: {}'.fo rma t(s tr( typ e(p age))))

        text = page.e xtr act Text()
        print("page text: ", text)
```

```
if __name__ == '__mai n__':
    path = 'filen ame.pdf'  # enter file for input
    text_extractor(path)
```

extract metadata, extract text, split/merge pdfs, rotate pages, encryption:
https://www.blog.pythonlibrary.org/2018/06/07/an-intro-to-pypdf2/

icecream

```
from icecream import ic
```

output

```
def foo(i):
    return i + 333
```

ic| foo(123): 456

```
d = {'key': {1: 'one'}}
ic(d['key'][1])
```

ic| d['key'][1]: 'one'

```
class klass():
    attr = 'yep'
ic(klass.attr)
```

ic| klass.attr: 'yep'

to use instead of print and log for debugging
https://github.com/gruns/icecream

Extract Text From HTML Pages / Websites

```
conda install -c conda- forge readab ili ty-lxml
```

```
import requests
from readab ility import Document
response = reques ts.g et ('h ttp :// exa mpl e.co m')
doc = Docume nt( res pon se.t ext)
print(doc.title())

print(doc.summary())
```

Example Domain
<html><body> <body id="readabilityBody">

<h1>Example Domain</h1>
This domain is for use in illustrative examples in documents. You may use this
domain in literature without prior coordination or asking for permission.
More information...

</body>
</body></html>

Beautiful Soup (XML extractor)

Beautiful Soup is a Python library for pulling data out of HTML and XML files.

```
conda install -c conda- forge beauti ful soup4
```

```
from bs4 import Beauti fulSoup
html_doc = str(do c.s umm ary())
soup = Beauti ful Sou p(h tml _doc, 'html.p ar ser')
print(html_doc)  # prints with tags
print(soup.prettify())  # formats string from html tags

soup = Beauti ful Sou p(h tml _doc, featur es= " l xm l")
print(soup.get_text())
```

BeautifulSoup (html_string, parser_type), lxml is a python module

Example Domain
This domain is for use in illustrative examples in documents. You may use this
domain in literature without prior coordination or asking for permission.
More information...

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Keywords: assert try except

```
x = " goo dby e"
```

if condition returns False, AssertionError is raised:

```
assert x == " hel lo"
```

or to add a message to console:

```
assert x == " hel lo", " should be goodby e"
```

try except block with AssertionError

```
x = 'hello'
try:
    assert x == 'goodbye'
except:
    print("An exception occurr ed")
```

(so that the program continues running after the error)

https://www.w3schools.com/python/ref_keyword_assert.asp

The assert keyword is used when debugging code. The assert keyword lets you test if a condition in your code returns True, if not, the program will raise an AssertionError.

Try / Except / Catch differences

try contains the code that may raise exceptions or errors.
except is used to catch the exceptions and handle them.
catch code is executed only when the corresponding exception is raised.

One Line Loops

examples:

```
for i in range(10): print(i)
print( [i**2 for i in range(10) if i%2==0])
for i in range(10): print(i**2 if i<5 else 0)
print( " Text: ", [token.text for token in doc])
```

Formatting Strings with Variables

Example:

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I will pay {2} dollars for {0} pieces of
 item {1}."
print( myo rde r.f orm at( qua ntity, itemno, price
))
```

Read Write Print Files

Type Casting

Get Type from Variable

```
name = "freeCodeCamp"
print("The variable, name is of type:", type(n ame)
)
```

output:

The variable, name is of type: <class 'str'>

Specify/Convert Variable Type

```
x = int(1)   # x will be 1
y = float( " 2.8 ")   # y will be 2.8
z = str(3)   # z will be "3"
```

Regular Expression (or re / regex)

```
import re
```

```
re.sub (pa ttern, repl, string, count=0, flags=
0)
 result = re.sub ('abc',  '',    input)
 result = re.sub ('abc',  'def', input)
 result = re.sub (r' \s+', ' ',   input)
result = re.sub ('a bc( def )ghi', r'\1', input
)
```

D
patt
abc
R
patt
abc
#
Elim
dup
whit
pac
usir
wild
R
a st
with
of it

```
re.sea rch (pa ttern, string, flags=0)
re.search("c", "abcdef")
re.search("^c", "abcdef")
re.search('^X', 'A\nB\nX', re.MUL TILINE)
```

M
N
mat
(beg
of s
M
mul
(beg
eac

```
re.mat ch( " c", " abc def ")
```

N
mat
(.r
for
beg
of s

More Info:

https://docs.python.org/3/library/re.html

https://lzone.de/examples/Python%20re.sub

open file and read lines with formatting

```
with open("t ext fil e.t xt", encoding = 'utf-8') as f:

    lines = f.read lines() # as a list
    docs = list(n lp.p ip e(l ines))
```

(to spaCy docs)

```
for text in docs: #iterate each list item
    for firsts in text: # iterate each word in list (sentence)
        if firsts.is _se nt_ start:
            print(firsts.text)
```

```
doc = [nlp(t xt.s tr ip()) for txt in lines]
```

removes escape sequences

or

```
f = open("H and boo kli st.t xt ", " r")
print(f.read())
```

```
f = open("d emo fil e.t xt", " r")
print(f.readline()) # read first line
```

```
f = open("d emo.tx t", 'r', encodi ng= 'utf8')
print(f.read()) # ^ another way to strip
```

```
f = open("d emo.tx t", " r")
for x in f:
    print(x)
  # and/or
  links.a pp end (x.s tr ip()) # declare arr beforehand; links = [ ]
```

```
f.close() # best to close when done unless first option used
```

Write Files *('w' overwrites; 'a' appends file)*

```
f = open("m yfi le.t xt ", " w")
f.write(doclist)
f.close()
```

List to New Text File:

```
with open(r 'ne wfi le.t xt', 'w') as fp:
    for text in doclist:
        # write each item on a new line
        fp.write("%s\n" % text)
    print('Done')
```

https://www.w3schools.com/python/python_file_handling.asp
https://pynative.com/python-write-list-to-file/

https://www.w3schools.com/python/python_regex.asp

Practice Regex: https://regex101.com/

Dictionary

Create a dictionary

```
thisdict = {
         " bra nd": " For d",
    "model": " Mus tan g",
    "year": 1964,
    "year": 2020,
    "colors": ["re d", " whi te", " blu e"]
}
```

```
print( thi sdi ct[ " yea r"])
#print dictionary key for 'year'
 #(notice the overwr itten output; duplicates are not
)
```

Output:

~~1964~~
2020

```
print( thi sdict)
```

Output:

{'brand': 'Ford', 'electric': False, 'year': 1964, 'colors': ['red', 'white', 'blue']

Dictionary (cont)

Alternative way to create dictionary (Constructor)

```
thisdict = dict(brand = " For d", electric = False, year =

    2020, colors = ["re d", " whi te", " blu e"])
```

Alternate retrievals of key values

```
x = thisdi ct.g et ("mo del ")
print(x)
```

Output:
Mustang

or to get dictionary key names and/or values

```
x = thisdi ct.k eys()
```

Output:
dict_keys(['brand', 'electric', 'year', 'colors'])

```
x = thisdi ct.v al ues()
```

Output:
dict_values(['Ford', False, 2020, ['red', 'white', 'blue']])

```
x = thisdi ct.i tems() #get all keys & values
```

Make changes to dictionary

```
car["co lor s"] = " red "
print(x)
```

Output:
dict_values(['Ford', 'Mustang', 2020, 'red'])

or

```
thisdi ct.u pd ate ({"y ear ": 2022})
```

check if key/entry exists:

```
if " mod el" in thisdict:
    print("Yes, this key is in {thisdict} dictio nar y")
```

Remove Items

```
thisdi ct.p op ("mo del ") #removes specified key item
 thisdi ct.p op item() #removes last inserted* item
del thisdi ct[ " mod el"] #to remove key item or...
del thisdict #to delete the entire dictionary
```

Loop/Iterations on Dictionaries

Dictionary (cont)

https://www.w3schools.com/python/python_dictionaries.asp

clear() Removes all the elements
fromkeys() Returns dict w/ specified keys and value
get() Returns value of specified key
items() Returns a tupled list for each key value pair
keys() Returns a list of dict keys
pop() Removes element of specified key
popitem() Removes last inserted key-value pair
setdefault() Returns value of specified key
update() Updates dict w/ specified key-value pairs
values() Returns list of all values in dict

```
for x in thisdict:
    print(thisdict[x])
    #print each value in dict per line

for x in thisdi ct.v al ues():
    print(x)
    #print values of dictionary

for x in thisdi ct.k eys():
    print(x)
    #print keys of dictionary

for x, y in thisdi ct.i te ms():
    print(x)
    #loop through both keys and values
```

Nested Dictionary

```
child1 = {
    "name" : " Emi l",
    "year" : 2004
} child2 = {
    "name" : " Tob ias ",
    "year" : 2007
} child3 = {
    "name" : " Lin us",
    "year" : 2011
}

myfamily = {
    "child1" : child1,
    "child2" : child2,
    "child3" : child3
}
print(myfamily["child2"]["name"])
```

Output:
Tobias

Other uses with Dictionary

No. of components: `len(th isdict)`
Confirm dict data type: `type(t his dict)`
Make copy of dictionary: `mydict = thisdi ct.c opy()`