Coursework overview

2021-2

Module assessment overview

Portfolio quizzes – 40%

Coursework – 60%

- Large case study
- Many components are lab exercises
- You need to put it all together and
- add code for elements not yet covered
- Upload zip file of code and relate files to moodle by Monday 6 December 3pm
- Demo (~15min)
  - *Week beginning 6/12/21*
  - *Or, submit early and do demo in week before*
  - *Probably choice of Online/on campus – details tbc*
  - *sign up for a slot in moodle - email when this is set up*

Discussion of the Coursework Specification

See separate Word/pdf document for the full spec

Assessment overview

- The analysis of network traces (packet captures) is core to any cybersecurity operation.
- In this coursework, you will therefore **build a network trace (pcap) analyser utility using Python 3** (3.7 or higher).
- This will open a packet capture (pcap file) and parse it for specified information such as IP and email addresses.
- It will also perform some statistical analysis on the contents and visualise the traffic flows.

Assessment overview

- Your code should be PEP8 compliant and get a high pylint score
- Code must be documented appropriately.
- Each task should be coded using one or more functions
- Use separate scripts for different aspects, grouping together related functions.
- One "master script" which imports the others
- You will have coded some of the required functionality in lab exercises - and could import those modules rather than copying the code.
  - *Or you may want to change how functions are grouped, that's fine too*
- The requirements are described in detail in the CW spec.

Coursework brief

- Builds on lab exercises
  - *Which elements have you completed already?*
  - *Which elements are covered in remaining labs?*
- How to integrate?
- Test pcap file: **evidence-packet-analysis.pcap** (available in moodle)
  - *Run script with this during demo*
  - *Large file – use smaller pcap files while developing code (you could also extract maybe 100 packets from this into a new pcap for testing)*
  - *Test also on other pcap files*

Requirements / Marking criteria

1	Open and parse a pcap file. Store the contents of the file in one or more python objects for later reuse.
	Open, parse and close a pcap file. For the demo, this will be **evidence-packet-analysis.pcap**, but it should be easy to replace this with a different file. Store the parsed packets in Python, in one or more objects, for later re-use. Choose suitable data types for the object(s).

parse_pcap.py

Knowledge of data types, opening and closing files

Knowledge of decoding and different types of encoding

All criteria are 4 marks, except where stated otherwise

Requirements / Marking criteria

2	Output a summary of the packets in a table format, with one row for each traffic type. Each row should show the number of packets of that type, the first and last timestamps and the mean packet length for that type.
	Summarise the contents of the capture in a table format, with one row for each traffic type. • Each row should show the **number of packets of that type**, the **first** and **last timestamps** and the **mean packet length** for that type. • For the mean packet length, use the len() function rather than the stored len attribute. The example packet capture contains only three types (UDP, TCP, IGMP), but for full marks your script should handle the types **adaptively** and add other packet types if they occur. To test this, you should use other pcap files, see additional information below for some suggestions

Knowledge from lectures and labs for pcap and statistics

Knowledge for formatting output (could search for packages that help with this)

Requirements - Marking criteria ctd

3	Find the email addresses in (a) To: and (b) From: fields of any emails included in the packets. Further, find (c) the URIs of all requests for image (jpg, gif, PNG) files and (d) extract the filenames themselves. Print unique email addresses, full URIs and filenames to screen in a suitable format.
	Find the information listed below. For some of these, you need to use regex, others are fields in the request parts of packets or can be extracted using the os module. a) Find any email addresses present in **To**: fields from emails in the packets b) Find any email addresses present in **From**: fields from emails included in the packets c) Find the **URIs** of all requests for image (jpg, gif, PNG) files and d) extract the **filenames** of the requested image files. Print the <u>unique</u> email addresses, full URIs and filenames to screen in a suitable format. The output must distinguish email addresses according to whether they were To: or From:, and must show the full URIs separately from the filenames

email_analysis.py
pcap_downloads.py

Knowledge of regex, re library and os/os.path

Requirements - Marking criteria ctd

4	Extract the sender and destination IP address pairs for all packets and count how many packets were sent from/to each address pair. Print the results to screen sorted so that most frequent traffic comes first and nicely formatted.
	Extract the **sender and destination IP address pairs** for all packets and count how many packets were sent from/to each. • Return the results in the form of a **dictionary**. • Consider carefully what you should use as the dictionary key. • Print the results to screen sorted so that most frequent traffic comes first, and nicely formatted.

parse_pcap.py

Knowledge of dictionary data type
Formatting output

Requirements - Marking criteria ctd

5	Find the geolocation information for every unique destination IP address for which this is available, and create a KML file that can be opened e.g., in Google Earth.
	Find the **geolocation information** for every unique destination IP address for which this is available, and **create a KML** file that can be opened e.g. in Google Earth. • Include only the destination IP addresses. • For each IP address, you should include city, country, latitude and longitude. • You will need to deal with incomplete information – this can be part of your exception handling • Store the results in a KML file. Choose a suitable structure. Each point should include the packet count and city (if available) in its information.

Knowledge of geoip2.database and kml libraries

pcap_analysis.py

pcap_kml.py

Requirements - Marking criteria ctd

6	Plot the number of packets against time, and flag up any time intervals with exceptionally heavy traffic (e.g. more than the mean plus 2 standard deviations). Save the plot as a PNG file in the results directory created for #5 above. **[6 marks]**

Plot the **number of packets** against time as a **line chart**.

Part a is about the preparation, while part b is the actual graph. Each part is worth 3 marks.

a. To start, you need to group packets into equal length time intervals, and store the start time and the number of packets for each interval.

- Experiment with the length of time interval to get a good graph – it needs to be long enough for each interval to contain a good number of packets and short enough to give you a bit of a timeline.
- The length should be a variable that is easy to change.
- Calculate the threshold for exceptionally heavy traffic.
  A rule of thumb in statistics is to use the mean number of packets per interval plus two standard deviations.

b. Plot the data as a line chart.

- It is expected that you use matplotlib.pyplot, but you can choose different tools if you prefer.
- Make sure the plot is annotated well – using suitable x- and y-axis labels, caption etc.
- **Show the threshold** visually in your plot. This could be a horizontal line, or represented e.g. by using different colours.
- Display the plot on screen automatically while the program runs and save it as a PNG file.

General knowledge – loops, data types etc

Knowledge of statistics library

Knowledge of matplotlib.pyplot

Requirements - Marking criteria ctd

7	Use comprehensive exception handling throughout your code.
	Include **comprehensive exception handling** in your scripts. a. For example, you should check for non-existent files and allow for packet captures with different types of traffic to be parsed and processed successfully, ignoring unexpected packet types etc. b. Test your exception handling with several pcap files, such as the ones used in the lab and some of the ones mentioned in the additional information below.

Knowledge of exception handling and exception types

Requirements - Marking criteria ctd

8	Modular, future-proof code that uses variables, functions and return statements to facilitate reuse. Good documentation (doc strings, references, comments) and suitable structure. Some consideration for efficiency. Clear dependency diagram; PEP8 compliant code with a high pylint score. **[6 marks]**
	Consider the **overall design, style, readability and efficiency** of your code. a. Each element of the functionality should be achieved by one (or more) functions. The functions should be grouped appropriately into several separate modules. b. The **central script** which is run to call the code must be named **pcap_analyser.py**. c. Use variables rather than hardcoding, use return statements to assign results to variables for future use. d. Do not repeat the same operations or code unnecessarily. e. Document functions, modules etc with appropriate doc strings. f. Use additional comments in key places to explain complex code. g. Reference/cite any websites and other sources of code from others that you make use of. Give the specific source (e.g. the stackoverflow page, ideally also the author). h. Create a dependency diagram using the instructions in moodle and submit it with your code. i. Make your code **PEP8 and pylint compliant**. Refer to the PEP8 style guide (https://www.python.org/dev/peps/pep-0008/). Run your code through pycodestyle and pylint. Submit evidence of this in a file.

Dependency diagram exercise
PEP8 exercise

pycodestyle and pylint libraries
PEP8 style guide

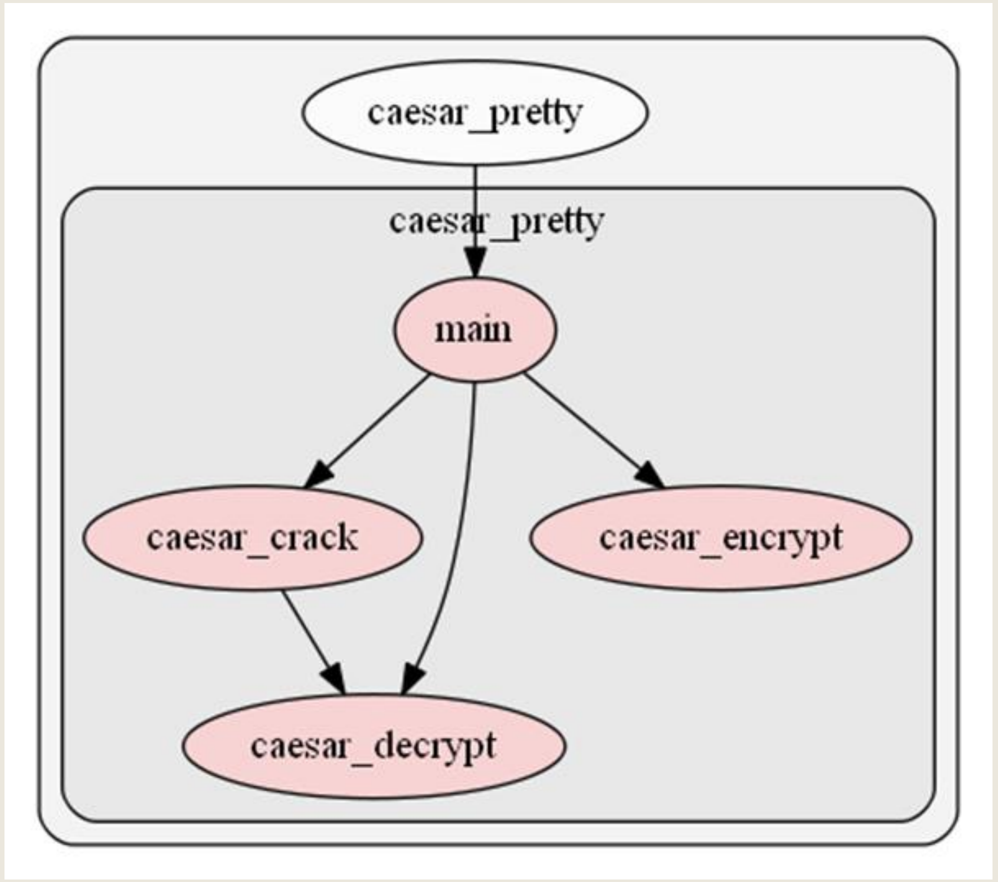
"Some consideration for efficiency"

For example,

- Think about the most suitable object type(s)
- Pass the variable(s) to the functions that need them
  - *Use return statements – handle printing separately*
- Store the packets in a suitable data structure for later re-use
  - *Requires changes to the functions in each of the pcap scripts from the labs, as they all read the file.*

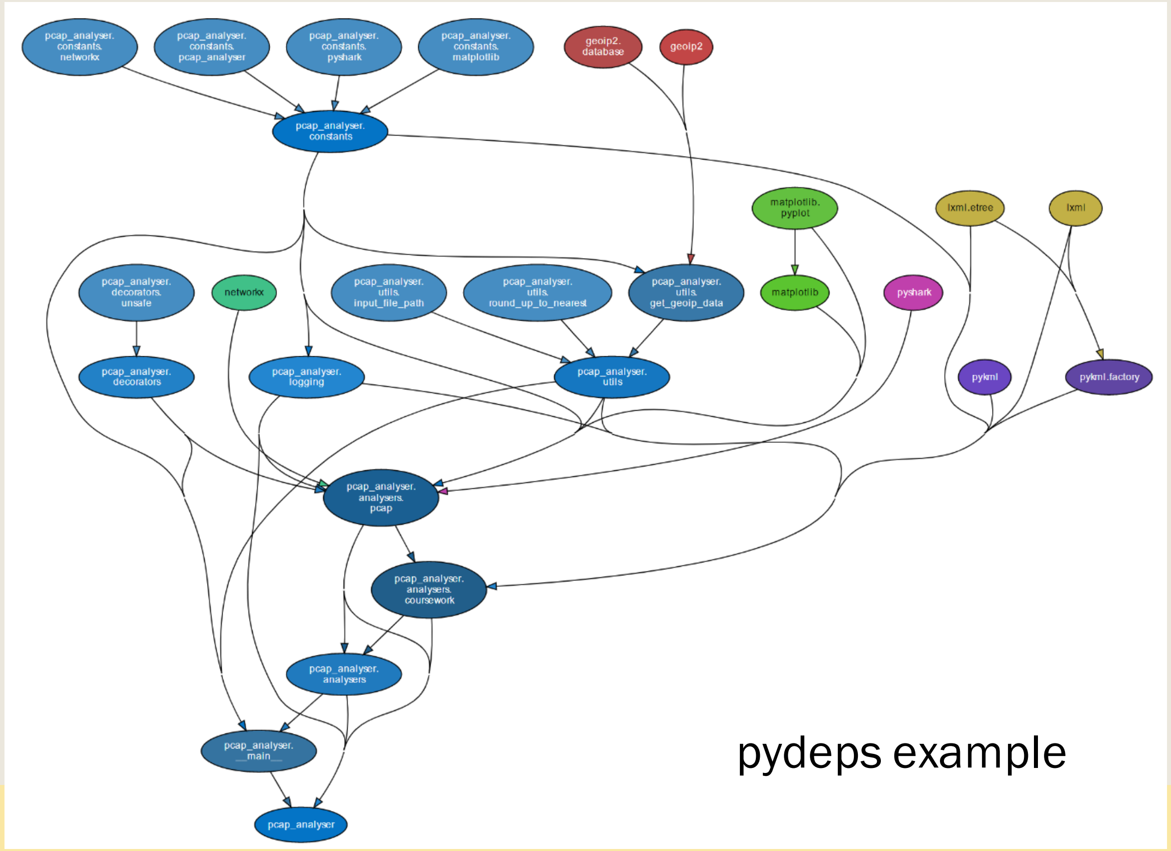
What is a dependency diagram?

- Maps all the modules with their functions, and which function calls which (dependency), which module imports which
- Useful for code testing and debugging
- Useful for getting to know code written by others (including coursework submissions!)
- More dependencies (higher complexity) may reduce code efficiency and make it harder to maintain
- More dependencies may also have a good point, showing that code is modular.

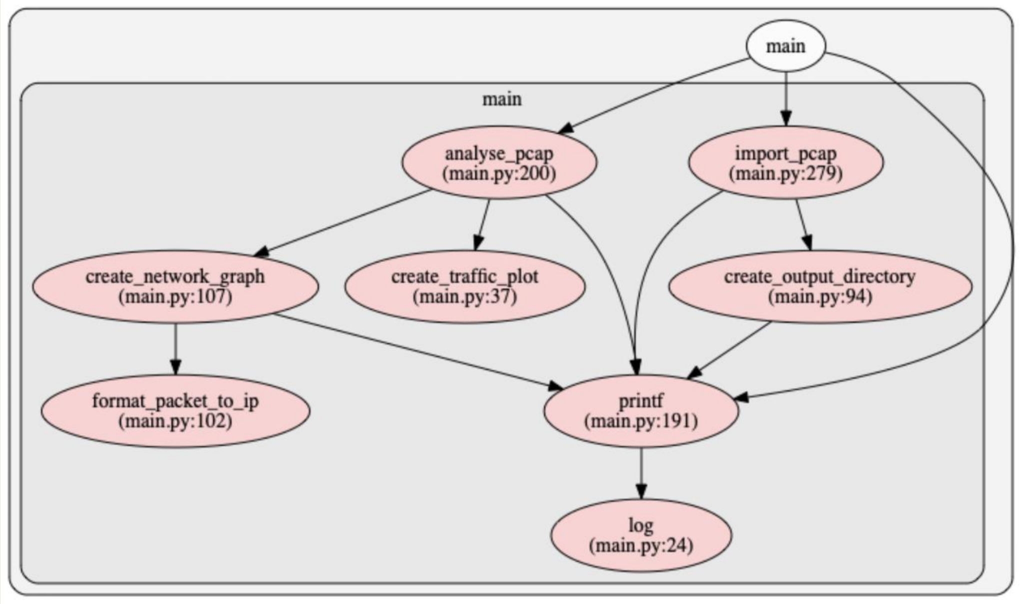


Creating a dependency diagram

- 2-step process; Various tools available
  - *pyan3 used in pdf with detailed instructions; an alternative is pydeps*
  - *graphviz tool to create diagram*
- Customise your diagram using various flags: Individual <u>functions</u> must be visible


pcap_analyser.constants.networkx
pcap_analyser.constants.pcap_analyser
pcap_analyser.constants.pyshark
pcap_analyser.constants.matplotlib
geoip2.database
geoip2
pcap_analyser.constants
matplotlib.pyplot
lxml.etree
lxml
pcap_analyser.decorators.unsafe
networkx
pcap_analyser.utils.input_file_path
pcap_analyser.utils.round_up_to_nearest
pcap_analyser.utils.get_geoip_data
matplotlib
pyshark
pcap_analyser.decorators
pcap_analyser.logging
pcap_analyser.utils
pyxml
pyxml.factory
pcap_analyser.analysers.pcap
pcap_analyser.analysers.coursework
pcap_analyser.analysers
pcap_analyser.__main__
pcap_analyser

pydeps example


main
main
analyse_pcap (main.py:200)
import_pcap (main.py:279)
create_network_graph (main.py:107)
create_traffic_plot (main.py:37)
create_output_directory (main.py:94)
format_packet_to_ip (main.py:102)
printf (main.py:191)
log (main.py:24)

pyan3 example showing filenames and line numbers for each function

The demo

9	Demo: running the test case, explanations of code and answers to related questions.
	During the **demo**, you will be asked to run your code while sharing your screen. Your code **must** use the pcap file **evidence-packet-analysis.pcap as the default** test case. You will be asked to explain selected aspects of your code. Make sure you know your code and its structure well.

Approx 15 minutes
Be prepared
Open up the tools you need
Share your screen (not the individual app)

What to submit

We expect your code to be in a series of .py files. You may additionally create a Jupyter Notebook if you wish – if you do, make sure you submit it

- All your code files
  - *But not the test pcap file*
  - *Make sure you include code for abandoned attempts: You can get marks even if code doesn't run, but only if it's submitted!*
- A readme file with info about "unusual" packages, if applicable
- Other files:
  - *KML file created for part 5*
  - *Png file created for part 6*
  - *Dependency diagram*
  - *Output from pycodestyle and pylint checking*

Getting Help

- Ask in the practical labs
- Last live lecture will be Q&A session for CW
- Use MS Teams
  - *Lab demonstrators can help you troubleshoot and give some feedback, but*
  - *Ask Petra, Sean or Mouad for clarification of the CW spec*

- To make use of the available help, you should start on the CW as soon as possible
- Lab 6 is the starting point

Using code from others

Internet, friends, etc

Can I use code from my friends?
Can I give my code to others?

- NO!!
- Exchanging code is Academic Deceit
- Penalties can be severe

- So does that mean we’re not allowed to talk to each other about the coursework?

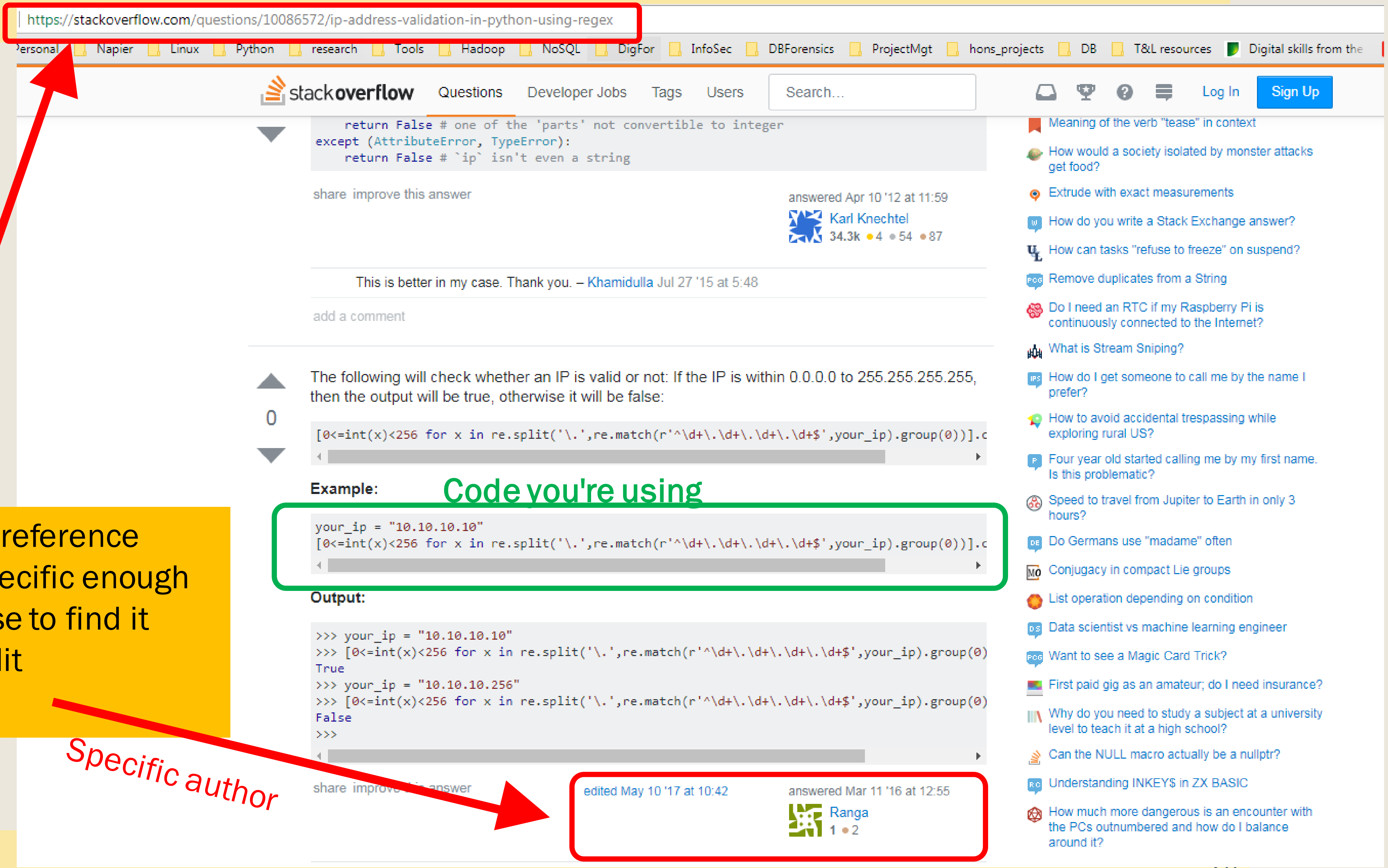
- You are allowed to talk …
- You are allowed to help each other, e.g. with fixing errors …

- … as long as you don’t give each other code!!

Can I use code from the internet?

- Yes, but:
  - *You need to clearly indicate where you have done this and give credit to your source*
  - *You need to make sure you fully understand it*
  - *Your code needs to run on Python3 in Windows - some websites will show examples for Python2 or linux*
  - *<u>Don't copy an entire solution for this problem</u> - I need to give YOU a mark, not the author of the code*

How to use code from the internet?



https://stackoverflow.com/questions/10086572/ip-address-validation-in-python-using-regex

```
    return False # one of the 'parts' not convertible to integer
except (AttributeError, TypeError):
    return False # `ip` isn't even a string
```

share improve this answer

answered Apr 10 '12 at 11:59
Karl Knechtel
34.3k ●4 ●54 ●87

This is better in my case. Thank you. – Khamidulla Jul 27 '15 at 5:48

add a comment

0

The following will check whether an IP is valid or not: If the IP is within 0.0.0.0 to 255.255.255.255, then the output will be true, otherwise it will be false:

```
[0<=int(x)<256 for x in re.split('\.',re.match(r'^\d+\.\d+\.\d+\.\d+$',your_ip).group(0))].c
```

Example:

```
your_ip = "10.10.10.10"
[0<=int(x)<256 for x in re.split('\.',re.match(r'^\d+\.\d+\.\d+\.\d+$',your_ip).group(0))].c
```

Output:

```
>>> your_ip = "10.10.10.10"
>>> [0<=int(x)<256 for x in re.split('\.',re.match(r'^\d+\.\d+\.\d+\.\d+$',your_ip).group(0)
True
>>> your_ip = "10.10.10.256"
>>> [0<=int(x)<256 for x in re.split('\.',re.match(r'^\d+\.\d+\.\d+\.\d+$',your_ip).group(0)
False
>>>
```

share improve this answer

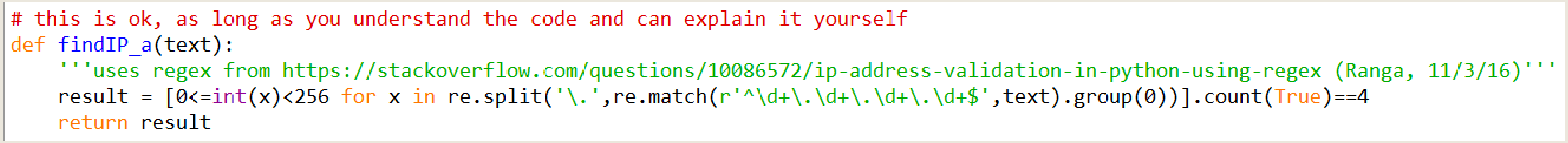
edited May 10 '17 at 10:42

answered Mar 11 '16 at 12:55
Ranga
1 ●2

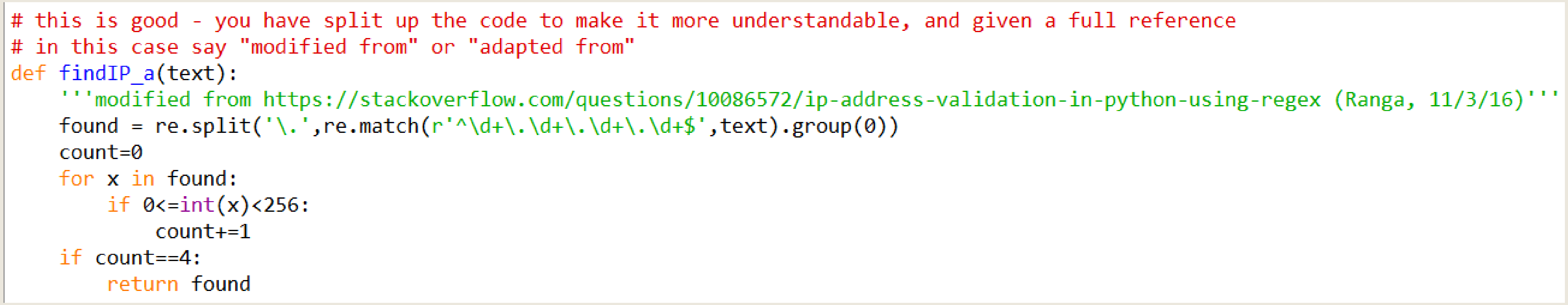
- Meaning of the verb "tease" in context
- How would a society isolated by monster attacks get food?
- Extrude with exact measurements
- How do you write a Stack Exchange answer?
- How can tasks "refuse to freeze" on suspend?
- Remove duplicates from a String
- Do I need an RTC if my Raspberry Pi is continuously connected to the Internet?
- What is Stream Sniping?
- How do I get someone to call me by the name I prefer?
- How to avoid accidental trespassing while exploring rural US?
- Four year old started calling me by my first name. Is this problematic?
- Speed to travel from Jupiter to Earth in only 3 hours?
- Do Germans use "madame" often
- Conjugacy in compact Lie groups
- List operation depending on condition
- Data scientist vs machine learning engineer
- Want to see a Magic Card Trick?
- First paid gig as an amateur; do I need insurance?
- Why do you need to study a subject at a university level to teach it at a high school?
- Can the NULL macro actually be a nullptr?
- Understanding INKEY$ in ZX BASIC
- How much more dangerous is an encounter with the PCs outnumbered and how do I balance around it?

You need to give a reference (citation) that is specific enough
- for someone else to find it
- And to give credit

How to use code from the internet?



```python
# this is ok, as long as you understand the code and can explain it yourself
def findIP_a(text):
    '''uses regex from https://stackoverflow.com/questions/10086572/ip-address-validation-in-python-using-regex (Ranga, 11/3/16)'''
    result = [0<=int(x)<256 for x in re.split('\.',re.match(r'^\d+\.\d+\.\d+\.\d+$',text).group(0))].count(True)==4
    return result
```



```python
# this is good - you have split up the code to make it more understandable, and given a full reference
# in this case say "modified from" or "adapted from"
def findIP_a(text):
    '''modified from https://stackoverflow.com/questions/10086572/ip-address-validation-in-python-using-regex (Ranga, 11/3/16)'''
    found = re.split('\.',re.match(r'^\d+\.\d+\.\d+\.\d+$',text).group(0))
    count=0
    for x in found:
        if 0<=int(x)<256:
            count+=1
    if count==4:
        return found
```

I have not tested this code! (don't be surprised if this example doesn’t actually work)

How NOT to use code from the internet?

```python
# this is not ok, as you are pretending you wrote this expression (you are stealing ideas)
def findIP_c(text):
    result = [0<=int(x)<256 for x in re.split('\.',re.match(r'^\d+\.\d+\.\d+\.\d+$',text).group(0))].count(True)==4
    return result
```

```python
# this is not ok, as the reference does not give enough detail for me to find it
def findIP_b(text):
    '''uses regex from stackoverflow'''
    result = [0<=int(x)<256 for x in re.split('\.',re.match(r'^\d+\.\d+\.\d+\.\d+$',text).group(0))].count(True)==4
    return result
```

I have not tested this code! (don't be surprised if this example doesn't actually work)

- Remember to use this approach for ALL exercises

- And in ALL your modules

- Applies also to references and ideas, not just code