Python

Python?????????????????????????????·?????????????1991??Python?ABC???????????????????????LISP???Python?????????????????????????C++?Java?Python?????????????????

- Learning
- pip
- maxmind_db_ip_geolocator.py
- Tips
- Functions
- THSRC API
- JSON
- Datetime
- One-Liners
- List
- String

Learning

Online Tutorials

- The Python Tutorial
- The Hitchhiker’s Guide to Python
- How to make an awesome command line tool like MyCLI, PgCLI
- A Beginner's Guide to Programming
- Beyond the Basic Stuff with Python
- 100 Page Python Intro
- Understanding Python re(gex)
- Everything You Need to Learn Python Programming (?????)
- Python Tutorial (??????)
- Learn Python Programming – Everything You Need to Know (??)

Web scraping

- A guide to web scraping in Python using Beautiful Soup

Binance Public API Connector Python

- GitHub - binance/binance-connector-python: a simple connector to Binance Public API
- Binance Public API Connector Python — binance-connector documentation
- ?? Python ? Binance ????????????????? vocus

Developers Forum

- DEV Community

pip

Installation

NOTE: The following commands still require internet connection.

get-pip.py

```
# Latest version of python
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

# For python 2.7.x
curl https://bootstrap.pypa.io/2.7/get-pip.py -o get-pip.py

# Offline Install the pip
sudo pyhon get-pip.py
```

Module mkdocs

```
# Downloading the source files required for the module mkdocs, which requires an internet.
pip download -d <output-dir> mkdocs

# Offline install the module mkdocs
pip install <output-dir>/*.whl
```

Proxy server

```
pip install --proxy http://<usr_name>:<password>@<proxyserver_name>:<port#> <pkg_name>
```

```
pip config set global.proxy http://account:password@xxx.com.tw:8080
pip config set global.trusted-host pypi.python.org\npypi.org\nfiles.pythonhosted.org
```

Command

List installed modules

```
sudo pip list
```

Upgrade module

```
sudo pip install --upgrade <MODULENAME>
```

Export the list of installed modules

```
pip freeze > requirements.txt
```

Install modules in requirements.txt

```
pip install -r requirements.txt
```

maxmind_db_ip_geolocator.py

Original Post: Python Basics for Hackers, Part 4: How to Find the Exact Location of any IP Address

```
#! /usr/bin/python

#Hello fellow hackers! My name is Defalt
#I built a very basic version of this tool a long time ago and recently did a re-write
#The first re-write had some awkward usage of the argparse module, so this update is going to fix it
#Original version: http://pastebin.com/J5NLnThL
#This will query the MaxMind database to get an approximate geolocation of an IP address
#Happy hacking! -Defalt

import sys
import socket
import urllib
import gzip
import os
try:
    import pygeoip
except ImportError:
    print '[!] Failed to Import pygeoip'
    try:
        choice = raw_input('[*] Attempt to Auto-install pygeoip? [y/N] ')
    except KeyboardInterrupt:
        print '\n[!] User Interrupted Choice'
        sys.exit(1)
    if choice.strip().lower()[0] == 'y':
        print '[*] Attempting to Install pygeoip... ',
        sys.stdout.flush()
        try:
            import pip
```

```
            pip.main(['install', '-q', 'pygeoip'])
            import pygeoip
            print '[DONE]'
        except Exception:
            print '[FAIL]'
            sys.exit(1)
    elif choice.strip().lower()[0] == 'n':
        print '[*] User Denied Auto-install'
        sys.exit(1)
    else:
        print '[!] Invalid Decision'
        sys.exit(1)

class Locator(object):
    def __init__(self, url=False, ip=False, datfile=False):
        self.url = url
        self.ip = ip
        self.datfile = datfile
        self.target = ''
    def check_database(self):
        if not self.datfile:
            self.datfile = '/usr/share/GeoIP/GeoLiteCity.dat'
        else:
            if not os.path.isfile(self.datfile):
                print '[!] Failed to Detect Specified Database'
                sys.exit(1)
            else:
                return
        if not os.path.isfile(self.datfile):
            print '[!] Default Database Detection Failed'
            try:
                choice = raw_input('[*] Attempt to Auto-install Database? [y/N] ')
            except KeyboardInterrupt:
                print '\n[!] User Interrupted Choice'
                sys.exit(1)
            if choice.strip().lower()[0] == 'y':
                print '[*] Attempting to Auto-install Database... ',
                sys.stdout.flush()
                if not os.path.isdir('/usr/share/GeoIP'):
                    os.makedirs('/usr/share/GeoIP')
```

```
        try:
            urllib.urlretrieve('http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz',
'/usr/share/GeoIP/GeoLiteCity.dat.gz')
        except Exception:
            print '[FAIL]'
            print '[!] Failed to Download Database'
            sys.exit(1)
        try:
            with gzip.open('/usr/share/GeoIP/GeoLiteCity.dat.gz', 'rb') as compressed_dat:
                with open('/usr/share/GeoIP/GeoLiteCity.dat', 'wb') as new_dat:
                    new_dat.write(compressed_dat.read())
        except IOError:
            print '[FAIL]'
            print '[!] Failed to Decompress Database'
            sys.exit(1)
        os.remove('/usr/share/GeoIP/GeoLiteCity.dat.gz')
        print '[DONE]\n'
    elif choice.strip().lower()[0] == 'n':
        print '[!] User Denied Auto-Install'
        sys.exit(1)
    else:
        print '[!] Invalid Choice'
        sys.exit(1)
def query(self):
    if not not self.url:
        print '[*] Translating %s: ' %(self.url),
        sys.stdout.flush()
        try:
            self.target += socket.gethostbyname(self.url)
            print self.target
        except Exception:
            print '\n[!] Failed to Resolve URL'
            return
    else:
        self.target += self.ip
    try:
        print '[*] Querying for Records of %s...\n' %(self.target)
        query_obj = pygeoip.GeoIP(self.datfile)
        for key, val in query_obj.record_by_addr(self.target).items():
            print '%s: %s' %(key, val)
```

```
            print '\n[*] Query Complete!'
        except Exception:
            print '\n[!] Failed to Retrieve Records'
            return

if __name__ == '__main__':
    import argparse
    parser = argparse.ArgumentParser(description='IP Geolocation Tool')
    parser.add_argument('--url', help='Locate an IP based on a URL', action='store', default=False, dest='url')
        parser.add_argument('-t', '--target', help='Locate the specified IP', action='store', default=False, dest='ip')
        parser.add_argument('--dat', help='Custom database filepath', action='store', default=False, dest='datfile')
    args = parser.parse_args()
    if ((not not args.url) and (not not args.ip)) or ((not args.url) and (not args.ip)):
        parser.error('invalid target specification')
    try:
        locate = Locator(url=args.url, ip=args.ip, datfile=args.datfile)
        locate.check_database()
        locate.query()
    except Exception:
        print '\n[!] An Unknown Error Occured'
```

Tips

Find all installed modules

```
help("modules");
```

Virtual Environment

- How to Install and Manage Python Versions in Linux - Make Tech Easier

Python 3.4+

```
mkdir myproject
cd myproject
python -m venv env

# Activate the venv
source env/bin/activate

# Delete the venv
deactivate
rm -rf env
```

With virtualenv and virtualenvwrapper

```
# Installing virtualenv and virtualenvwrapper
sudo pip install virtualenv virtualenvwrapper

# Update the profile ~/.bashrc
# Add the  following lines

# Python virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

```
# Reload the profile
source ~/.bashrc

# Creating python virtual environment
# The py3cv3 is a self-defined name
mkvirtualenv py3cv3 -p python3

# Enter the specified virtual environment
workon py3cv3

# Exit the the specified virtual environment
deactivate

# List all of the environments.
lsvirtualenv

# Remove an environment
rmvirtualenv py3cv3
```

Timestamp

```
timestamp = datetime.datetime.now()
print("It is {}".format(timestamp.strftime("%A %d %B %Y %I:%M:%S%p")))
```

Math

```
total += 1
```

If-then

```
# Boolean, none
if motion is not None:
if not flag:

# Number
if delay > 0:
if delay == 0:
if total > frameCount:
```

```
# String
if "blue" in style:
if authors.startswith('['):
    authors = authors.lstrip('[').rstrip(']')

# One-liner
def doi_url(d): return f'http://{d}' if d.startswith('doi.org') else f'http://doi.org/{d}'

# Multiple conditions
temperature = 25
if temperature > 30:
    print('Hot')
elif temperature > 20 and temperature <= 30:
    print('Warm')
else:
    print('Cool')

# Reverse the True
temperature = 15
if not temperature > 20:
    print('Cool')
#
temperature = 25
humidity = 55
rain = 0
if temperature > 30 or humidity < 70 and not rain > 0:
    print('Dry conditions')
```

Command Arguments

- Manual: https://docs.python.org/3/library/argparse.html

```
import argparse
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--interval", required=False,
        help="Seconds to Interval (Default:30)", default="30", type=int)
ap.add_argument("-o", "--output", required=False,
        help="Path to Output Logs (Default:std-out)")
ap.add_argument("mac",
```

```
        help="MAC address of LYWSD02 device", nargs="+")
args = vars(ap.parse_args())

# Usage
intv = args["interval"]
logfile = args["output"]
```

```
#
# Nagios2 HTTP proxy test
#
# usage: check_http_proxy --proxy=proxy:port --auth=user:pass --url=url --timeout=10 --warntime=5 --
expect=content

import sys
import getopt

def get_cmdline_cfg():
    try:
        opts, args = getopt.getopt(
            sys.argv[1:],
            "p:a:t:w:e:u:",
            ["proxy=", "auth=", "timeout=", "warntime=", "expect=", "url="]
        )
    except getopt.GetoptError, err:
        print("SCRIPT CALLING ERROR: {0}".format(str(err)))

    ### Build cfg dictionary
    cfg = {}
    for o, a in opts:
        if o in ("-p", "--proxy"):
            cfg["proxy"] = a
        elif o in ("-a","--auth"):
            cfg["auth"] = a
        elif o in ("-t","--timeout"):
            cfg["timeout"] = float(a)
        elif o in ("-w","--warntime"):
            cfg["warntime"] = float(a)
        elif o in ("-e","--expect"):
            cfg["expect"] = a
        elif o in ("-u","--url"):
```

```
        cfg["url"] = a

    # These are required
    for req_param in ("url", "proxy"):
        if req_param not in cfg:
            print("Missing parameter: {0}".format(req_param))

    return cfg

# Usage
if __name__ == '__main__':
    cfg = get_cmdline_cfg()

    if "auth" in cfg:
        proxy_url = "http://{auth}@{proxy}/".format(**cfg)
    else:
        proxy_url = "http://{proxy}/".format(**cfg)
```

Functions

Print and Log

```
def print_f(*msg):
    '''print and log!'''
    # import datetime for timestamps
    import datetime as dt
    # convert input arguments to strings for concatenation
    message = []
    for m in msg:
        message.append(str(m))
    message = ' '.join(message)
    # append to the log file
    with open('/tmp/test.log','a') as log:
        log.write(f'{dt.datetime.now()} | {message}\n')
    # print the message using the copy of the original print function to stdout
    print(message)

print_f('Test Message')
```

Sendmail via SMTP

```
def send_message(body, subject, to_addr):
    import smtplib
    from email.message import EmailMessage
    smtp_user = "your-smtp-user"
    smtp_pass = "your-smtp-pass"
    smtp_server = "smtp-relay.your.server"
    smtp_port = "587"

    msg = EmailMessage()
    msg['Subject'] = subject
    msg['From'] = smtp_user
    msg['To'] = to_addr
    msg.set_content(body)
```

```
    with smtplib.SMTP(smtp_server, smtp_port) as smtp:
        smtp.login(smtp_user, smtp_pass)
        smtp.send_message(msg)

debug = send_message("This is plain TEXT email", "Test from SMTP", "alang.hsu@gmail.com")
print(debug)
```

THSRC API

Links

- TDX ???????
- TDX ????
- TDX????????API?????????
- TDX - ?? API ??
- ??API????????
- API ??????
- MOTC Transport API V2 (????????)

API ????

1. Client Id: ??????
2. Client Secret: ??????
3. Access Token: ?? HTTP POST ??Client Id ? Client Secret ??????? Access Token?

Get Access Token

```
curl --request POST \
  --url 'https://tdx.transportdata.tw/auth/realms/TDXConnect/protocol/openid-connect/token' \
  --header 'content-type: application/x-www-form-urlencoded' \
  --data grant_type=client_credentials \
  --data client_id=YOUR_CLIENT_ID \
  --data client_secret=YOUR_CLIENT_SECRET \
```

???????

- access_token? ????API???token????JWT
- expires_in?token?????????????86400?(1?)
- token_type?token??????"Bearer"

Case: ??????????????????????

API?

1. `/v2/Rail/THSR/DailyTimetable/Station/{StationID}/{TrainDate}`
   - ??????,????????
   - ???????????????
2. `/v2/Rail/THSR/AvailableSeatStatus/Train/OD/{OriginStationID}/to/{DestinationStationID}/TrainDate/{TrainDate}`
   - ????[??], [???]???????????
   - ???????????

> NOTE: ?????????????????????????????????????? 10, 16, 22 ??

JSON

JSON to dict

> `json.loads` ??????; `json.load` ?????

```
import json

person = '{"name": "Bob", "languages": ["English", "French"]}'
person_dict = json.loads(person)

# Output: {'name': 'Bob', 'languages': ['English', 'French']}
print( person_dict)

# Output: ['English', 'French']
print(person_dict['languages'])
```

Dict to JSON

```
import json

person_dict = {'name': 'Bob',
'age': 12,
'children': None
}
person_json = json.dumps(person_dict)

# Output: {"name": "Bob", "age": 12, "children": null}
print(person_json)
```

Read JSON file

```
import json

with open('path_to_file/person.json', 'r') as f:
  data = json.load(f)
```

```
# Output: {'name': 'Bob', 'languages': ['English', 'French']}
print(data)
```

Write JSON file

> `json.dump` ?????; `json.dumps` ???????

```
import json

person_dict = {"name": "Bob",
"languages": ["English", "French"],
"married": True,
"age": 32
}

with open('person.txt', 'w') as json_file:
  json.dump(person_dict, json_file)
```

Print JSON

```
import json

person_string = '{"name": "Bob", "languages": "English", "numbers": [2, 1.6, null]}'

# Getting dictionary
person_dict = json.loads(person_string)

# Pretty Printing JSON string back
print(json.dumps(person_dict, indent = 4, sort_keys=True))
```

Access JSON

```
import json

json_data = '''
{
   "students": [
      {
```

```
            "name": "David",
            "age": 19,
            "grades": {
                "math": 90,
                "english": 87
            }
        },
        {
            "name": "Harry",
            "age": 21,
            "grades": {
                "math": 85,
                "english": 95
            }
        }
    ]
}
'''

# Parse JSON Data
data = json.loads(json_data)

# To access a large dataset we can use `for loop`
for student in data["students"]:
    name = student["name"]
    math_mark = student["grades"]["math"]
    english_mark = student["grades"]["english"]
    average_mark = (math_mark + english_mark) / 2
    print(f"{name}, Avarage Marks: {average_mark:.2f}")



# Output:
# David, Average Marks: 88.50
# Harry, Average Marks: 90.00
```

```
import json

original_data_file="students_data.json"
updated_data_file="students_data_updated.json"
```

```
# reading `JSON file`
with open(original_data_file,"r") as file:
   students_result = json.load(file)

# Updating JSON Data
for student in students_result['students']:
    print(student['name'])

    if student['name'] == "Kabir":
        student['name'] = "John"

    grades = student['grades']
    avarage_mark= sum(grades.values()) / len(grades)
    student['avarage_mark'] = avarage_mark

# Saving updated data into a new file
with open(updated_data_file,"w") as file:
    json.dump(students_result,file,indent=4)
```

Get JSON from URL

```
import requests, json

# Response will be saved here
weather_data="weather_data.json"

# Request to `openweathermap` API
api_key = "6423af6e554f98cf1e6b8c6a7700986b"    #REPLACE_WITH_YOUR_API_KEY
location = "Dhaka"
url = f"https://api.openweathermap.org/data/2.5/weather?q={location}&appid={api_key}&units=metric"

# Response
response = requests.get(url)

# Get `Place` and `Temperature` from the Response
if response.status_code == 200:
    json_data = response.json()
    print(f"Place: {json_data['name']}, Temperature: {json_data['main']['temp']} celsius")
else:
```

```
    print(f"Request failed with status code {response.status_code}")

# Save the Response to a file
with open(weather_data,"w") as file:
        json.dump(json_data,file,indent=4)



# Output:
# Place: Dhaka, Temperature: 27.99 celsius
```

```
# Handling a JSONDecodeError in Python
from json import JSONDecodeError
import requests
resp = requests.get('https://reqres.in/api/users/page4')
try:
    resp_dict = resp.json()
except JSONDecodeError:
    print('Response could not be serialized')
```

Data Type

> ?? `json.loads` ??????????????????? dict ?? array??????JSON ??????

JSON	Python
object	dict
array	list
string	str
number (integer)	int
number (real)	float
true	True
false	False
null	N

Library

jsonpath-ng

```
{
  "employees": [
    {
      "id": 1,
      "name": "Pankaj",
      "salary": "10000"
    },
    {
      "name": "David",
      "salary": "5000",
      "id": 2
    }
  ]
}
```

```
import json
from jsonpath_ng import jsonpath, parse

with open("db.json", 'r') as json_file:
    json_data = json.load(json_file)

print(json_data)

jsonpath_expression = parse('employees[*].id')

for match in jsonpath_expression.find(json_data):
    print(f'Employee id: {match.value}')
```

```
{'employees': [{'id': 1, 'name': 'Pankaj', 'salary': '10000'}, {'name': 'David', 'salary': '5000', 'id': 2}]}
Employee id: 1
Employee id: 2
```

Datetime

String to Datetime

```
from datetime import datetime


date_str = '09-19-2022'


date_object = datetime.strptime(date_str, '%m-%d-%Y').date()
print(type(date_object))
print(date_object)  # printed in default format

# Output:
# <class 'datetime.date'>
# 2022-09-19
```

```
from datetime import datetime


time_str = '13::55::26'
time_object = datetime.strptime(time_str, '%H::%M::%S').time()
print(type(time_object))
print(time_object)

# Output:
# <class 'datetime.time'>
# 13:55:26
```

```
from datetime import datetime
import locale


locale.setlocale(locale.LC_ALL, 'de_DE')
date_str_de_DE = '16-Dezember-2022 Freitag'  # de_DE locale
datetime_object = datetime.strptime(date_str_de_DE, '%d-%B-%Y %A')
print(type(datetime_object))
print(datetime_object)
```

```
# Output:
# <class 'datetime.datetime'>
# 2022-12-16 00:00:00
```

date

```
import datetime
d = datetime.date(2020,1,1)   # 2020-01-01
```

```
import datetime
today = datetime.date.today()
print(today)                # 2021-10-19
print(today.year)           # 2021
print(today.month)          # 10
print(today.day)            # 19
print(today.weekday())      # 1    ( □□□□□□□□□□ 1 )
print(today.isoweekday())   # 2    ( □□□□□□□□□□ 2 )
print(today.isocalendar())  # (2021, 42, 2)  ( □□□□□□□□□□□□□ 2 )
print(today.isoformat())    # 2021-10-19
print(today.ctime())        # Tue Oct 19 00:00:00 2021
print(today.strftime('%Y.%m.%d'))    # 2021.10.19

newDay = today.replace(year=2020)
print(newDay)               # 2020-10-19
```

```
import datetime
d1 = datetime.date(2020, 6, 24)
d2 = datetime.date(2021, 11, 24)
print(abs(d1-d2).days)      # 518
```

time

```
import datetime
thisTime = datetime.time(12,0,0,1)
print(thisTime)   # 12:00:00.000001
```

```
import datetime
thisTime = datetime.time(14,0,0,1,tzinfo=datetime.timezone(datetime.timedelta(hours=8)))
print(thisTime)             # 14:00:00.000001+08:00
```

```
print(thisTime.isoformat())   # 14:00:00.000001+08:00
print(thisTime.tzname())      # UTC+08:00
print( thisTime.strftime('%H:%M:%S'))   # 14:00:00

newTime = today.replace(hour=20)
print(newTime)               # 20:00:00.000001+08:00
```

datetime

```
import datetime
thisTime = datetime.datetime(2020,1,1,20,20,20,20)
print(thisTime)    # 2020-01-01 20:20:20.000020
```

```
import datetime
print(datetime.datetime.today())    # 2021-10-19 06:15:46.022925
print(datetime.datetime.now(tz=datetime.timezone(datetime.timedelta(hours=8))))
# 2021-10-19 14:15:46.027982+08:00
print(datetime.datetime.utcnow())   # 2021-10-19 06:15:46.028630
```

```
import datetime
now = datetime.datetime.now(tz=datetime.timezone(datetime.timedelta(hours=8)))
print(now)              # 2021-10-19 14:25:46.962975+08:00
print(now.date())       # 2021-10-19
print(now.time())       # 14:25:46.962975
print(now.tzname())      # UTC+08:00
print(now.weekday())     # 1
print(now.isoweekday())   # 2
print(now.isocalendar())  # (2021, 42, 2)
print(now.isoformat())    # 2021-10-19 14:25:46.962975+08:00
print(now.ctime())       # Tue Oct 19 14:48:38 2021
print(now.strftime('%Y/%m/%d %H:%M:%S'))  # 2021/10/19 14:48:38
print(now.timetuple())    # time.struct_time(tm_year=2021, tm_mon=10, tm_mday=19, tm_hour=16,
tm_min=8, tm_sec=6, tm_wday=1, tm_yday=292, tm_isdst=-1)
```

> - %Y: 4-digit year.
> - %y: 2-digit year.
> - %m: Month as a zero-padded number.
> - %d: Day of the month as a zero-padded number.

> - %H: Hour (24-hour clock) as a zero-padded number.
> - %M: Minute as a zero-padded number.
> - %S: Second as a zero-padded number.

timedelta

??/????

```
import datetime
today = datetime.datetime.now()
yesterday = today - datetime.timedelta(days=1)
tomorrow = today + datetime.timedelta(days=1)
nextweek = today + datetime.timedelta(weeks=1)
print(today)       # 2021-10-19 07:01:22.669886
print(yesterday)   # 2021-10-18 07:01:22.669886
print(tomorrow)    # 2021-10-20 07:01:22.669886
print(nextweek)    # 2021-10-26 07:01:22.669886
```

Timezone

```
import datetime
tzone = datetime.timezone(datetime.timedelta(hours=8))
now = datetime.datetime.now(tz=tzone)
print(now)    # 2021-10-19 15:07:51.128092+08:00
```

```
from datetime import datetime, timezone


# Get the current time in UTC
utc_time = datetime.now(timezone.utc)


print(utc_time)
```

```
from datetime import datetime
import pytz


timezone = pytz.timezone("America/New_York")


current_time_in_timezone = datetime.now(timezone)
```

```
print(current_time_in_timezone)
```

Sleep

```
import time

time.sleep(5) # Pauses the code for 5 seconds
```

Timestamp

Get Current Time in Milliseconds

```
milliseconds_since_epoch = time.time() * 1000
```

Get Current Timestamp

```
current_timestamp = time.time()

print(current_timestamp)
```

Timestamp to a human-readable date

```
timestamp = time.time()

readable_date = datetime.fromtimestamp(timestamp)

print(readable_date)
```

Time Diff.

```
time1 = datetime.now()

# ... some operations ...

time2 = datetime.now()

difference = time2 - time1
```

```
print(difference)
```

```
start_time = time.time()

# ... some operations ...

end_time = time.time()

elapsed_time = end_time - start_time

print(f"Time elapsed: {elapsed_time} seconds")
```

One-Liners

1) Multiple Variable Assignment

```
# Traditional way
a = 1
b = "ok"
c = False

# Pythonic way
a, b, c = 1, "ok", False

# Result
print(a, b, c)
# Show: 1 ok False
```

2) Variable Swap

```
# Traditional way
a = 1
b = "ok"

c = a
a = b
b = c

# Pythonic way
a, b = 1, "ok"
a, b = b, a

# Result
print(a, b)
# Shows: ok 1
```

```
# Pythonic way
a, b, c, d = 1, "ok", True, ["i", "j"]
```

```
a, b, c, d = c, a, d, b

# Result
print(a, b, c, d)
# Shows: True 1 ["i", "j"] ok
```

3) Variable Conditional Assignment

```
x = 3

# Traditional way
if x % 2 == 1:
result = f"{x} is odd"
else:
    result = f"{x} is even"

# Pythonic way
result = f"{x} " + ("is odd" if x % 2 == 1 else "is even")

# Result
print(result)
# Shows: 3 is odd
```

4) Presence of a Value in a List

```
pet_list = ["cat", "dog", "parrot"]

# Traditional way
found = False
for item in my_list:
    if item == "cat":
        found = True
        break

# Pythonic way
found = "cat" in pet_list

# Result
print(found)
# Shows: True
```

```
pet_dict = {"cat": "Mitchi", "dog": "Max", "parrot": "Pepe"}
found = "cat" in pet_dict
print(found)
# Shows: True
```

5) Operations on Lists

```
my_list = [1, 2, 3, 4, 5]

# Traditional way
max_value = 0
for value in my_list:
if value > max_value:
max_value = value

# Pythonic way
max_value = max(my_list)

# Result
print(max_value)
# Shows: 5
```

6) List Creation with Duplicate Values

```
size = 10

# Traditional way
my_list = []
for i in range(size):
    my_list.append(0)

# Pythonic way
my_list = [0] * size

# Result
print(my_list)
# Shows: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
my_list = [1, 2] * 5

# Result: [1, 2, 1, 2, 1, 2, 1, 2, 1, 2]
```

```
my_tuple = (1, 2) * 5
print(my_tuple)
# Shows: (1, 2, 1, 2, 1, 2, 1, 2, 1, 2)
```

7) List Creation with Sequential Values

```
count = 10

# Traditional way
my_list = []
for i in range(count):
    my_list.append(i)

# Pythonic way
my_list = list(range(count))

# Result
print(my_list)
# Shows: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
# List with odd values
my_list = list(range(1, 10, 2))
print(my_list)
# Shows: [1, 3, 5, 7, 9]
```

```
# List with descending values and negative values
my_list = list(range(5, -5, -1))
print(my_list)
# Shows: [5, 4, 3, 2, 1, 0, -1, -2, -3, -4]
```

```
my_set = set(range(count))
my_tuple = tuple(range(count))

# Result
print(my_set)
```

```
# Shows: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
print(my_tuple)
# Shows: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

8) List Creation with a Loop

```
count = 4


# Traditional way
my_list = []
for i in range(count):
my_list.append(count**i)

# Pythonic way
my_list = [count**x for x in range(count)]

# Result
print(my_list)
# Shows: [1, 4, 16, 64]
```

```
my_set = set(count**x for x in range(count))
print(my_set)
# Shows: {1, 4, 16, 64}
```

9) List Creation with Conditions

```
users = [("Megan", 56),
("Karen", 32),
("Chad", 28),
("Brent", 44)]

# Traditional way
young_users = []
for user in users:
    if (user[1] < 35):
        young_users.append(user[0])

# Pythonic way
young_users = [x for x, y in users if y < 35]
```

```
# Result
print(young_users)
# ["Karen", "Chad"]
```

10) Reading a File Line by Line

```
# Traditional way
lines = []
with open(filename) as file:
for count, line in enumerate(file):
lines.append(f"Line {count + 1}: " + line.strip())

# Pythonic way
with open(filename) as file:
lines = [f"Line {count + 1}: " + line.strip() for count, line in enumerate(file)]
```

List

list.append()

```
numbers = [1, 2, 3, 4]
numbers.append(5)
print(numbers)

# output: [1, 2, 3, 4, 5]
```

list.insert()

```
animals = ["cat", "dog", "fish"]
animals.insert(1, "monkey")
print(animals)

# output: ["cat", "monkey", "dog", "fish"]

animals = ["cat", "dog", "fish"]
animals.insert(200, "monkey")
print(animals)

# output: ["cat", "dog", "fish", "monkey"]
```

list.extend()

```
things = ["John", 42, True]
other_things = [0.0, False]
things.append(other_things)
print(things)

# output: ["John", 42, True, [0.0, False]]

things = ["John", 42, True]
other_things = [0.0, False]
things.extend(other_things)
print(things)
```

```
# output: ["John", 42, True, 0.0, False]
```

list.remove()

```
booleans = [True, False, True, True, False]

booleans.remove(False)   # Removes the first False value
print(booleans)

# output: [True, True, True, False]

booleans.remove(False)   # Removes the other False value
print(booleans)

# output: [True, True, True]

booleans.remove(False)   # ValueError! No more False values to remove
```

list.pop()

```
fruits = ["apple", "orange", "banana", "peach"]
last_fruit = fruits.pop()  # takes the last element
print(last_fruit)

# output: "peach"

second_fruit = fruits.pop(1)  # takes the second element ( = index 1)
print(second_fruit)

# output: "orange"

print(fruits)  # only fruits that have not been "popped"
               # are still in the list

# output: ["apple", "banana"]
```

list.clear()

```
decimals = [0.1, 0.2, 0.3, 0.4, 0.5]
decimals.clear()  # remove all values!
print(decimals)

# output: []
```

list.count()

```
grades = [7.8, 10.0, 7.9, 9.5, 10.0, 6.5, 9.8, 10.0]
n = grades.count(10.0)
print(n)

# output: 3
```

list.index()

```
friends = ["John", "James", "Jessica", "Jack"]
position = friends.index("Jessica")
print(position)

# output: 2
```

list.sort() and list.reverse()

```
values = [10, 4, -2, 1, 5]

values.reverse()
print(values)  # list is reversed

# output: [5, 1, -2, 4, 10]

values.sort()
print(values)  # list is sorted

# output: [-2, 1, 4, 5, 10]
```

```
values = [10, 4, -2, 1, 5]

values.sort(reverse=True)
```

```
print(values)  # list is sorted in reverse order


# output: [10, 5, 4, 1, -2]
```

list.copy()

```
values_01 = [1, 2, 3, 4]
values_02 = values_01  # not an actual copy: same list object!

values_02.append(5)  # we modify the "values_02" list...
print(values_01)     # ... but changes appear also in "values_01"
                     #     because they reference the same list!

# output: [1, 2, 3, 4, 5]



values_01 = [1, 2, 3, 4]
values_02 = values_01.copy()  # create an independent copy!

values_02.append(5)  # we modify the "values_02" list...
print(values_01)     # ... and changes DO NOT appear in "values_01"
                     #     because it is a copy!

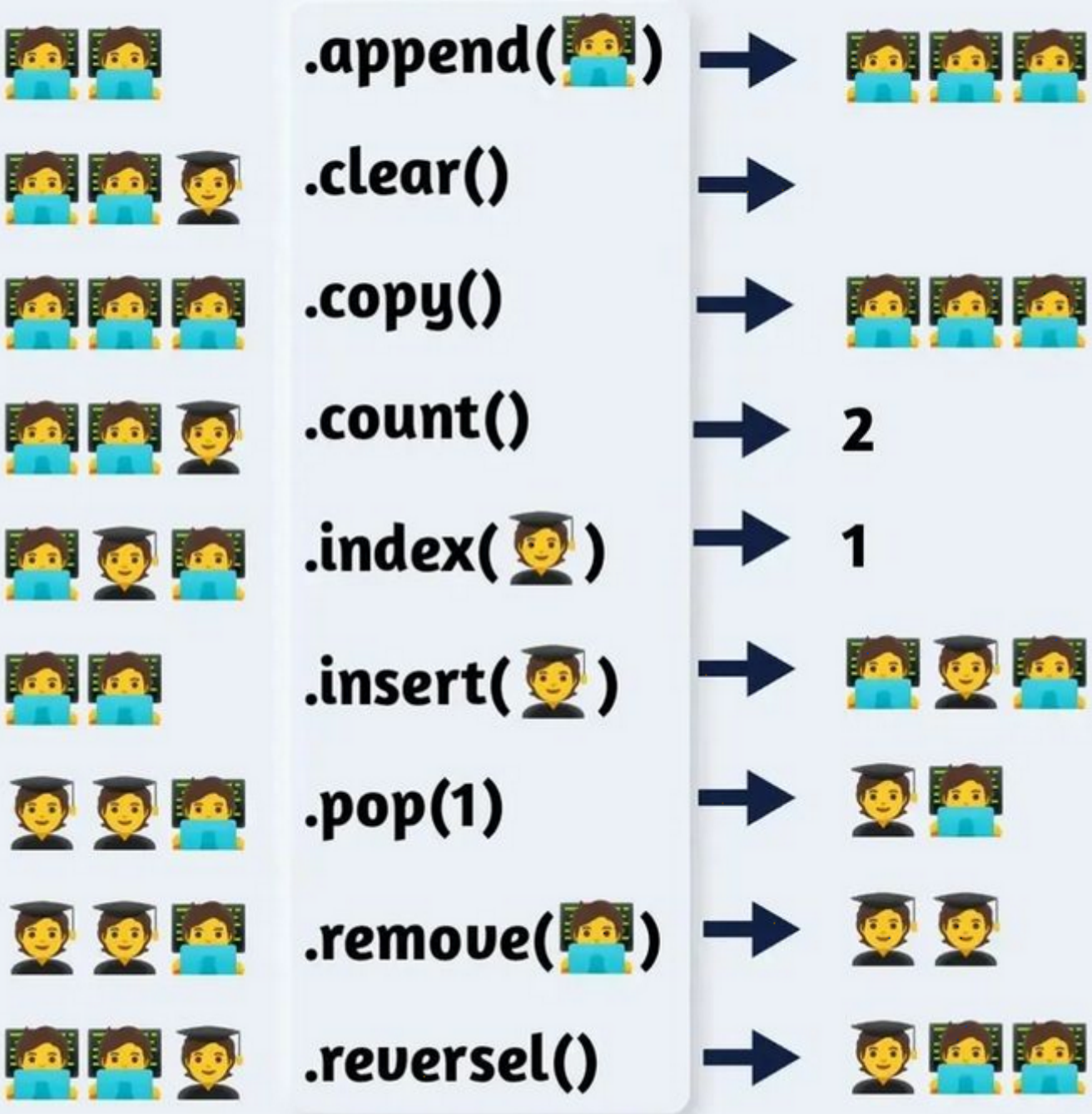
# output: [1, 2, 3, 4]
```

Cheat Sheet

IMPORTANT METHODS IN PYTHON

SET	LIST	DICTIONARY
add()	append()	copy()
clear()	copy()	clear()
pop()	count()	fromkeys()
union()	insert()	items()
issuperset()	reverse()	get()
issubset()	remove()	keys()
intersection()	sort()	pop()
difference()	pop()	values()
isdisjoint()	extend()	update()
setdiscard()	index()	isetdefault()
copy()	clear()	popitem()

PYTHON LIST CHEATSHEET



String

Sample codes

```
secret_password = 'jhk7GSH8ds'
print('Password hint: the third letter of your password is ' + secret_password[2])
```

```
# Escaping characters
introduction = 'Hello, I\'m John!'
print(introduction)

# Joining strings
user_age = 28
user_name = 'John'
greeting = user_name + ', you are ' + str(user_age) + '!'
print(greeting)
```

Cheat Sheet

String Methods

String Methods In Python



str.lower()	Convert all character in the string to lowercase.
str.upper()	Convert all character in the string to uppercase.
str.capitalize()	Capitalizes the first character of the string.
str.title()	Capitalizes the first character of each word in the string.
str.strip()	Removes leading and trailing whitespaces from the string.
str.startwith()	Checks if the string starts with a specific substring.
str.endwith()	Checks if the string ends with a specific substring.
str.replace()	Replaces occurrences of a substring with another substring.
str.split()	Splits the string into a list of substrings based on a declimiter.
str.join()	Joins elements of an iterable into a string with a specific separator.
str.find()	Finds the first occurrence of a substring in the string.
str.count()	Counts the occurrence of a substring in the string.
str.isapha()	Checks if all character in the string are alphabetic.
str.isdigit()	Checks if all character in the string are digits.
str.isalnum()	Checks if all character in the string are alphanumeric.
str.islower()	Checks if all character in the string are lowercase.
str.isupper()	Checks if all character in the string are uppercase.
str.isspace()	Checks if the string contains only white space characters.
str.isnumeric()	Checks if all character in the string are numeric.