Midterm Study Guide

# Structure

The midterm will be structured as follows:

* Part I: Coding Fundamentals
  + Multiple Choice / Short Answer Section
  + 30 Minutes, no Deepnote usage
  + 25 points evenly distributed
* Part II: Infosec Topics
  + Intel report
  + 30 Minutes
  + 25 points
* Part II: Writing Code
  + Coding Problems
  + 60 Minutes, done in Deepnote
  + 50 points with opportunities for bonus points

## Details

For the duration of the exam, you may access your own personal notes (including your work on this guide), the [Python Standard Library](https://docs.python.org/3/library/), and your textbook(s) at any time. You may not work together with others or access the Internet broadly during the exam at all. Access to the Python interpreter will be restricted to the second section only.

### **Section One**

Expect the midterm to consist of three sections; in the first section (30 minutes), there will be no interpreter allowed. This section will consist of multiple choice and short answer questions focused on definitions, **recognizing data types, interpreting operators, and other basic fundamentals** of Python. Themes you should expect and be prepared for include the following:

Variables

Operators

Core Data Types

Strings

Lists

Dictionaries

Tuples

Sets

For Loops

Conditional Statements

Functions and Methods

Expect questions in this section to look similar to the types of questions seen during previous quizzes and for them to be presented in the same general format (e.g. via Google forms).

Some Examples:

1. What data type is x?

x = [{'some\_key': 'some\_value'}]

1. What is the syntax to declare a dictionary in Python?

The focus here should be on rapid visual recognition and interpretation of Python code and basic definitions or key concepts. You will be expected here to move quickly and you will not have access to your interpreter. None of these problems will be very difficult or require extensive logic or critical thinking skills - but you will need to be able to move quickly to get through on time.

**Week 2: *Python Fundamentals***

\t, \n, \r - whitespace

; - allow multiple statements per line

\ - allows a statement to continue on another line

Objects - define what something is, what properties it has, and what can be done with it creating clear rules and definition for programmers.

* All objects have a **type** (below image) and all objects of the same type have common properties and functionalities

A picture containing graphical user interface

Description automatically generated

isinstance(1, int) – checks if the first item is of the specific type mentioned in the second item

**Integer**, int(x) – whole numbers that you can use in mathematical operations

**Floating Point**, float(x) – decimal-point precision real numbers that you can use in mathematical operations

Graphical user interface, text, application, chat or text message

Description automatically generatedTable

Description automatically generated

Reference HW 1 for more details on ints and floats

**Week 3: *Fundamentals***

Binary aka base 2 consist of 1s and 0s

* A byte is 8 bits

Table

Description automatically generated with medium confidence

Hexadecimal has 16 digits 0-9 and a-f

Table

Description automatically generated

Variables – store and retrieve data when programming

Boolean – True or False

* Python has implicit logic to compare objects as if they were Booleans deemed **truthy** or **falsy**.
  + **Falsy** are non-existent in some fashion (e.g. they are None or False), they contain nothing or contain a representation of nothing (e.g. zero as an integer or float), or they are empty (a string with no characters, a list with no objects, and so on). Most everything else is considered **truthy** just for existing.

Logical Operators

Table

Description automatically generated

Week 4: *Strings*

**Strings**, str(x) – sequence of text characters, cannot be used in mathematical operations

* No implicit conversion like int and floats
* + means concatenation
* \* means to double the string
* You can compare string because of their unicode values
  + ord(x) displays unicode value of a letter
  + chr(x) displays unicode value of a set of numbers

INDEXING: Indexing starts at zero (“zeroeth index”). The “first” item in an iterable is the item at the first index, not the zeroeth.

* [Start:Stop:Step]
  + “example”[0:6:2] → “example”[::2] → “eape”

Methods:

“string”.lower() – makes everything lower

“string”.upper() – makes everything upper

Functions:

len(“string”) – return the length of a function

“string”.replace(“st” , “”) – this will strip out a specified *substring* and replace it

Week 5: *List*

**Lists** (mutable) - an ordered set of objects. Held in [ ]

* indexing works the same way as str

Modifying Items in list:

* Slice through indexing:

combined = numbers + letters

print(combined)

combined[0]

print(combined)

[1,2,3,a,b,c]

[‘one’, 2,3,a,b,c]

* Sort vs sorted does not work on mixed type list
  + list.sort() will replace the list in place by modifying the original
  + sorted() will create a new list and return it
* Add items to a list
  + append adds OBJECTS *item.append(x)*
    - numbers = [1,2,3]

letters = [‘a’,’b’,’c’]

numbers.append(letters)

numbers

[1,2,3,[‘a’,’b’,’c’]]

* + extends expects an ITERABLE either *item.extend(x) or +*
    - numbers.extend(letters

numbers

[1,2,3,‘a’,’b’,’c’]

* Remove all items: list.clear()
* Remove last item and get its value passively changing parent: list.pop()
  + Combined = [1,2,3,’a’,’b’,’c’]

Combined.pop()

‘c’

* Remove an item by value: list.remove()
* Remove items by index or slice:
  + Combined = [1,2,3,’b’,’c’]

Del combined [-4:]

[1,2]

Text

Description automatically generatedIteration through For Loops

Methods

Graphical user interface, text, application

Description automatically generatedstr.join() – concatenates any number of **strings**  (emphasis on strings not list)

str.split() + string.splitline() converts text strings into structured data into a list

Graphical user interface, text, application

Description automatically generated

List Comprehensions – best when running a for loop in which you are creating an empty list to store some results based on another list. This will allow some action for each item in some iterable and return results as a new (unnamed list).

Graphical user interface, text, application, email

Description automatically generated

Zip() – produces a zip type object. Converting it to a list will create a list of tuples.

Graphical user interface, text, application, email

Description automatically generated

Week 6: *Dictionaries, Tuples, Sets*

**Dictionaries (dict)** – an unordered collection that allows for the storage of unique key : non-unique value pairs. Because they are unordered, items are referenced by their unique keys and not by the index position.

* {‘key’ : ‘value’ , ‘key2’ : ‘value’}

Text

Description automatically generated with medium confidence

Modifying Dictionaries

# use slice notation

x[‘key’] = [‘value’]

# use .update() with iterables

x.update([‘key2’, ‘value’])

# mix purpose: get the value of a specific key

# but if that key does not exist, insert the key and the specified value

x.setdefault(‘key3’, ‘default\_value’)

# use clear to delete dict contents

x.clear()

# del operator on slice

del x[‘key’]

# pop off the value at the specified key

# nb: returns the popped value

x.pop(‘key’)

# pop off the last key:value pair

# nb: returns the pair

x.popitem()

Reading Dictionaries

# gets a value for a given key

x.get(‘key’)

# this has an optional parameter that sets a default value to return if the key does not exist

x.get(‘key’, ‘default’)

# The next set of methods are useful if you need to *iterate* over dict data

# Get all key:value pairs

x.items()

# Get all the keys

x.keys()

# Get all the values

x.values()

# Get a new dictionary made from specific key:value pairs by naming selected keys

x.fromkeys()

**# DO NOT USE THIS SYNTAX**

# you will see this but do not do it.

# if you try to get a key that does not exist

# Python will error out.

x[‘key’]

**TUPLES**  is an immutable ordered object that can be of any length or empty, but commonly used to create dict or when scraping dictionary contents to a new structure because they can forcibly pair keys and values since they often come in pairs.

Text, letter

Description automatically generated

**SETS** – are collections, like lists or tuples, but sets can only contain one of each value.

* They are mutable and unordered (so you cannot slice them).

x = {1, 2, 3} # set, not a dictionary - no k, v pair

x = set(1, 2, 3) # set conversion from tuple

Sets are valuable for de-duplicating values (since they can only contain one of each value) and they have various methods for logic that are very helpful for solving problems.

Diagram

Description automatically generated with medium confidence

**Some of the most valuable set comparison methods:**

set.union()

set.difference()

set.intersection()

set.isdisjoint()

set.issubset()

Other set methods to know (these work like other similar methods for other data types we have covered previously):

set.clear()

set.pop()

set.update()

set.add()

**Other set methods to know** (these work like other similar methods for other data types we have covered previously):

set.clear()

set.pop()

set.update()

set.add()

Week 8: *Functions*

Timeline

Description automatically generated

**Methods** - methods are just functions that are specific to particular objects. They must be called with dot notation.

* Every week we’ve gone over a method related mainly to strings, this weeks are:
  + Str.swapcase() – calls code on a specified string to swap the case of the letter provided
  + Str.capitalize() – capitalizes a string object

\*remember in most cases you can chain these methods

### **Section Two**

The second section will feel a lot like the infosec skills section of the homework each week. In this section you will be provided with some set of security materials and you will be asked to write an intelligence report on the topic. You may be expected to interact with data in VirusTotal as part of this section so be prepared to log into your VT account.

Week 1: *Careers in Info Sec + Applying Python Skills*

Cybersecurity Industry roughly started in 1970s and since then has experienced rapid growth:

Text

Description automatically generated

Text

Description automatically generated

US has an annual talent shortfall with projections indicating shortfall will continue:

* JUN19 – MAY20: 171,000 jobs but only 125,000 workers
* (ISC)2 world’s largest nonprofit association of certified cybersecurity professionals indicates there is a gap of almost 3 million cybersecurity jobs
* 7% of industry is over 30 and aging to retirement

Careers

|  |  |
| --- | --- |
| Public | Private |
| Defensive (e.g. CISA)  Offensive (e.g. NSA)  Investigative (e.g. FBI)  Regulators (nascent)  In-house teams (every agency) | Organizations with an in-house cybersecurity team  Consulting firms  Product companies  Information vendors |

|  |  |
| --- | --- |
| Sectors: | Industry: *within each sector there are sector-specific roles focused on areas such as:* |
| * Security architecture * Vulnerability and patch management * Cyber threat intelligence * Security Operations Center (SOC) * Incident response * Security operations * Compliance * Network defense * Security risk assessment * Security audit * Cybersecurity project management * Penetration testing * Cyber threat hunting * Information security engineering | * Penetration tester * Cyber threat hunter * Security engineer * Security architect * Vulnerability and patch manager * Threat intelligence analyst * Security Operations Center (SOC) operator / analyst * Incident responder * Malware reverse engineer * Security auditor * Compliance manager |

Scripting helps to:

* Scale your work
* Automate routine functions
* Access data without having to wait for a tool to be built
* Find patterns in data you were unlikely to find using manual approaches

Interview Types

* **Technical Interview** - a technical scenario-based question that requires demonstrating clear comfort with technical topics, knowledge of the appropriate tools to use in different scenarios, and awareness of the other partner teams you might work with on different issues.
* **Scripting Interview** - involves live coding either on a whiteboard or on the computer in a tool like CoderPad. Generally looking for a narrower set of languages and lower level of skill then in a full coding interview.
* **Coding Interview** - involve live coding either on a whiteboard or on the computer in a tool like CoderPad. Looking for a high level of skill with multiple languages and ability to debug and try multiple paths to accomplish the same problem.

Week 2: *Counter-espionage Investigations: APT & IOCs*

Two common frameworks to understand the steps that APT actors use to conduct an attack:

the Lockheed Kill chain

Mitra Att&ck Matrix

Common Tactics:

* Reconnaissance
* Account compromise
* Malware delivery and exploitation

IOC - are pieces of forensic data, such as data found in system log entries or files, that identify potentially malicious activity on a system or network.

* Analysts use the term more broadly to mean any indicators associated with a particular activity set
  + EX: IP addresses, domains, malware hashes, email addresses, phone number

Table

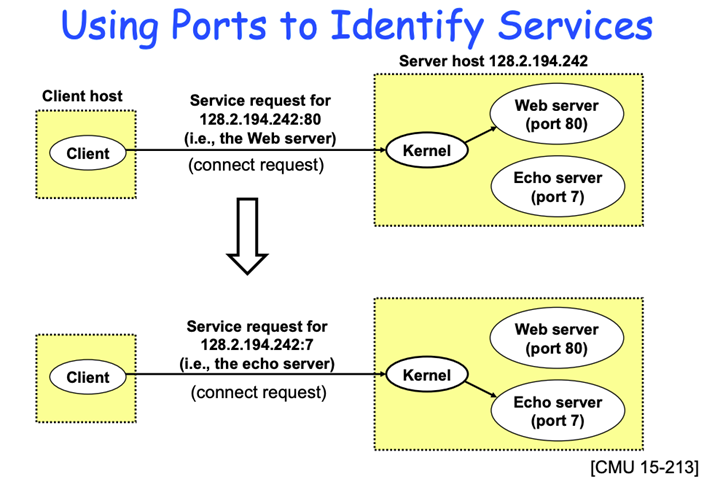
Description automatically generatedBreakdown IOCs using the TCP/IP model:

1. Physical:
   1. Device ID
2. Data Link:
   1. Number of routers a given web request went through before reaching destination i.e. hops
3. IP Addresses: *another unique identifier which identifies each device’s unique connection to the internet or to an intranet*.
   1. IPv4 has limited “IP Space” from 0.0.0.0 to 255.255.255.255
      1. Only 4B -5B allocatable addresses
      2. CIDR notation IPv4 with a / to give it more IP space
   2. IPv6 uses hexadecimal numbers 0-9 and a-f
      1. EX: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
      2. To shorten:
         1. Replace first two hextext of all 0s with a :: = 2001:0db8:85a3::8a2e:0370:7334
         2. All leading 0s drop = 2001:db8:85a3::8a2e:370:7334
   3. 127.0.0.1 is localhost that enables a device to access the network services specific to that device
4. Transport:
   1. Connections are generally facilitated through a client-server architecture in which the endpoints are called sockets that consist of software to handle communication between the client and server. Ports are used to identify and connect to the right program on the other machine

Diagram, timeline

Description automatically generatedTimeline

Description automatically generated



* 1. We care because certain ports are affiliated with certain common services. If you know the common ports, you can determine what services a server offers and guess at how the server is used. We often use this in investigations to determine what an attacker is doing.
     1. 0 – 1023 are most commonly used ports dedicated for internet use. Port 80 is used for HTTP traffic aka web traffic.
     2. We can open and close ports. Many businesses close FTP 20 21 on their devices so that no one can exfiltrate data

1. Application:
   1. Domain Name System (DNS) aka phonebook of the internet translates domain names to IP address.
      1. There can be many domains assigned to the same IP address

Week 3: *Crafting Comprehensive Intel Questions*

A picture containing chart

Description automatically generated

Week 4: *Intel Writing and a good BLUF*

Text

Description automatically generated with medium confidenceGraphical user interface, text, application, email

Description automatically generated

1. Put your main point up front aka bottom line up front (BLUF)
2. Write short paragraphs
   1. Sentence 1: Topic sentence (the controlling idea of the paragraph).
   2. Sentence 2: Explanation/elaboration of the topic sentence (if needed).
   3. Sentence 3: Fact/example/illustration #1 to support the topic sentence.
      1. Note: Facts should include technical artifacts inline or in an appendix.
   4. Sentence 4: Fact/example/illustration #2 to support the topic sentence.
   5. Sentence 5: Analysis (a sentence that answers the question “so what?”).
3. Use active voice
4. Use short, conventional words
5. Write short sentences
6. Include probability assessments
7. Be correct, credible, and complete
   1. Correct: Grammar and syntax
   2. Credible: State the assumptions on which your analysis rely
   3. Complete: State the unknowns or gaps in your analysis
8. Know your audience

Week 6: *Introduction to Malware, Working with Graphs, VirusTotal*

Malware – code designed to disrupt, disable, take control of, or co-opt a computing system

* Interact through links, files, zero-click exploitation
* Types
* A [**worm**](https://www.csoonline.com/article/3429569/what-is-a-computer-worm-how-this-self-spreading-malware-wreaks-havoc.html)is standalone software whichcan replicate itself and infect multiple computers on a network. Network worms often use computer networks to spread, slowing down traffic and relying on security failures (such as outdated operating systems and lack of AV) to spread. While viruses require the spreading of a file, worms are a standalone software and do not require a host or human help to spread.
* A [**virus**](https://www.csoonline.com/article/3406446/what-is-a-computer-virus-how-they-spread-and-5-signs-youve-been-infected.html)is typically attached to an executable file which means that it might exist on a system but will not spread until a user opens the infected program. Viruses often originate on the internet and spread when downloading a file infected with a virus. Peer to peer file sharing and email attachments are other common methods through which viruses are spread.
* A **Botnet** is a network of infected computers. There is a master computer which is infecting zombie computers. Then it can distribute its activity across all of the zombie computers. This is particularly important when you need a lot of traffic for your attack such as in a DDOS.
* A [**trojan**](https://www.csoonline.com/article/3403381/what-is-a-trojan-horse-how-this-tricky-malware-works.html)is a program that cannot reproduce itself but masquerades as something the user wants and tricks them into activating it so it can do its damage and spread.
* **Spyware** typically rests on a system and collects information about the user and relays that back to the operator of the spyware. Adware is a type of spyware which relays this info back to an advertiser.
* A [**rootkit**](https://www.csoonline.com/article/3222066/how-to-detect-and-remove-a-rootkit-in-windows-10.html)is "a program or, more often, a collection of software tools that gives a threat actor remote access to and control over a computer or other system." ([TechTarget](https://searchsecurity.techtarget.com/definition/rootkit)) It gets its name because it's a kit of tools that (generally illicitly) gain *root access* (administrator-level control, in Unix terms) over the target system, and use that power to hide their presence.
* ***Remote Access Trojans* or RATs**—essentially, rootkits that propagate like Trojans.
* [**Ransomware**](https://www.csoonline.com/article/3236183/ransomware/what-is-ransomware-how-it-works-and-how-to-remove-it.html)is a flavor of malware that encrypts your hard drive's files and demands a payment, usually in Bitcoin, in exchange for the decryption key.
* **Backdoors** are installed on a system by an attacker so that they can circumvent security measures to have persistent access to the system.
* Grayware – software or code that falls in between normal software and malware i.e stalkerware

Hashes – unique identifier of a file

Common hashes:

* The Message-Digest Algorithm 5 (MD5) - produces a 128-bit hash value. Although MD5 is historically very popular in malware research, it is vulnerable to collision attacks, meaning two different inputs may have the same hash.
* The Secure Hash Algorithm (SHA):
  + SHA-1 algorithm is also vulnerable to the collision attack.
  + SHA-2 is a set of cryptographic hash function including SHA-224, SHA-256, etc. SHA-256 generates a 256-bit hash and is more reliable then SHA1 or MD5s.

Malware Analysis

Dynamic vs. Static

* Static analysis is the process of analyzing malware without executing or running it. The objective is to extract as much metadata from the malware as possible.
* EX: you might extract unique strings or IP addresses from the malware.
* Dynamic analysis is the process of executing malware and analyzing its functionality and behavior. The objective is to understand exactly how and what the malware does during the execution.

Week 7: *OSINT +* *OPSEC*

* Suface/Clear Web/Clearnet – portion of the internet readily available to the public and searchable with standard web search engines
* Deep Web – content on internet not indexed and therefore not surface able by search engines
* Dark Web – content intentionally hidden, encrypted and requires specific browser to access it

OSINT techniques

* Reverse Image Searching (yandex, TinEye)
* Geolocation
* Accessing legal and government documents
* Social media analysis
* Search query manipulation (Google Dorking <https://www.sans.org/security-resources/GoogleCheatSheet.pdf>)

OPSEC

|  |  |
| --- | --- |
| Personal Security Practices | OPSEC for Security Investigators |
| 1. Think before you share    1. Understand what data is available on you online 2. Think before you record information about yourself on technology of any kind 3. Create compartments    1. Do NOT reuse passwords    2. Use two-factor authentication everywhere available 4. Have AV on your computer 5. Backup your data regularly 6. Use encrypted platforms 7. Ensure you have a custom password on your home wifi and change it every so often 8. Keep your devices patched! 9. Shutdown old and unused accounts | 1. Use virtual machines 2. Use VPNs 3. Consider whether or not using TOR is right for you 4. Do not send operational messages from your personal phone - use a burner phone 5. Operate in the local time you are emulating 6. Do not click on malicious links unless you are in a secure test environment 7. Do not download or open malware unless you are in a secure test environment 8. Make purchases on gift cards not credit cards 9. Encrypt your hard drive, files, and drives |

### Section Three

The third section will be presented in Deepnote and will consist of a series of coding problems. These problems will focus on combining different practical coding techniques together in concert. **You should expect every problem to request that you write a function, so you will need to understand well how to declare and call a function!**

Other core themes will include for loops, controlling code flow with conditional statements and operators, and manipulating strings and collection structures (lists and dictionaries, predominantly). We’ll be focusing on fundamentals and we won't try to trip you up with too much difficult logic; however, almost every problem will require multiple tools to solve.

Example:

1. Write a function that, given a list containing only integers, will print out the even integers from the list and return the odd integers from the list. If the list is empty of either type, print or return None.
2. Write a function that, given a string, will return a list of the same letters in reverse order.

There will be fewer problems in this section than in the first one.

**Hard Problems**

We will also provide a handful of fairly difficult problems from which you will choose TWO to answer.

* The five base points can be earned through describing how you might tackle the problem with text or a solid outline in pseudocode. These will be fairly substantial problems and we don’t necessarily expect a perfect answer, but you’ll need to at least sketch an effective approach that we can interpret and generally understand and follow to code a solution.
* Five bonus points per problem can be awarded for making a solid or complete attempt at solving the problem in code. The closer your solution is to a working solution, the more likely we are to award bonus points.

Expect these to be similar to (though simpler than) the Caesar Cipher homework. Remember, you do not need to code these for the base points. Just think through them logically and write out ideas.

What the midterm will NOT cover:

* List and dictionary comprehensions
* Anything using requests or web APIs

# Study Guide Assignment

**Study Guide Assignment:** To prepare for the midterm exam, you should write notes and solve problems that will help you understand the core concepts outlined for the midterm.

We will look for the following to grade your study guide:

* Did the student show understand of each of the core data types?
* Did the student demonstrate an understanding of the core concepts covered in this course?

At a minimum, we expect coverage of the following:

* All the core data types
* Use of functions and methods
* Use of for loops
* Use of conditional statements

If these characteristics are met, the study guide will be considered a success. You can turn in literally any format for this (a text file, a Deepnote notebook, a video of a study session with a classmate; you name it). If you do choose something somewhat unconventional, please let us know in advance - we may have some additional requests (such as timestamps for content in a video).

* A good potential direction to go for this exercise is to pair with other students and write challenge problems for one another. This is a great way to learn content and test your own understanding.
* Another might be to create an account and solve Edabit problems in different categories and link us to your solutions and comments.
* Photos of flash cards or other content is fine as well.

We’re very flexible here on how this study guide is built and submitted. We want you to focus on how you learn best and simply demonstrate to us how you’ve approached studying so that we can reward it in your grade.

We’re also providing a list of publicly accessible Edabit problems that we think are helpful. These do NOT need to be used for the guide - however, we think they are helpful examples of what to expect and you’re welcome to use them if you’re less creative.

**Edabit: Very Easy**

These problems are generally easier than what will be on the exam but can be good practice problems if you are struggling with how to declare and return data from functions.

[~~Power Calculator~~](https://edabit.com/challenge/v5gc8FQkDEepkqpfp)

[~~Remainder of Two Numbers~~](https://edabit.com/challenge/KWoj7kWiHRqJtG6S2)

[How Many Legs](https://edabit.com/challenge/QzXtDnSZL6y4ZcEvT)

[~~Less than 100~~](https://edabit.com/challenge/HRu9WggWxdSpYjxNf)

**Edabit: Easy**

Expect coding problems at this level for the exam.

[Factorial](https://edabit.com/challenge/FF6kYPHdAcJnoosr5)

[Scrabble](https://edabit.com/challenge/cH5ce3f4QgnreDW4v)

[Index Addition](https://edabit.com/challenge/gr4ihixfTaoEmZiin)

[Alphabet Soup](https://edabit.com/challenge/4Agr8CTY7x2rAhh5n)

**Edabit: Medium**

These are a little harder than where we will aim for the midterm problems. Solving these is good practice so you’re able to move quickly on easier problems.

[Profit](https://edabit.com/challenge/YfoKQWNeYETb9PYpw)

[Date Format](https://edabit.com/challenge/co4nwXSvnCjGEu8vp)

[Smile](https://edabit.com/challenge/8qD23E6XRMaWhyJ5z)

[Speeding](https://edabit.com/challenge/QgSMSMpfcEebAyCye)