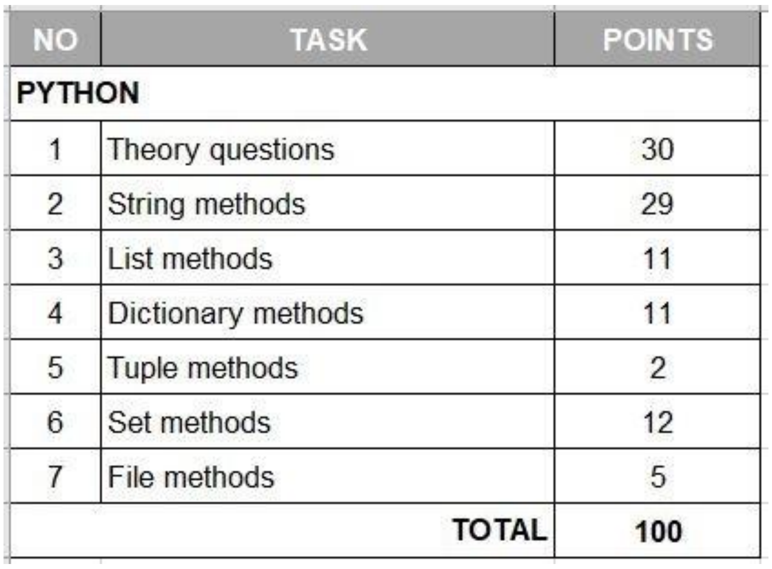
THEORYQUESTIONSASSIGNMENT

Python based theory

Tobecompletedatstudent’sownpaceandsubmittedbeforegivendeadline



NO	TASK	POINTS
PYTHON		
1	Theory questions	30
2	String methods	29
3	List methods	11
4	Dictionary methods	11
5	Tuple methods	2
6	Set methods	12
7	File methods	5
	TOTAL	100

1. Python theoryquestions	**30 points**

1. WhatisPythonandwhatareitsmainfeatures?
    - Python is high level language that is free and open source as well as easy to use and code with. One of the main features is that it is an object-oriented programming language. Moreover, it is a portable and interpreted language. It also is a dynamically typed language.
2. DiscussthedifferencebetweenPython2andPython3
    - The differences between Python 2 and Python 3 was obtained from the following websites: https://www.guru99.com/python-2-vs-python-3.html and https://www.interviewbit.com/blog/difference-between-python-2-and-3/.
    - Python 2 and Python 3 have different syntax, and Python 3 is therefore more user-friendly because of the easier syntax.
    - Python 3 stores strings as Unicode by default, where Python 2 does not do it by default and it has been to specified with u.
    - Python 2 uses xrange() whereas Python 3 uses range().
    - Python 2 exceptions are enclosed in notations, whereas in Python 3 exceptions are enclosed in parentheses.
    - In Python 2 the global variables if used in for-loop, but this is not the case with Python 3.
3. WhatisPEP8?

    - Information obtained from https://realpython.com/python-pep8/#:~:text=PEP%208%2C%20sometimes%20spelled%20PEP8,and%20consistency%20of%20Python%20code.

    - PEP8 provides guidelines and best practises on how to write Python code to ensure that it is easily readable.

4. In computing / computer science what is a program?
    - Information obtained from https://www.techtarget.com/searchsoftwarequality/definition/program#:~:text=In%20computing%2C%20a%20program%20is,area%20accessible%20to%20the%20computer..
    - A program is instructions (or ordered operations) that has been written in a programming language that the computer can follow and execute.
5. In computing / computer science what is a process?
    - Information obtained from: https://medium.com/@imdadahad/a-quick-introduction-to-processes-in-computer-science-271f01c780da.
    - A process is a program that is running/being executed on the computer.
6. In computing / computer science what is cache?
    - Information obtained from: https://www.bbc.co.uk/bitesize/guides/zmb9mp3/revision/3.
    - Cache is memory that is being used to temporarily hold information (set of instructions).
7. In computing / computer science what is a thread and what do we mean by multithreading?
    - Information obtained from: https://realpython.com/intro-to-python-threading/ and https://www.tutorialspoint.com/single-threaded-and-multi-threaded-processes.
    - A thread is a set of instructions to be executed by the computer. Multithreading is where multiple parts of a programme are executed at the same time.
8. In computing / computer science what is concurrency and parallelism and what are the differences?
    - Information obtained from: https://stackoverflow.com/questions/1050222/what-is-the-difference-between-concurrency-and-parallelism#:~:text=Concurrency%20is%20when%20two%20or,e.g.%2C%20on%20a%20multicore%20processor.
    - Concurrency is where two or more tasks start and finish during overlapping time periods whereas parallelism is where the tasks actually run at the same time e.g using a multicore processor.
9. What is GIL in Python and how does it work?
    - Information obtained from: https://realpython.com/python-gil/#:~:text=The%20Python%20Global%20Interpreter%20Lock%20or%20GIL%2C%20in%20simple%20words,at%20any%20point%20in%20time.
    - GIL stands for Global Interpreter Lock, which means that only one thread can control the Python interpreter and be executed at a time. Therefore, because there is only 1 lock, any code needs access to that lock to run, and can increase the speed of the program.

10. What do these software development principles mean: DRY, KISS, BDUF
    - Information obtained from: https://tech-en.netlify.app/articles/en546372/index.html
    - DRY means: Don't Repeat Yourself. This means reduce repetition in code.
    - KISS means: Keep It Simple, Stupid. This means keep the system as simple as possible.

- BDUF means: Big Design Up Front. Design the program before you implement it.

11. What is a Garbage Collector in Python and how does it work?
    - Information obtained from: https://towardsdatascience.com/memory-management-and-garbage-collection-in-python-c1cb51d1612c and https://www.tutorialspoint.com/How-does-garbage-collection-work-in-Python.
    - Garbage collector in Python is form of memory management and deletes object it no longer needs in order to free up some memory. When an object is assigned to a container in Python, it will have a reference count. If this reference count reaches 0, it is collected by Python and essentially removed.
12. How is memory managed in Python?
    - Information obtained from http://net-informations.com/python/basics/mem.htm:
    - The python objects and data structures form part of something called the Python private heap, which is managed by the interpreter. The Python memory manager will manage this private heap and identify objects that are not being used.
13. What is a Python module?
    - Information obtained from: https://docs.python.org/3/tutorial/modules.html
    - A python module contains definitions (from functions and modules) and statements. It can also contain executable code.
14. What is docstring in Python?
    - Information obtained from: https://docs.python.org/3/glossary.html.
    - A string literal that is used for code documentation and is not stripped like a comment is. It provides information about classes, functions and modules.
15. What is pickling and unpickling in Python? Example usage.
    - Information obtained from: https://www.tutorialspoint.com/python-pickling.
    - Pickling is used to serialise a python object where the object in Python is converted to a byte stream, and unpickling is the opposite where a byte stream is converted to a Python object. An example of its usage would be transferring data across servers/systems and store it.
16. What are the tools that help to find bugs or perform static analysis?
    - Information obtained from: https://www.tutorialspoint.com/what-are-the-tools-that-help-to-find-bugs-or-perform-static-analysis-in-python#:~:text=Pychecker%20and%20Pylint%20are%20the,and%20complexity%20of%20the%20bug.
    - Pylint and Pychecker are the two tools needed to help find bugs or perform static analysis.
17. How are arguments passed in Python by value or by reference? Give an example.

- Information obtained from: https://www.tutorialspoint.com/how-are-arguments-passed-by-value-or-by-reference-in-python. All arguments are passed by reference. This means that if you change the function parameter in the function it will also change it in the calling function.

Example:
```
people ={'Matthew':1,'Mark':2,'Luke':3,'John':4}
def followers(people):
  new_person = {'Mary': 5 }
  people.update(new_person)
  print("Inner",people)
  return
followers(people)
```

```python
print("Outer:",people)
```

- Output:

```
Inner {'Matthew': 1, 'Mark': 2, 'Luke': 3, 'John': 4, 'Mary': 5}
Outer: {'Matthew': 1, 'Mark': 2, 'Luke': 3, 'John': 4, 'Mary': 5}
```

18. What are Dictionary and List comprehensions in Python? Provide examples.

- Information obtained from: https://medium.com/analytics-vidhya/list-set-dictionary-comprehensions-in-python-8a0a7c06115e and https://www.tutorialsteacher.com/python/python-list-comprehension#:~:text=List%20comprehension%20in%20Python%20is,list%20using%20the%20for%20loop and https://www.datacamp.com/tutorial/python-dictionary-comprehension.
- Dictionary comprehensions transform one dictionary to another and makes it easier to access keys and values.
- List comprehensions are used to create lists from string/another list and has the syntax [expression for element in iterable if condition]. It can be broken down into 3 parts: the expression, for loop and if condition (although this if condition is optional.
- List and dictionary comprehensions make code easier to read, and are quicker.

19. What is namespace in Python?

- Information obtained from: https://www.javatpoint.com/namespace-in-python.
- Unique name for every object in Python. Often can be thought of like a dictionary, where the name and objects are represented as keys and values, respectively. There are four types: built-in, global, enclosing and local.

20. What is pass in Python?

- Information obtained from : https://www.w3schools.com/python/ref_keyword_pass.asp#:~:text=Python%20pass%20Statement&text=The%20pass%20statement%20is%20used,definitions%2C%20or%20in%20if%20statements.
- A pass can be thought of as a placeholder, this is where future code can be inserted. It prevents errors occurring.

21. What is unit test in Python?

- Information obtained from: https://machinelearningmastery.com/a-gentle-introduction-to-unit-testing-in-python/#:~:text=A%20unit%20test%20is%20a,and%20helps%20ensure%20code%20stability.
- A unit test will check individual components of code, most likely formatted in functions.

22. In Python what is slicing?

- Information obtained from: https://www.simplilearn.com/tutorials/python-tutorial/python-slicing#:~:text=Slice()%20Function%20in%20Python,of%20elements%20without%20changing%20it.
- Slice() is a function that will allow us extract sections of data and has the format slice(start,stop,step).

23. What is a negative index in Python?

- Information obtained from: https://www.i2tutorials.com/what-are-negative-indexes-and-why-are-they-used/#:~:text=Python%20programming%20language%20supports%20negative,from%20where%20the%20array%20ends.
- A negative index starts at the end of the index. Therefore -1 is the first element of the end.

24. How can the ternary operators be used in python? Give an example
    - Information obtained from https://book.pythontips.com/en/latest/ternary_operators.html.
    - Ternary operators are conditional expressions, and we can easily test a condition rather than using if-else statements, it makes code more concise.

```
is_tru = True
statement = "hello" if is_tru else "goodbye"
print(statement)
```

- The output for this code would be ‘hello’ as we have tested the condition and the condition is true.

25. What does this mean: *args, **kwargs? And why would we use it?
    - Information obtained from: https://www.programiz.com/python-programming/args-and-kwargs#:~:text=*args%20passes%20variable%20number%20of,a%20dictionary%20can%20be%20performed.
    - *args are non-keyword arguments. Asterisk is used to show that the argument length is variable. On the other hand, **kwargs are keyword arguments, and the double asterisk is used to show we have a variable length keyword argument. We would use these if we did not know the length of the arguments.
26. How are range and xrange different from one another?
    - Information obtained from: https://www.pythoncentral.io/how-to-use-pythons-xrange-and-range/#:~:text=For%20the%20most%20part%2C%20xrange,xrange%20returns%20an%20xrange%20object
    - Both create list of integers, but range is used in Python 2 and Python 3 and xrange is only used in Python 2. The range results in a python list and is static, but xrange results in xrange object and is used for very large ranges, and would prevent a memory error, but has to generate integers every time you access the index.
27. What is Flask and what can we use it for?
    - Information obtained from: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3
    - Flask is a python web framework that can be used to create python web applications.
28. What are clustered and non-clustered index in a relational database?
    - Information obtained from: https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-ver16 and https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/.
    - A clustered index defines the order the data is data stored in the table. On the other hand, a non-clustered index does not define the order of the data stored in the table.
29. What is a ‘deadlock’ a relational database?
    - Information obtained from: https://docs.oracle.com/javadb/10.6.2.1/devguide/cdevconcepts28436.html
    - A deadlock in a relational database is where two or more transactions are unable to finish, and are waiting for the other transaction to give up their lock.
30. What is a ‘livelock’ a relational database?
    - Information obtained from: https://blog.sqlauthority.com/2008/03/21/sql-server-introduction-to-live-lock-what-is-live-lock/

- Request denied for a lock repeatedly, due to many overlapping shared locks that cause interference with each other.

2. Python string methods: **describeeachmethodandprovideanexample** **The descriptions and examples were obtained from CFG notes and https://www.w3schools.com/python/python_ref_string.asp.**	**29 points**

METHOD	DESCRIPTION	EXAMPLE

capitalize()	First character of string converted to an uppercase character.	`str = 'hello'`
		`str_capitalize = str.capitalize()`
		`print(str_capitalize)`
		
		Output: Hello

casefold()	All characters in the string are converted to lowercase.	`str = 'Hello There'`
		`str_casefold = str.casefold()`
		`print(str_casefold)`
		
		Output: hello there

center()	String is center aligned, and a character can be specified as the fill character. The default is blank, however, in the example to the right, a hyphen has been used.	`str = 'Hello There'`
		`str_center = str.center(30, "-")`
		`print(str_center)`
		
		Output:
		---------Hello There----------

count()	Counts the number of times a value appears in a string.	`str = 'CFG teaches women and non-binaries to code. CFG are great.'`
		`str_count = str.count('CFG')`
		`print(str_count)`
		Output: 2

endswith()	Determines if the string ends with a specific value. Returns True or False. The example ends with 'great.' And therefore the output will be True.	`str = 'CFG teaches women and non-binaries to code. CFG are great.'`
		
		`str_end_with = str.endswith('great.')`
		`print(str_end_with)`
		
		Output: True

find()	This identifies the first instance of the value.	`str = 'CFG teaches women and non-binaries to code. CFG are great.'`
		`str_find= str.find('great')`
		`print(str_find)`
		Output: 52

format()	Formats a string and inserts the values into the string, and the values are separated by a comma.	`formatted_str = "Hello, my name is {} and I am {} years old".format("Gemma", 24)`
		`print(formatted_str)`

		Output: Hello, my name is Gemma and I am 24 years old

index()	Indicates position of the value.	`person = "Matthew"`
		`indx = person.index("a")`
		`print(indx)`
		
		Output: 1

isalnum()	Indicates whether all the characters in a string are alphanumeric (letters and numbers), will return a True or False. Returns True if all characters are alphanumeric, and False if they are not.	`person = "Matthew"`
		`alnum = person.isalnum()`
		`print(alnum)`
		
		Output: True

isalpha()	Indicates whether all the characters in a string are alphabetical (a-z), will return a True or False. Returns True if all characters are alphabetical, and False if they are not.	`person = "Matthew"` `alph = person.isalpha()` `print(alph)` Output: True
isdigit()	Indicates whether all the characters in a string are digits, will return True or False. Returns True if all characters in the string are digits, and False if they are not.	`people = ("1234")` `ppl_dig = people.isdigit()` `print(ppl_dig)` Output: True
islower()	Indicates whether the characters are lowercase, will return True or False. Returns True if all characters are lowercase, and False if they are not.	`lwer = ("hello")` `is_lwer = lwer.islower()` `print(is_lwer)` Output: True
isnumeric()	Indicates whether the characters are numeric, will return True or False. Returns True if all characters are numeric, and False if they are not.	`numer = ("hello")` `is_numer = numer.isnumeric()` `print(is_numer)` Output: False
isspace()	Indicates whether the characters are spaces, will return True or False. Returns True if all characters are spaces, and False if they are not.	`spaces = 'Gemma'` `space = spaces.isspace()` `print(space)` Output: False
istitle()	Indicates whether the words begin with an uppercase character, and the rest of the word is lowercase characters. Will return True or False. Returns True if word begins with uppercase character and ends in lowercase, but False if the words do not.	`string = 'Gemma'` `lwer = string.istitle()` `print(lwer)` Output: True
isupper()	Indicates whether the characters in the string are all uppercase, will return True of False value. Returns True if all characters are uppercase, and False if they are not.	`string = 'GEMMA'` `upp = string.isupper()` `print(upp)` Output: True
join()	All iterables (such as strings) will be joined as one string. Can specify character that acts as the separator.	`people= ('ABCDEF')` `people_join = ' '.join(people)` `print(people_join)` Output: A B C D E F
lower()	The characters in the string are converted to lowercase.	`people= ('GEMMA')` `people_lower = people.lower()` `print(people_lower)` Output: gemma
lstrip()	Removes characters on left hand side (the beginning), space is default, but can be other characters, as indicated in example.	`name = 'abcGemma'` `name_strip = name.lstrip('abc')` `print(name_strip)` Output: Gemma
replace()	Changes one value for another. The old value is first, and the new value is second.	`str = "Jasmine went to visit Gemma, but she did not bring his phone. What should Jasmine do?"` `str_rep = str.replace("his", "her")` `print(str_rep)`

		Output: Jasmine went to visit Gemma, but she did not bring her phone. What should Jasmine do?

rsplit()	Splits a string on the separator and converts it to a list.	`str = "Matthew, Mark, Luke, John"`
		`str_spl = str.split(", ") print(str_spl)`
		
		Output: ['Matthew', 'Mark', 'Luke', 'John']

rstrip()	Removes characters on right hand side (the end), space is default, but can be other characters, as indicated in example.	`name = 'GemmaABC'`
		`name_strip = name.rstrip('ABC')`
		`print(name_strip)`
		
		Output: Gemma

split()	Splits a string on the separator, and the output is a list.	`name = 'Gemma,John,Jack,Roger'`
		`name_spl = name.split(",")`
		`print(name_spl)`
		
		Output: ['Gemma', 'John', 'Jack', 'Roger']

splitlines()	Splits a string on the line breaks, and the output is a list.	`names = 'Gemma Baldock \n Roger`
		`Baldock \n Sandra Baldock'`
		`names_spl = names.splitlines()`
		Output:
		['Gemma Baldock ', ' Roger Baldock ', ' Sandra Baldock']

startswith()	Check whether a string starts with a specific value.	`names = 'Gemma Baldock \n Roger`
		`Baldock \n Sandra Baldock'`
		`names_sw = names.startswith('G')`
		`print(names_sw)`
		Output: True

strip()	Strips the spaces off the beginning and end of the string.	`name = '            Gemma Baldock`
		`'`
		
		`name_str = name.strip()`
		`print(name_str)`
		
		Output: Gemma Baldock

swapcase()	Makes the lowercase characters uppercase, and the uppercase characters lowercase.	`name = 'mY NAME IS gEMMA`
		`bALDOCK'`
		`name_rev = name.swapcase()`
		`print(name_rev)`
		Output: My name is Gemma Baldock

title()	In the string, each word starts with an uppercase character.	`name = 'my name is gemma'`
		`name_titl = name.title()`
		`print(name_titl)`
		Output:
		My Name Is Gemma

upper()	Converts the characters in the string to uppercase (all of them).	`name = 'my name is gemma'`
		`name_upp = name.upper()`
		`print(name_upp)`
		
		Output: MY NAME IS GEMMA

3. Python listmethods: **describeeach method and provide anexample** **The descriptions and examples were obtained from CFG notes and https://www.w3schools.com/python/python_ref_list.asp.**	**11 points**

Method	Description	Example

append()	Adds element to end of list.	`name = ["Matthew", "Mark", "Luke"]`
		`name.append("John")`
		`print(name)`
		Output: ['Matthew', 'Mark', 'Luke', 'John']

clear()	Clears elements in list.	`name = ["Matthew", "Mark", "Luke"]`
		`name.clear()`
		`print(name)`
		Output: []

copy()	Copies list.	`name = ["Matthew", "Mark", "Luke"]`
		`name_copy = name.copy()`
		`print(name_copy)`
		Output: ['Matthew', 'Mark', 'Luke']

count()	Counts the number of occurrences of the elements in the list.	`name = ["Matthew", "Mark", "Luke"]`
		`name_count = name.count("Luke")`
		`print(name_count)`
		Output: 1

extend()	Elements of one list added on to the end of another.	`name = ["Matthew", "Mark", "Luke"]`
		`name_two = ["John", "Mary", "Joseph"]`
		`name.extend(name_two)`
		`print(name)`

		Output: ['Matthew', 'Mark', 'Luke', 'John', 'Mary', 'Joseph']

index()	Prints the value of the index.	`name = ["Matthew", "Mark", "Luke"]`
		`inx = name.index("Mark")`
		`print(inx)`
		
		Output: 1

insert()	Inserts value at specified position. Position you want to insert comes first, then what value you want to insert comes next, and is comma separated.	`name = ["Matthew", "Mark", "Luke"]`
		`name.insert(3, "John")`
		`print(name)`
		
		Output: ['Matthew', 'Mark', 'Luke', 'John']

pop()	Removes value at specified position.	`name = ["Matthew", "Mark", "Luke"]`
		`name.pop(2)`
		`print(name)`
		
		Output: ['Matthew', 'Mark']

remove()	Removes the first occurrence of a specified value.	`name = ["Matthew", "Mark", "Luke", "Mark"]`
		`name.remove("Mark")`
		`print(name)`
		
		Output: ['Matthew', 'Luke', 'Mark']

reverse()	Reverses order of the list.	`name = ["Matthew", "Mark", "Luke", "John"]`
		`name.reverse()`
		`print(name)`
		Output: ['John', 'Luke', 'Mark', 'Matthew']

sort()	Sorts the elements of the list alphabetically.	`name = ["Matthew", "Mark", "Luke", "John"]`
		`name.sort()`
		`print(name)`
		
		Output: ['John', 'Luke', 'Mark', 'Matthew']

4. Python tuple methods: **describe each method and provide an example** **The descriptions and examples were obtained from CFG notes and https://www.w3schools.com/python/python_ref_tuple.asp.**	**2 points**

Method	Description	Example

count()	Counts occurrences of a specific value in the tuple.	name = ("Matthew", "Mark",
		"Luke", "Mark")
		name_count =
		name.count("Mark")
		print(name_count)
		Output: 2

index()	Gives position of value where it first occurs.	name = ("Matthew", "Mark",
		"Luke", "Mark")
		name_inx =
		name.index("Mark")
		print(name_inx)
		
		Output: 1

5. Python dictionary methods: **describe each method and provide an example** Description and examples obtained from CFG notes and https://www.w3schools.com/python/python_ref_dictionary.asp	**11 points**

Method	Description	Example

clear()	Elements of dictionary cleared.	dog = {
		"breed": "Labrador",
		  "age": 14,
		  "colour": "white"}
		
		dog.clear()
		print(dog)
		
		Output: {}

copy()	Copies dictionary elements.	dog = {
		"breed": "Labrador",
		  "age": 14,
		  "colour": "white"}
		
		copy_dic = dog.copy()

		print(copy_dic)
		
		Output: {'breed': 'Labrador', 'age': 14, 'colour': 'white'}

fromkeys()	Creates dictionary from an iterable (keys), with default value as None, but in the example the default value specified as 12.	dogs = ('Jaxon', 'Billy', 'Roger')
		
		ages = 12
		
		dog_ages = dict.fromkeys(dogs, ages)
		print(dog_ages)
		
		Output: {'Jaxon': 12, 'Billy': 12, 'Roger': 12}

get()	Gets item value from key.	dog = {
		"breed": "Labrador",
		  "age": 14,
		  "colour": "white"}
		
		dog_get = dog.get("age")
		print(dog_get)
		
		Output: 14

items()	Lists the key, value pair	dog = {
		"breed": "Labrador",
		  "age": 14,
		  "colour": "white"}
		
		dogs = dog.items()
		print(dogs)
		Output: dict_items([('breed', 'Labrador'), ('age', 14), ('colour', 'white')])

keys()	Lists dictionary keys.	people = {"Man 1":"Matthew", "Man 2":"Mark"}
		kys = people.keys()
		print(kys)
		
		Output: dict_keys(['Man 1', 'Man 2'])

pop()	Removes the item from dictionary, using specific key.	people = {"Man 1":"Matthew", "Man 2":"Mark"}
		people.pop("Man 1")
		print(people)
		
		Output: {'Man 2': 'Mark'}

popitem()	Removes last item in dictionary.	people = {"Man 1":"Matthew", "Man 2":"Mark", "Man 3":"John"}

		`people.popitem()`
		`print(people)`

setdefault()	Returns value of specific key, if key does not exist it inserts key and creates default value.	people = {"Man 1":"Matthew", "Man 2":"Mark", "Man 3":"John"}
		people.setdefault("Man 4", "Luke")
		print(people)
		Output:
		{'Man 1': 'Matthew', 'Man 2': 'Mark', 'Man 3': 'John', 'Man 4': 'Luke'}

update()	Inserts item (with key value pairs).	people = {"Man 1":"Matthew", "Man 2":"Mark", "Man 3":"John"}
		people.update({"Man 4": "Luke"})
		print(people)
		
		Output:
		{'Man 1': 'Matthew', 'Man 2': 'Mark', 'Man 3': 'John', 'Man 4': 'Luke'}

values()	Returns all values.	people = {"Man 1":"Matthew", "Man 2":"Mark", "Man 3":"John"}
		ppl_values = people.values()
		print(ppl_values)
		Output:
		dict_values(['Matthew', 'Mark', 'John'])

6. Python setmethods: describeeachmethodandprovideanexample Description and examples obtained from CFG notes and https://www.w3schools.com/python/python_ref_set.asp.	**12 points**

Method	Description	Example

add()	Element added to set.	people = {"Matthew", "Mark", "Luke"}
		people.add("John")
		print(people)
		Output:
		{'Matthew', 'Mark', 'John', 'Luke'}

clear()	Clears set. All elements removed.	people = {"Matthew", "Mark", "Luke"}
		
		people.clear()
		
		print(people)

		Output: set()

copy()	Copies set.	`people = {"Matthew", "Mark", "Luke"}`
		`copy_people = people.copy()`
		`print(copy_people)`
		
		Output:
		{'Mark', 'Luke', 'Matthew'}

difference()	Differences between the sets. In the example, it lists only the items that appear in the set people and not in the set people_two.	`people = {"Matthew", "Mark", "Luke"}`
		`people_two = {"Mark", "Luke", "John"}`
		`people_diff = people.difference(people_two)`
		`print(people_diff)`
		Output:
		{'Matthew'}

intersection()	Commonalities between the sets. In the example, it lists only the items appear both in the sets people and people_two.	`people = {"Matthew", "Mark", "Luke"}`
		`people_two = {"Mark", "Luke", "John"}`
		`people_inter = people.intersection(people_two)`
		`print(people_inter)`
		
		Output: {'Mark', 'Luke'}

issubset()	Checks to see if items in one set also belongs to other set and will return a True or False value.
	
	In the example, true means that the items in set people are also in set people_two. However, the output is actually False as the items in set people are not in set people_two.
		`people = {"Matthew", "Mark", "Luke"}`
		`people_two = {"Mark", "Luke", "John"}`
		`people_inter = people.issubset(people_two)`
		`print(people_inter)`
		Output:
		False

issuperset()	Determines whether the specified set is in the original set and will return True or False. True means that the items in the specified set are in the original set, and False means that they are not in original set.	`people = {"Mark"}`
		`people_two = {"Mark", "Luke", "John"}`
		`people_inter = people_two.issubset(people)`
		`print(people_inter)`
		Output: False

pop()	Item removed from set (in a random manner). So the output could be different in the example if the code was run again.	`people = {"Mark", "Matthew", "John"}`
		`people.pop()`
		`print(people)`
		
		Output:
		{'John', 'Mark'}

remove()	Removes element in the set.	`people = {"Mark", "Matthew", "John"}`
		`people.remove("Mark")`

		`print(people)`
		Output:
		{'Matthew', 'John'}

symmetric_difference()	Lists items in both sets excluding the items common to each set.	`people = {"Mark", "Matthew", "John"}`
		`people_two = {"Mark", "Matthew", "John", "Luke"}`
		`people_sym_diff = people.symmetric_difference(people_two)`
		`print(people_sym_diff)`
		Output: {'Luke'}

union()	Returns items in both sets, excluding duplicates.	`people = {"Mark", "Matthew", "John"}`
		`people_two = {"Mark", "Matthew", "John", "Luke"}`
		`people_union = people.union(people_two)`
		`print(people_union)`
		
		Output:
		{'Matthew', 'John', 'Mark', 'Luke'}

update()	Inserts the second set into the first set. So in the example, the people_two is inserted into the people set.	`people = {"Mark", "Matthew"}`
		`people_two = {"John", "Luke"}`
		`people.update(people_two)`
		`print(people)`
		Output:
		{'Mark', 'Matthew', 'Luke', 'John'}

7. Python filemethods: describeeachmethodandprovideanexample Description and examples obtained from CFG notes and https://www.w3schools.com/python/python_ref_file.asp.	**5 points**

Method	Description	Example

read()	Reads file contents.	`file = open("file.txt", "r")`
		`print(file.read())`

readline()	Reads first file line.	`file = open("file.txt", "r")`
		`print(file.readline())`

readlines()	Lines in file returned as list.	`file = open("file.txt", "r")`
		`print(file.readlines())`

write()	Adds text to file.	`newfile = open("newfile.txt", "a")`
		`newfile.write("This is my CFG theory assessment")`
		`newfile.close()`

writelines()	Adds texts to file as a list.	`newfile2 = open("newfile2.txt", "a")`
		`newfile2.writelines("This is my CFG theory assessment", "I hope I pass")`
		`newfile2.close()`