

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 PdfFileReader

def text_extractor(path):
    with open(path, 'rb') as f:
        pdf = PdfFileReader(f) # get the first page
        page = pdf.getPage(1)
        print("page : \n ", page)
        print('\n Page type: {}'.format(str(type(page))))

    text = page.extractText()
    print("page text: ", text)

if __name__ == '__main__':
    path = 'filename.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: 'yep'
ic(klass.attr)	

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 ility lxml

import requests
from readability import Document
response = requests.get ('http:// example.com')

doc = Document( response.text)
print(doc.title())

print(doc.summary())
```

Example Domain

```
<html><body><div><body id="readabilityBody">
<div>
<h1>Example Domain</h1>
<p>This domain is for use in illustrative examples in documents.
You may use this
domain in literature without prior coordination or asking for permis-
sion.</p>
<p><a href="https://www.iana.org/domains/example">More inform-
ation...</a></p>
</div>
</body>
</div></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 BeautifulSoup
html_doc = str(doc.summary())
soup = BeautifulSoup(html_doc, 'html.parser')

print(html_doc) # prints with tags
print(soup.prettify()) # formats string from html tags

soup = BeautifulSoup(html_doc, features= "lxml")
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.

C

By LoiVyen

cheatography.com/loivyen/

Not published yet.

Last updated 8th February, 2023.

Page 1 of 6.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

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( myorder.format( quantity, itemno, price
))
```

Read Write Print Files

Type Casting

Get Type from Variable

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

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 (pattern, repl, string, count=0, flags=0)
# Default: replace all occurrences of pattern with repl
result = re.sub ('abc', '', input) # Replace 'abc' with ''
result = re.sub ('abc', 'def', input) # Replace 'abc' with 'def'
result = re.sub (r' \s+', ' ', input) # Replace all whitespace with a single space
result = re.sub ('a bc( def )ghi', r'\1', input) # Replace 'a bc( def )ghi' with '\1'

re.search (pattern, string, flags=0)
# Return a match object if the pattern is found in the string, otherwise return None
re.match (pattern, string, flags=0)
# Return a match object if the pattern is found at the beginning of the string, otherwise return None
re.findall (pattern, string, flags=0)
# Return a list of all non-overlapping matches of the pattern in the string, as a list of strings
re.finditer (pattern, string, flags=0)
# Return an iterator over all non-overlapping matches of the pattern in the string, as a list of match objects
re.subn (pattern, repl, string, count=0, flags=0)
# Return the number of substitutions made by calling re.sub(pattern, repl, string, count=0, flags=0)
```

```
re.search ("c", "abcdef") # Match 'c'
re.search ("^c", "abcdef") # Match 'c' at the beginning
re.search ('^X', 'A\nB\nX', re.MULTILINE) # Match 'X' at the beginning of a line
re.match (" c", " abc def ") # Match ' c' at the beginning
```

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_file.txt", encoding = 'utf-8') as f:

```
lines = f.readlines() # as a list
docs = list(map(lambda line: line.strip(), lines))
```

(to space docs)

```
for text in docs: #iterate each list item
    for firsts in text: #iterate each word in list (sentence)
        if firsts.is_whitespace:
            print(firsts.text)
```

```
doc = [nlp(text.strip()) for text in lines]
```

removes escape sequences

or

```
f = open("H_and_booklist.txt", "r")
print(f.read())
```

```
f = open("demo_file.txt", "r")
print(f.readline()) # read first line
```

```
f = open("demo.txt", 'r', encoding= 'utf8')
print(f.read()) # ^ another way to strip
```

```
f = open("demo.txt", "r")
for x in f:
    print(x)
```

and/or

```
links.append(x.strip()) # declare arr beforehand; links = []
```

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

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

```
f = open("myfile.txt", "w")
f.write(doclist)
f.close()
```

List to New Text File:

```
with open('new_file.txt', '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 = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964,
    "year": 2020,
    "colors": ["red", "white", "blue"]
}
```

```
print(thisdict["year"])
```

```
#print dictionary key for 'year'
```

```
 #(notice the overwritten output; duplicates are not allowed)
```

Output:

1964

2020

```
print(thisdict)
```

Output:

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



By LoiVyen

cheatography.com/loivyen/

Not published yet.

Last updated 8th February, 2023.

Page 2 of 6.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

Dictionary (cont)

Alternative way to create dictionary (Constructor)

```
thisdict = dict(brand = "Ford", electric = False, year = 2020, colors = ["red", "white", "blue"])
```

Alternate retrievals of key values

```
x = thisdict.get("model")
print(x)
```

Output:
Mustang

or to get dictionary key names and/or values

```
x = thisdict.keys()
```

Output:

```
dict_keys(['brand', 'electric', 'year', 'colors'])
```

```
x = thisdict.values()
```

Output:

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

```
x = thisdict.items() #get all keys & values
```

Make changes to dictionary

```
car["colors"] = "red"
print(x)
```

Output:

```
dict_values(['Ford', 'Mustang', 2020, 'red'])
```

or

```
thisdict.update({"year": 2022})
```

check if key/entry exists:

```
if "model" in thisdict:
    print("Yes, this key is in {thisdict} dictionary")
```

Remove Items

```
thisdict.pop("model") #removes specified key item
thisdict.popitem() #removes last inserted* item
del thisdict["model"] #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 thisdict.values():
    print(x)
    #print values of dictionary

for x in thisdict.keys():
    print(x)
    #print keys of dictionary

for x, y in thisdict.items():
    print(x)
    #loop through both keys and values

```

Nested Dictionary

```

child1 = {
    "name" : "Emil",
    "year" : 2004
}
child2 = {
    "name" : "Tobias",
    "year" : 2007
}
child3 = {
    "name" : "Linus",
    "year" : 2011
}

myfamily = {
    "child1" : child1,
    "child2" : child2,
    "child3" : child3
}

print(myfamily["child2"]["name"])

```

Output:

Tobias

Other uses with Dictionary

```

No. of components: len(thisdict)
Confirm dict data type: type(thisdict)
Make copy of dictionary: mydict = thisdict.copy()

```



By **LoiVyen**
cheatography.com/loivyen/

Not published yet.
 Last updated 8th February, 2023.
 Page 3 of 6.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopad.com>