String

```
message = 'Hello world'
print(message[1:5])
```
Result : Hello

print(message.count('l')) → How many time does l exist in var message

print(message.find('world')) → If can't find, will return -1 as result.

message.replace('oldword', 'newworld')

formating : greeting, name = 'Hello', 'Ibrahim'

→ message = greeting + ', ' + name + '. welcome'

→ message = '{}, {}. welcome'.format(greeting, name)

→ message = f'{greeting}, {name}. welcome'

print(dir(message)) → show all method that can be used in message var

print(help(str)) → get " " in details

print(help(str.lower())) → show what lower() does.

Lists

```
Courses = ['php', 'python', 'secruity*', 'RHEL']
```

Courses.append('js') → add

Courses.insert(index 0, value 'CSS') → add to specific location in list

Courses.extend(new_list) → add new list to courses list value individuall

Courses.remove('php')

Courses.pop('html') → remove, by default will remove last value.

Course.reverse()

Course.sort()

Courses.sort(reverse = True) → sort + reverse in one line.

sorted_courses = sorted(courses) → it's useful because some time we won't to alter our main var

courses.index('php') → where php value exists

Courses_str = '-'.join(courses)

new list = course_str.split('-') → results as a list

tuples and list are exact the same, but list add be edited tuples can't

Sets

Set will remove duplicate value

Courses = {'php', 'js', 'python'}

new_Courses = {'php', 'js', 'ruby', 'C++'}

print(Courses.intersection(new_courses)) → Show similar courses

print(Courses.difference(new_courses)) → Show difference courses

print(Courses.union(new_courses))

How to create

```
# Empty Lists
empty_list = []
empty_list = list()

# Empty Tuples
empty_tuple = ()
empty_tuple = tuple()

# Empty Sets
empty_set = {} # This isn't right! It's a dict
empty_set = set()
```

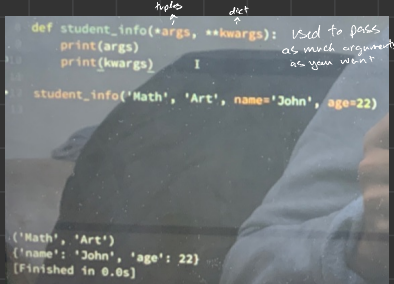
dicts

```
Student = {'name': 'Ibrahim', 'age': 20}
print(Student['name'])          -> get value of name key, not exist? return error key
print(Student.get('name'))      -> get  "  "  "  "  "  "  "  "  return None
print(Student.get('name', 'Not found'))   -> exercise not found, if key not exist
Student['name'] = 'Ahmed'       update one key.
Student.update({'name': 'Karim', 'age': 18})   update multiple keys.
del Student['num']              -> delete
age1 = Student.pop['age']       -> delete age, but save value in var age1 (pop will remove last also will return that no)
Student.keys()
Student.values()
Student.items()                 -> print key + values
for key, value in Student.items():
    print(key, value)

range(start, end)
```

Additional Note:

```
# False Values:
# False
# None
# Zero of any numeric type
# Any empty sequence. For example, '', (), [].
# Any empty mapping. For example, {}.

condition = 10                  -> everything else will be true

if condition:
    print('Evaluated to True')
else:
    print('Evaluated to False')
```

functions




```
def student_info(*args, **kwargs):
    print(args)
    print(kwargs)

courses = ['Math', 'Art']
info = {'name': 'John', 'age': 22}

student_info(*courses, **info)
```

for list for dict

Modules

How to set a pythonpath enviroment variable

Sys.path -> will show you all the python enviroment variable

in Mac: open terminal -> nano ~/.bash_profile -> and set following

export PYTHONPATH="your module path" + restart terminal

in Windows: System info window -> advanced system setting -> enviroment variable -> new -> name = PYTHONPATH

Value = your module path

in any python file: sys.path.append('your module python path')

LEGB (local, Enclosing, Global, Built-in)

Within a function, Top level in a module

- → x = 'global x' → global variable
- → def test():
  - Y = 'local y' → local variable, only in def is defind
  - global x → global "
- → def test(x): → x variable is local in the function
  - print(x)
- → import buildins
  - print(dir(buildins)) → will show all buildin variables.
  - Compiler will search firstly in our Global scope then build-in scope.
  - nonlocal x

Slicing lists

```
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
#          0, 1, 2, 3, 4, 5, 6, 7, 8, 9
#         -10,-9,-8,-7,-6,-5,-4,-3,-2,-1

# list[start:end:step]

print my_list[1:9]
```

mylist [start : End : step]

```
[1, 2, 3, 4, 5, 6, 7, 8]
[Finished in 0.1s]
```

```
sample_url = 'http://coreyms.com'
print sample_url

# Reverse the url
# print sample_url[::-1]

# # Get the top level domain
# print sample_url[-4:]

# # Print the url without the http://
# print sample_url[7:]

# # Print the url without the http:// or the top level domain
# print sample_url
```

```
http://coreyms.com
coreyms.com
[Finished in 0.1s]
```

Comprehensions :

1)

comprehensions.py

```
nums = [1,2,3,4,5,6,7,8,9,10]

# I want 'n' for each 'n' in nums
# my_list = []
# for n in nums:
#    my_list.append(n)
# print my_list

my_list = [n for n in nums]
print my_list
```

2)

```
# I want 'n*n' for each 'n' in nums
# my_list = []
# for n in nums:
#    my_list.append(n*n)
# print my_list

# my_list = [n*n for n in nums]
# print my_list

# Using a map + lambda
my_list = map(lambda n: n*n, nums)
print my_list
```

3)

```
nums = [1,2,3,4,5,6,7,8,9,10]

#I want 'n' for each 'n' in nums if 'n' is even
# my_list = []
# for n in nums:
#    if n%2 == 0:
#       my_list.append(n)
# print my_list

my_list = [n for n in nums if n%2 == 0]
print my_list

# Using a filter + lambda
# my_list = filter(lambda n: n%2 == 0, nums)
# print my_list
```

4)

```
nums = [1,2,3,4,5,6,7,8,9,10]

# I want a (letter, num) pair for each letter in 'abcd' and each number in '0123'
# my_list = []
# for letter in 'abcd':
#    for num in range(4):
#       my_list.append((letter,num))
# print my_list

my_list = [(letter, num) for letter in 'abcd' for num in range(4)]
print my_list
```

5

```
nums = [1,2,3,4,5,6,7,8,9,10]

# Dictionary Comprehensions
names = ['Bruce', 'Clark', 'Peter', 'Logan', 'Wade']
heros = ['Batman', 'Superman', 'Spiderman', 'Wolverine', 'Deadpool']

# I want a dict{'name': 'hero'} for each name,hero in zip(names, heros)
# my_dict = {}
# for name, hero in zip(names, heros):
#     my_dict[name] = hero
# print my_dict

my_dict = {name: hero for name, hero in zip (names, heros)}  if name != 'peter'  -> if you don't want to add peter.
print my_dict


# If name not equal to Peter
```

```
{'Bruce': 'Batman', 'Wade': 'Deadpool', 'Logan': 'Wolverine', 'Peter': 'Spiderman', 'Clark': 'Superman'}
[Finished in 0.1s]
```

6

```
# Set Comprehensions
nums = [1,1,2,1,3,4,3,4,5,5,6,7,8,7,9,9]
# my_set = set()
# for n in nums:
#     my_set.add(n)
# print my_set

my_set = {n for n in nums}
print my_set
```

Note:
Set() is like list but it has only unique value

print 'I am CS in CS!', (** per)

, , , (* per)

String Formatting advanced

```
tag = 'h1'
text = 'This is a headline'

sentence = '<{0}>{1}</{0}>'.format(tag, text)
print(sentence)
```

```
class Person():

    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person('Jack', '33')

sentence = 'My name is {0.name} and I am {0.age} years old.'.format(p1)
print(sentence)
```

```
person = {'name': 'Jenn', 'age': 23}

sentence = 'My name is {0[name]} and I am {0[age]} years old.'.format(person)
print(sentence)
```

```
person = {'name': 'Jenn', 'age': 23}

sentence = 'My name is {name} and I am {age} years old.'.format(**person)
print(sentence)
```

```
person = {'name': 'Jenn', 'age': 23}

l = ['Jenn', 23]

sentence = 'My name is {0[0]} and I am {0[1]} years old.'.format(l)
print(sentence)
```

```
for i in range(1, 11):
    sentence = 'The value is {:02}'.format(i)
    print(sentence)
```

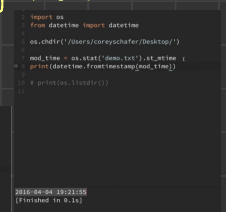
2f -> 3.1432 -> 3.14

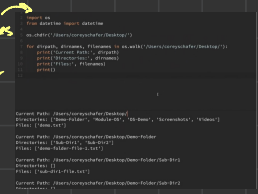
to format your strings print 2 digits

```
sentence = 'My name is {name} and I am {age} years old.'.format(name='Jenn', age='30')
print(sentence)
```

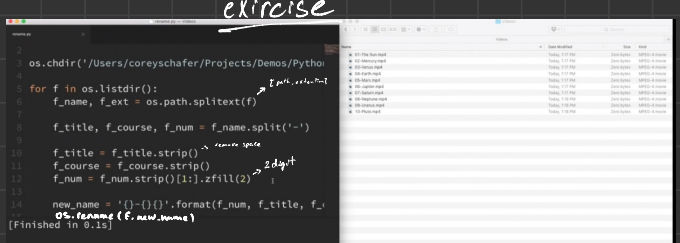
OS Module

```
import os
os.getcwd()                 -> get current dir
os.chdir("c:\")             -> Change dir
os.listdir()
os.mkdir("dir1")            -> create only dir no subdir create supported
os.makedirs("dir1/dir2")    -> create dir & subdir if needed
os.rmdir('dir1')
os.removedir("dir1/dir2")   -> Support to remove subdir, it's danger, be careful with it.
os.rename("oldname.txt", "newname.txt")
os.stat("demo.txt")         -> print file infos
os.walk("path")             -> will show all dir + subdir + files
os.environ.get('HOME')      -> get location of home dir
os.path.join(os.environ.get('HOME'), 'test.txt')
    ↳ for solving slash problem.
os.path.basename('/tmp/text.txt')   -> get file name
os.path.dirname('/tmp/text.txt')    -> get dir name
os.path.exists('file.txt')
os.path.isfile('path')
os.path.isdir('path')
os.path.splitext('/tmp/text.txt')   -> split file root & extension
print(dir(os.path))         -> show all function that can be used.
```





exercise



Datetime

import datetime

datetime.date(2016, 7, 24)

datetime.date.today() [.year .month .day]

datetime.timedelta(days=7)
hours=12
minutes=60

```python
import datetime

tday = datetime.date.today()

tdelta = datetime.timedelta(days=7)

# print(tday - tdelta)

# date2 = date1 + timedelta
# timedelta = date1 + date2

bday = datetime.date(2016, 9, 24)

till_bday = bday - tday
print(till_bday.total_seconds())
```

```
5184000.0
[Finished in 0.0s]
```

```python
import datetime

tday = datetime.date.today()

tdelta = datetime.timedelta(days=7)

print(tday - tdelta)
```

```
2016-07-19
[Finished in 0.0s]
```

datetime.date → weeks, months, years only

datetime.time → hours, minute, second, micro seconds only

datetime.datetime → will give access to all above

datetime.time(hours, minutes, seconds, microsecond)

d = datetime.datetime(year, month, day, hour, minute, second, microsecond)

print(d.[time(), date(), year ...])

Same
- datetime.datetime.today() → return normal
- datetime.datetime.now() → " " + give us the option to set time
- datetime.datetime.utcnow →

import pytz

datetime.datetime.today(tz=pytz.UTC)

Files (r, w, e)

`f = open('text.txt', 'r')` r = reading, w = writing, a = append

r+ = read + write, rb, wb = for bytes

- `f.name` → print name file
- `f.mode` → " mode
- `f.read()` → read each line all content
- `f.readlines()` → read lines as a list
- `f.readline()` → read first line (end='')
- `f.tell()` → How much char's is read
- `f.seek()` → Start reading frome beginning of the file, overwrite also
- `f.seek(0)` → will overwrite first character, will not delete the rest.

```
with open('bronx.jpg', 'rb') as rf:
    with open('bronx_copy.jpg', 'wb') as wf:
        for line in rf:
            wf.write(line)
```

```
# File Objects

with open('bronx.jpg', 'rb') as rf:
    with open('bronx_copy.jpg', 'wb') as wf:
        chunk_size = 4096
        rf_chunk = rf.read(chunk_size)
        while len(rf_chunk) > 0:
            wf.write(rf_chunk)
            rf_chunk = rf.read(chunk_size)
```

Random

import random

random.random() -> to get value between 0 - 1 (0,234678...)

random.uniform(1,10) -> " " " " 1 - 10 (8.345...)

random.randint(1,10) -> " " " " 1 - 10 (1, 2, 3, 5..)

random.choice(mylist) -> grabs random value from mylist

random.choices(mylist, k=10, weights=[25, 50, 25])
- Show 10 values
- Value 1 25% chance
- " " 50% chance
- " " 25% chance

random.shuffle(mylist) -> print all values, but random places (index)

random.sample(deck, k=5) -> print 5 random unique values.

```python
fake_cities = ['Metropolis', 'Eerie', "King's Landing", 'Sunnydale', 'Bedrock', 'South Park', 'Atlantis',

states = ['AL', 'AK', 'AZ', 'AR', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'ID', 'IL', 'IN', 'IA',

for num in range(100):
    first = random.choice(first_names)
    last = random.choice(last_names)

    phone = f'{random.randint(100, 999)}-555-{random.randint(1000,9999)}'

    street_num = random.randint(100, 999)
    street = random.choice(street_names)
    city = random.choice(fake_cities)
    state = random.choice(states)
    zip_code = random.randint(10000, 99999)
    address = f'{street_num} {street} St., {city} {state} {zip_code}'

    email = first.lower() + last.lower() + '@bogusemail.com'

    print(f'{first} {last}\n{phone}\n{address}\n{email}\n')
```

CSV module

import csv

(1)

```
import csv

with open('names.csv', 'r') as csv_file:
    csv_reader = csv.reader(csv_file)

    next(csv_reader)

    for line in csv_reader:
        print(line[2])
```

← bypass first line

```
john-doe@bogusemail.com
maryjacobs@bogusemail.com
davesmith@bogusemail.com
janestuart@bogusemail.com
tomwright@bogusemail.com
steverobinson@bogusemail.com
nicolejacobs@bogusemail.com
janewright@bogusemail.com
janedoe@bogusemail.com
```

(2)

```
import csv

with open('names.csv', 'r') as csv_file:
    csv_reader = csv.reader(csv_file)

    with open('new_names.csv', 'w') as new_file:
        csv_writer = csv.writer(new_file, delimiter='-')

        for line in csv_reader:
            csv_writer.writerow(line)
```

values ال محتاجين

(3)

```
import csv

with open('names.csv', 'r') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    for line in csv_reader:
        print(line['email'])

#     with open('new_names.csv', 'w') as new_file:
#         csv_writer = csv.writer(new_file, delimiter='\t')

#         for line in csv_reader:
#             csv_writer.writerow(line)
```

```
OrderedDict([('first_name', 'John'), ('last_name', 'Doe'), ('email', 'john-doe@bogusemail.com')])
OrderedDict([('first_name', 'Mary'), ('last_name', 'Smith-Robinson'), ('email', 'maryjacobs@bogusemail.com')])
OrderedDict([('first_name', 'Dave'), ('last_name', 'Smith'), ('email', 'davesmith@bogusemail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Stuart'), ('email', 'janestuart@bogusemail.com')])
OrderedDict([('first_name', 'Tom'), ('last_name', 'Wright'), ('email', 'tomwright@bogusemail.com')])
OrderedDict([('first_name', 'Steve'), ('last_name', 'Robinson'), ('email', 'steverobinson@bogusemail.com')])
OrderedDict([('first_name', 'Nicole'), ('last_name', 'Jacobs'), ('email', 'nicolejacobs@bogusemail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Wright'), ('email', 'janewright@bogusemail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Doe'), ('email', 'janedoe@bogusemail.com')])
```

(4)

```
import csv

with open('names.csv', 'r') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    with open('new_names.csv', 'w') as new_file:
        fieldnames = ['first_name', 'last_name', 'email']

        csv_writer = csv.DictWriter(new_file, fieldnames=fieldnames, delimiter='\t')

        csv_writer.writeheader()

        for line in csv_reader:
            csv_writer.writerow(line)
```

(5)

```
import csv

with open('names.csv', 'r') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    with open('new_names.csv', 'w') as new_file:
        fieldnames = ['first_name', 'last_name']

        csv_writer = csv.DictWriter(new_file, fieldnames=fieldnames, delimiter='\t')

        csv_writer.writeheader()

        for line in csv_reader:
            del line['email']
            csv_writer.writerow(line)
```

(6)

```
import csv

html_output = ''
names = []

with open('patrons.csv', 'r') as data_file:
    csv_data = csv.DictReader(data_file)

    # We don't want first line of bad data
    next(csv_data)

    for line in csv_data:
        if line['FirstName'] == 'No Reward':
            break
        names.append(f"{line['FirstName']} {line['LastName']}")

html_output += f'<p>There are currently {len(names)} public contributors. Thank You!</p>'

html_output += '\n<ul>'

for name in names:
    html_output += f'\n\t<li>{name}</li>'

html_output += '\n</ul>'

print(html_output)
```

```
<p>There are currently 30 public contributors. Thank You!</p>
<ul>
    <li>John Doe</li>
    <li>Dave Smith</li>
    <li>Mary Jacobs</li>
    <li>Jane Stuart</li>
    <li>Tom Wright</li>
    <li>Steve Robinson</li>
    <li>Nicole Jacobs</li>
    <li>Jane Wright</li>
    <li>Jane Doe</li>
    <li>Kurt Wright</li>
```

Regular expressions

```
import re
print('\tTab')      -> \t will make space
print(r'\tTab')     -> because of r, there will not be any space \t
patter = re.compile (r'abc')      -> compile will put our search in var
```

```
import re

text_to_search = '''
abcdefghijklmnopqurtuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890

Ha HaHa

MetaCharacters (Need to be escaped):
. ^ $ * + ? { } [ ] \ | ( )

coreyms.com

321-555-4321
123.555.1234

Mr. Schafer
Mr Smith
Ms Davis
Mrs. Robinson
```

need to be escaped

```
sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'abc')

matches = pattern.finditer(text_to_search)

for match in matches:
    print(match)

print(text_to_search[1:4])
```

```
<_sre.SRE_Match object; span=(1, 4), match='abc'>
abc
```

```
.       - Any Character Except New Line
\d      - Digit (0-9)
\D      - Not a Digit (0-9)
\w      - Word Character (a-z, A-Z, 0-9, _)
\W      - Not a Word Character
\s      - Whitespace (space, tab, newline)
\S      - Not Whitespace (space, tab,
newline)
```

```
\b      - Word Boundary
\B      - Not a Word Boundary
^       - Beginning of a String
$       - End of a String
```

-> match words with boundary
-> = = without =

```
re.compile ('\d\d\d.\d\d\d.\d\d\d\d')
```

match only dot.

any character

[-.] match dash/dot
[1-5] match this range
[^a-z A-Z] match which is not char

```
Mrs. Robinson
Mr. T
'''

sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'[89]00[-.]\d\d\d[-.]\d\d\d\d')

matches = pattern.finditer(text_to_search)

for match in matches:
    print(match)

# with open('data.txt', 'r', encoding='utf-8') as f:
#     contents = f.read()
```

```
<_sre.SRE_Match object; span=(194, 206), match='800-555-1234'>
<_sre.SRE_Match object; span=(207, 219), match='900-555-1234'>
```

[^b]at match anything that does not begin with b follow by at.

```
[]      - Matches Characters in brackets
[^ ]    - Matches Characters NOT in brackets
|       - Either Or
( )     - Group
```

```
Quantifiers:
*       - 0 or More
+       - 1 or More
?       - 0 or One
{3}     - Exact Number
{3,4}   - Range of Numbers (Minimum, Maximum)
```

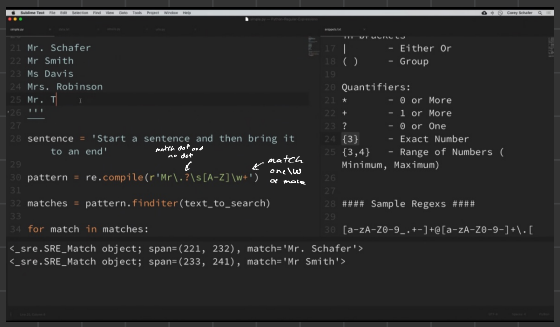
-> match 3 digits \d{3} = \d\d\d

```
Mr. Schafer
Mr Smith
Ms Davis
Mrs. Robinson
Mr. T
'''

sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'Mr\.?')

matches = pattern.finditer(text_to_search)

for match in matches:
```

```
<_sre.SRE_Match object; span=(221, 224), match='Mr.'>
<_sre.SRE_Match object; span=(233, 235), match='Mr'>
<_sre.SRE_Match object; span=(251, 253), match='Mr'>
<_sre.SRE_Match object; span=(265, 268), match='Mr.'>
```

```
Mr. Schafer
Mr Smith
Ms Davis
Mrs. Robinson
Mr. T
'''

sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'Mr\.?\s[A-Z]')

matches = pattern.finditer(text_to_search)

for match in matches:
```

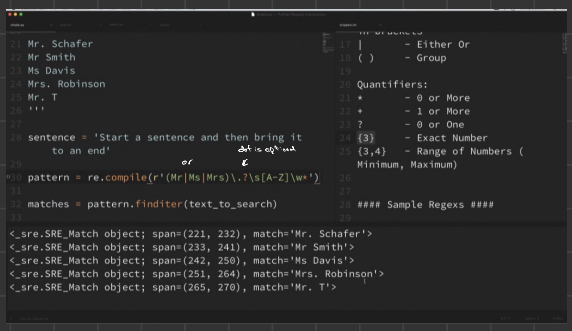
```
<_sre.SRE_Match object; span=(221, 226), match='Mr. S'>
<_sre.SRE_Match object; span=(233, 237), match='Mr S'>
<_sre.SRE_Match object; span=(265, 270), match='Mr. T'>
```



```
Mr. Schafer
Mr Smith
Ms Davis
Mrs. Robinson
Mr. T
'''

sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'Mr\.?\s[A-Z]\w*')

matches = pattern.finditer(text_to_search)

for match in matches:
```

```
<_sre.SRE_Match object; span=(221, 232), match='Mr. Schafer'>
<_sre.SRE_Match object; span=(233, 241), match='Mr Smith'>
```

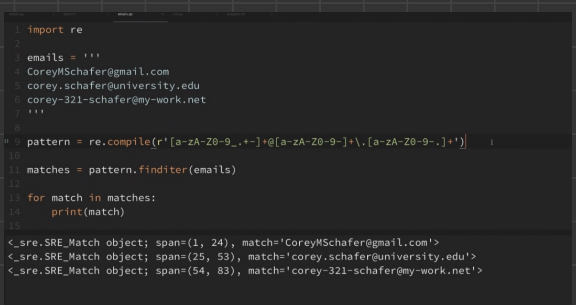
```
|       - Either Or
( )     - Group

Quantifiers:
*       - 0 or More
+       - 1 or More
?       - 0 or One
{3}     - Exact Number
{3,4}   - Range of Numbers (Minimum, Maximum)

#### Sample Regexs ####

[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[
```



```
Mr. Schafer
Mr Smith
Ms Davis
Mrs. Robinson
Mr. T
'''

sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'(Mr|Ms|Mrs)\.?\s[A-Z]\w*')

matches = pattern.finditer(text_to_search)
```

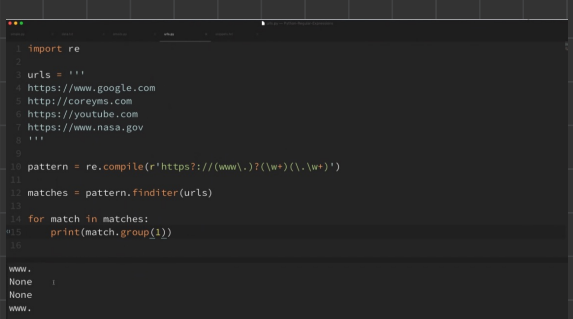
```
<_sre.SRE_Match object; span=(221, 232), match='Mr. Schafer'>
<_sre.SRE_Match object; span=(233, 241), match='Mr Smith'>
<_sre.SRE_Match object; span=(242, 250), match='Ms Davis'>
<_sre.SRE_Match object; span=(251, 264), match='Mrs. Robinson'>
<_sre.SRE_Match object; span=(265, 270), match='Mr. T'>
```

```
|       - Either Or
( )     - Group

Quantifiers:
*       - 0 or More
+       - 1 or More
?       - 0 or One
{3}     - Exact Number
{3,4}   - Range of Numbers (Minimum, Maximum)

#### Sample Regexs ####
```



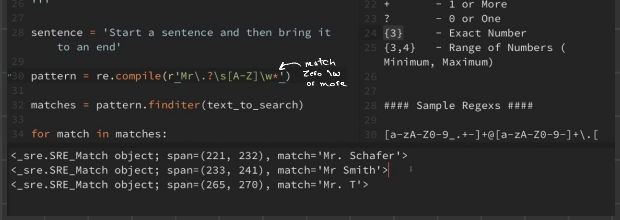
```
import re

emails = '''
CoreyMSchafer@gmail.com
corey.schafer@university.edu
corey-321-schafer@my-work.net
'''

pattern = re.compile(r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+')

matches = pattern.finditer(emails)

for match in matches:
    print(match)
```

```
<_sre.SRE_Match object; span=(1, 24), match='CoreyMSchafer@gmail.com'>
<_sre.SRE_Match object; span=(25, 53), match='corey.schafer@university.edu'>
<_sre.SRE_Match object; span=(54, 83), match='corey-321-schafer@my-work.net'>
```



```
import re

urls = '''
https://www.google.com
http://coreyms.com
https://youtube.com
https://www.nasa.gov
'''

pattern = re.compile(r'https?://(www\.)?(\w+)(\.\w+)')

matches = pattern.finditer(urls)

for match in matches:
    print(match.group(1))
```

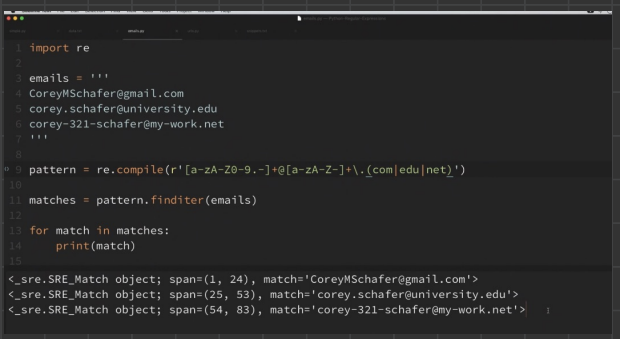
```
www.
None
None
www.
```



```
Mr. Schafer
Mr Smith
Ms Davis
Mrs. Robinson
Mr. T
'''

sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'Mr\.?\s[A-Z]\w*')

matches = pattern.finditer(text_to_search)

for match in matches:
```

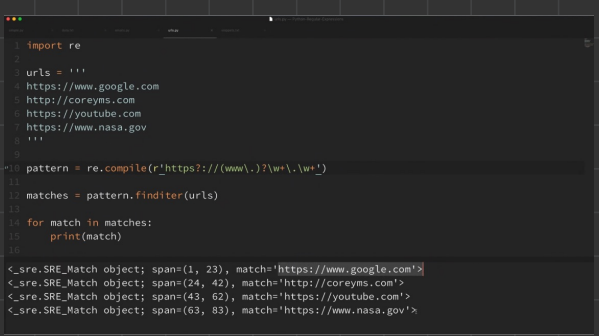
```
<_sre.SRE_Match object; span=(221, 232), match='Mr. Schafer'>
<_sre.SRE_Match object; span=(233, 241), match='Mr Smith'>
<_sre.SRE_Match object; span=(265, 270), match='Mr. T'>
```

```
|       - Either Or
( )     - Group

Quantifiers:
*       - 0 or More
+       - 1 or More
?       - 0 or One
{3}     - Exact Number
{3,4}   - Range of Numbers (Minimum, Maximum)

#### Sample Regexs ####

[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[
```



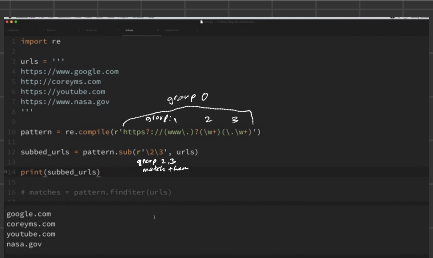
```
import re

emails = '''
CoreyMSchafer@gmail.com
corey.schafer@university.edu
corey-321-schafer@my-work.net
'''

pattern = re.compile(r'[a-zA-Z0-9.-]+@[a-zA-Z-]+\.(com|edu|net)')

matches = pattern.finditer(emails)

for match in matches:
    print(match)
```

```
<_sre.SRE_Match object; span=(1, 24), match='CoreyMSchafer@gmail.com'>
<_sre.SRE_Match object; span=(25, 53), match='corey.schafer@university.edu'>
<_sre.SRE_Match object; span=(54, 83), match='corey-321-schafer@my-work.net'>
```



```
import re

urls = '''
https://www.google.com
http://coreyms.com
https://youtube.com
https://www.nasa.gov
'''

pattern = re.compile(r'https?://(www\.)?\w+\.\w+')

matches = pattern.finditer(urls)

for match in matches:
    print(match)
```

```
<_sre.SRE_Match object; span=(1, 23), match='https://www.google.com'>
<_sre.SRE_Match object; span=(24, 42), match='http://coreyms.com'>
<_sre.SRE_Match object; span=(43, 62), match='https://youtube.com'>
<_sre.SRE_Match object; span=(63, 83), match='https://www.nasa.gov'>
```



```
import re

urls = '''
https://www.google.com
http://coreyms.com
https://youtube.com
https://www.nasa.gov
'''

pattern = re.compile(r'https?://(www\.)?(\w+)(\.\w+)')

subbed_urls = pattern.sub(r'\2\3', urls)

print(subbed_urls)

# matches = pattern.finditer(urls)
```

```
google.com
coreyms.com
youtube.com
nasa.gov
```

```
27
28 sentence = 'Start a sentence and then bring it to an end'
29
30 pattern = re.compile(r'Start')
31
32 matches = pattern.match(sentence)
33
34 print(matches)
35
```

flags = re.IGNORECASE

-> Will return the first match, None if no match

```
<_sre.SRE_Match object; span=(0, 5), match='Start'>
```

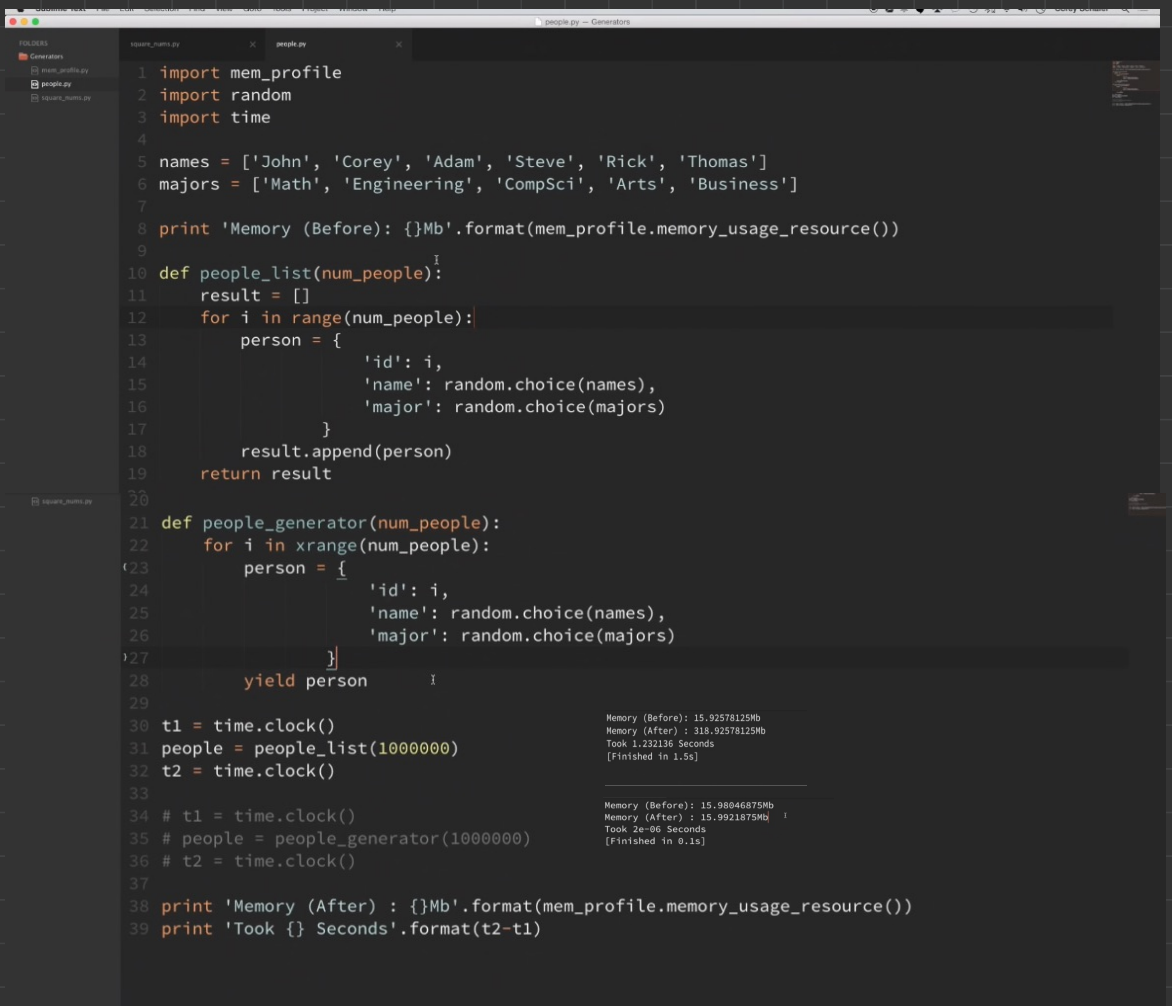
Try / except / else / finally

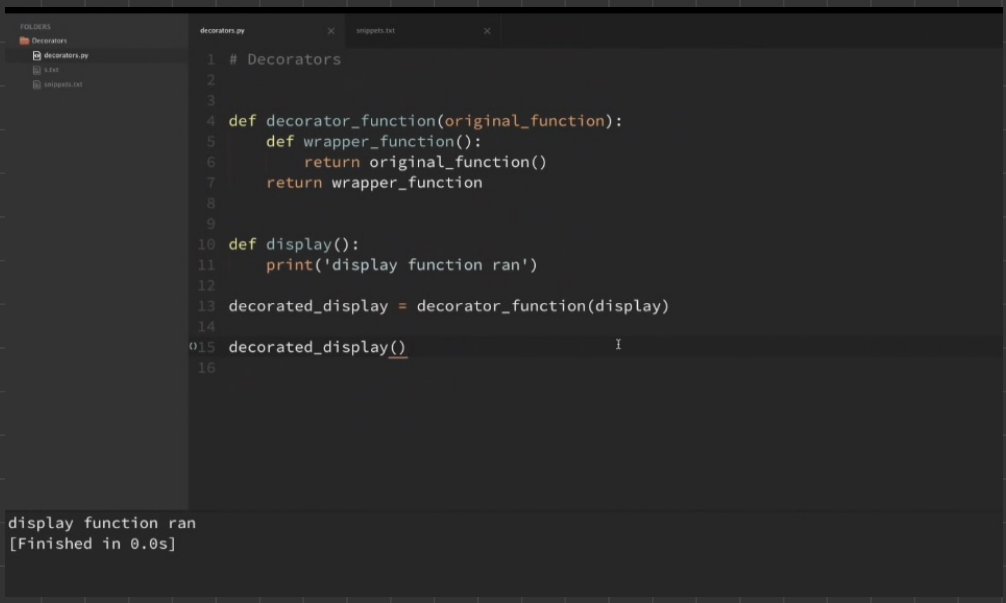
exceptions.py
test_file.txt

```
1
2  try:
3      f = open('currupt_file.txt')
4      if f.name == 'currupt_file.txt':
5          raise Exception
6  except FileNotFoundError as e:
7      print(e)
8  except Exception as e:
9      print('Error!')
10 else:
11     print(f.read())
12     f.close()
13 finally:
14     print("Executing Finally...")
```

```
Error!
Executing Finally...
[Finished in 0.1s]
```

Generators


people.py — Generators
FOLDERS
Generators
mem_profile.py
people.py
square_nums.py
square_nums.py
people.py
import mem_profile
import random
import time
names = ['John', 'Corey', 'Adam', 'Steve', 'Rick', 'Thomas']
majors = ['Math', 'Engineering', 'CompSci', 'Arts', 'Business']
print 'Memory (Before): {}Mb'.format(mem_profile.memory_usage_resource())
def people_list(num_people):
result = []
for i in range(num_people):
person = {
'id': i,
'name': random.choice(names),
'major': random.choice(majors)
}
result.append(person)
return result
def people_generator(num_people):
for i in xrange(num_people):
person = {
'id': i,
'name': random.choice(names),
'major': random.choice(majors)
}
yield person
t1 = time.clock()
people = people_list(1000000)
t2 = time.clock()
# t1 = time.clock()
# people = people_generator(1000000)
# t2 = time.clock()
print 'Memory (After) : {}Mb'.format(mem_profile.memory_usage_resource())
print 'Took {} Seconds'.format(t2-t1)
Memory (Before): 15.92578125Mb
Memory (After) : 318.92578125Mb
Took 1.232136 Seconds
[Finished in 1.5s]
Memory (Before): 15.98046875Mb
Memory (After) : 15.9921875Mb
Took 2e-06 Seconds
[Finished in 0.1s]


FOLDERS
Decorators
decorators.py
a.txt
snippets.txt
decorators.py
snippets.txt
1 # Decorators
2
3
4 def decorator_function(original_function):
5 def wrapper_function():
6 return original_function()
7 return wrapper_function
8
9
10 def display():
11 print('display function ran')
12
13 decorated_display = decorator_function(display)
14
15 decorated_display()
16
display function ran
[Finished in 0.0s]

Classes:

oop.py

```python
# Python Object-Oriented Programming


class Employee:

    def __init__(self, first, last, pay):
        self.first = first
        self.last = last
        self.pay = pay
        self.email = first + '.' + last + '@company.com'

    def fullname(self):
        return '{} {}'.format(self.first, self.last)


emp_1 = Employee('Corey', 'Schafer', 50000)
emp_2 = Employee('Test', 'User', 60000)

emp_1.fullname()
print(Employee.fullname(emp_1))
# print(emp_2.fullname())
```

employee.last or

```
Corey Schafer
[Finished in 0.0s]
```

```python
emp_1 = Employee('Corey', 'Schafer', 50000)
emp_2 = Employee('Test', 'User', 60000)

print(emp_1.__dict__)

# print(Employee.raise_amount)
# print(emp_1.raise_amount)
# print(emp_2.raise_amount)
```

```
{'first': 'Corey', 'pay': 50000, 'email': 'Corey.Schafer@company.com', 'last': 'Schafer'}
[Finished in 0.0s]
```

```python
class Developer(Employee):
    raise_amt = 1.10

    def __init__(self, first, last, pay, prog_lang):
        super().__init__(first, last, pay)
        Employee.__init__(self, first, last, pay)
```

same

oop.py exceptions.py

```python

class Employee:

    raise_amt = 1.04

    def __init__(self, first, last, pay):
        self.first = first
        self.last = last
        self.email = first + '.' + last + '@email.com'
        self.pay = pay

    def fullname(self):
        return '{} {}'.format(self.first, self.last)

    def apply_raise(self):
        self.pay = int(self.pay * self.raise_amt)


class Developer(Employee):
    raise_amt = 1.10

    def __init__(self, first, last, pay, prog_lang):
        super().__init__(first, last, pay)
        self.prog_lang = prog_lang

class Manager(Employee):

    def __init__(self, first, last, pay, employees=None):
        super().__init__(first, last, pay)
        if employees is None:
            self.employees = []
        else:
            self.employees = employees

    def add_emp(self, emp):
        if emp not in self.employees:
            self.employees.append(emp)

    def remove_emp(self, emp):
        if emp in self.employees:
            self.employees.remove(emp)

    def print_emps(self):
        for emp in self.employees:
            print('-->', emp.fullname())
```

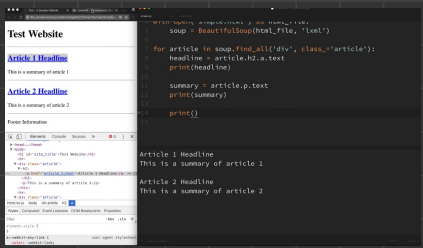
```
Corey.Schafer@email.com
Python
[Finished in 0.0s]
```

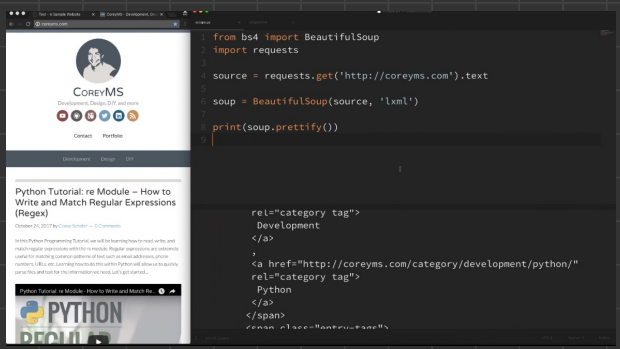
```python

class Employee:

    raise_amt = 1.04

    def __init__(self, first, last, pay):
        self.first = first
        self.last = last
        self.email = first + '.' + last + '@email.com'
        self.pay = pay

    def fullname(self):
        return '{} {}'.format(self.first, self.last)

    def apply_raise(self):
        self.pay = int(self.pay * self.raise_amt)

    def __repr__(self):
        return "Employee('{}', '{}', {})".format(self.first, self.last, self.pay)

    def __str__(self):
        return '{} - {}'.format(self.fullname(), self.email)


emp_1 = Employee('Corey', 'Schafer', 50000)
emp_2 = Employee('Test', 'Employee', 60000)

# print(emp_1)

# print(repr(emp_1))
# print(str(emp_1))

print(emp_1.__repr__())
print(emp_1.__str__())

```

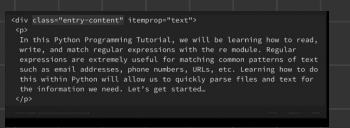
```
Employee('Corey', 'Schafer', 50000)
Corey Schafer - Corey.Schafer@email.com
[Finished in 0.0s]
```

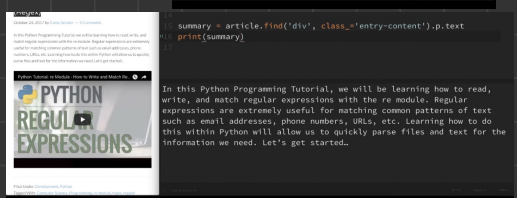
BeautifulSoup

```
Soup = BeautifulSoup(html_file, 'lxml')
match = Soup.title
      = Soup.title.text
      = Soup.div
      = Soup.find('div')
      = Soup.find('div', class_='footer')
```










```
vid_src = article.find('iframe', class_='youtube-player')
print(vid_src)
```

```
<iframe allowfullscreen="true" class="youtube-player" height="390"
src="http://www.youtube.com/embed/K8L6KVGG-7o?version=3&amp;rel=1&amp;fs=
1&amp;autohide=2&amp;showsearch=0&amp;showinfo=1&amp;iv_load_policy=1&amp
;wmode=transparent" style="border:0;" type="text/html" width="640"></
iframe>
```

We want the url id →

```
vid_src = article.find('iframe', class_='youtube-player')['src']
# print(vid_src)

vid_id = vid_src.split('/')[4]
vid_id = vid_id.split('?')[0]
print(vid_id)
```

```
K8L6KVGG-7o
```

```
from bs4 import BeautifulSoup
import requests

source = requests.get('http://coreyms.com').text

soup = BeautifulSoup(source, 'lxml')

for article in soup.find_all('article'):
    headline = article.h2.a.text
    print(headline)

    summary = article.find('div', class_='entry-content').p.text
    print(summary)
```

```
    try:
        vid_src = article.find('iframe', class_='youtube-player')['s

        vid_id = vid_src.split('/')[4]
        vid_id = vid_id.split('?')[0]

        yt_link = f'https://youtube.com/watch?v={vid_id}'
    except Exception as e:
        yt_link = None

    print(yt_link)
```

Working with json :

Simple example

```
''' JavaScript Object Notation '''
import json

people_string = '''
{
  "people": [
    {
      "name": "John Smith",
      "phone": "615-555-7164",
      "emails": ["johnsmith@bogusemail.com", "john.smith@work-place.com"],
      "has_license": false
    },
    {
      "name": "Jane Doe",
      "phone": "560-555-5153",
      "emails": null,
      "has_license": true
    }
  ]
}
'''
```

print json string

```
data = json.loads(people_string)

print(data)
```

```
{'people': [{'name': 'John Smith', 'phone': '615-555-7164', 'emails': ['johnsmith@bogusemail.com', 'john.smith@work-place.com'], 'has_license': False}, {'name': 'Jane Doe', 'phone': '560-555-5153', 'emails': None, 'has_license': True}]}
```

print only the names

```
data = json.loads(people_string)

for person in data['people']:
    print(person['name'])
```

```
John Smith
Jane Doe
```

delete users number

```
data = json.loads(people_string)

for person in data['people']:
    del person['phone']

new_string = json.dumps(data, indent=2, sort_keys=True)

print(new_string)
```

print it pretty json

indent=2, sort_keys=True

```
{"people": [{"name": "John Smith", "emails": ["johnsmith@bogusemail.com", "john.smith@work-place.com"], "has_license": false}, {"name": "Jane Doe", "emails": null, "has_license": true}]}
```

19.2.2. Encoders and Decoders

class json.**JSONDecoder**(**, object_hook=None, parse_float=*

Simple JSON decoder.

Performs the following translations in decoding by default:

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

```
''' JavaScript Object Notation '''
import json

with open('states.json') as f:
    data = json.load(f)

for state in data['states']:
    del state['area_codes']

with open('new_states.json', 'w') as f:
    json.dump(data, f, indent=2)
```

load json from file, delete area_code and write results to new file

indent=2

Data that we want to dump

dump to

```
import json
from urllib.request import urlopen

with urlopen("https://finance.yahoo.com/webservice/v1/symbols/allcurrencies/
        quote?format=json") as response:
        source = response.read()

    data = json.loads(source)

print(json.dumps(data, indent=2))
```

```
"list": {
  "meta": {
    "type": "resource-list",
    "start": 0,
    "count": 188
  },
  "resources": [
    {
      "resource": {
        "classname": "Quote",
        "fields": {
```

```
{'resource': {'classname': 'Quote', 'fields': {'name': 'USD/ADA', 'price': '165.098007', 'symbol': 'ADA=X', 'ts': '1510873502', 'type': 'currency', 'utctime': '2017-11-16T23:05:02+0000', 'volume': '0'}}}
{'resource': {'classname': 'Quote', 'fields': {'name': 'USD/CNY', 'price': '6.626300', 'symbol': 'CNY=X', 'ts': '1510891207', 'type': 'currency', 'utctime': '2017-11-17T04:00:07+0000', 'volume': '0'}}}
{'resource': {'classname': 'Quote', 'fields': {'name': 'USD/TTD', 'price': '6.729500', 'symbol': 'TTD=X', 'ts': '1510891190', 'type': 'currency', 'utctime': '2017-11-17T03:59:50+0000', 'volume': '0'}}}
{'resource': {'classname': 'Quote', 'fields': {'name': 'USD/BBD', 'price': '2.000200', 'symbol': 'BBD=X', 'ts': '1510845605', 'type': 'currency', 'utctime': '2017-11-16T15:20:05+0000', 'volume': '0'}}}
{'resource': {'classname': 'Quote', 'fields': {'name': 'USD/ECS', 'price': '25000.000000', 'symbol': 'ECS=X', 'ts': '1497964442', 'type': 'currency', 'utctime': '2017-02-24T19:27:22+0000', 'volume': '0'}}}
```

```
for item in data['list']['resources']:
    name = item['resource']['fields']['name']
    price = item['resource']['fields']['price']
    print(name, price)
```

results

```
USD/DOP 47.200001
USD/ALL 113.230003
USD/HTG 61.639999
USD/AMD 486.920013
USD/KMF 423.500000
USD/MRO 351.000000
USD/HRK 6.400500
USD/HTG 61.639999
USD/KHR 4031.000000
USD/PHP 50.810001
CYP=X 0.000000
```

```
# print(json.dumps(data, indent=2))

usd_rates = dict()

for item in data['list']['resources']:
    name = item['resource']['fields']['name']
    price = item['resource']['fields']['price']
    usd_rates[name] = price

print(50 * float(usd_rates['USD/INR']))
```

```
3240.2500000000005
```