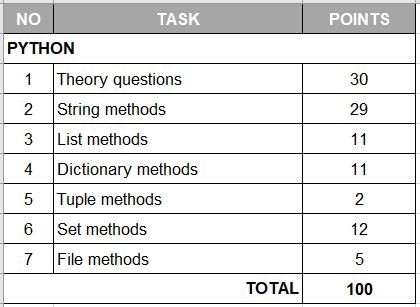
THEORY QUESTIONS ASSIGNMENT – Aurora Sanchez Diaz

Python based theory

To be completed at student’s own pace and submitted before given deadline



**30 points**

**1. Python theory questions**

1. What is Python and what are its main features?

Python is a programming language created by Guido van Rossum in the late 1980s. Main features:

* Object oriented programming (OOP)
* Dynamically typed.
* It does not need to compile code
* Open source. It is freely available to the public
* Platform independent. The code you write in Python will work in Mac, Windows or Linux
* Relatively simple syntax, which makes it easy to write and to read (it uses indentation instead of curly braces {} )

1. Discuss the difference between Python 2 and Python 3

Python Python 3 was released in 2008 to fix problems on Python 2. Python 2 was discontinued on 2022. There are some key differences:

* Python 3’s syntax is easier to understand than Python 2’s.
* In Python 3, when 2 integers are divided, you get a float value. On Python 2 you always got an integer value.
* Python 3 makes ordering comparisons easier than in Python 2.
* In Python 3 the range() function was introduced to perform iterations. In Python 2, the xrange() is used for iterations.

Adapted from the following source(s): [guru99.com/python-2-vs-python-3.html](file:///C:\Users\auror\Downloads\guru99.com\python-2-vs-python-3.html)

1. What is PEP 8?

It is a style guide that provides coding conventions for Python. It prioritises readability. Some of its main guidelines are:

* Using 4 spaces per indentation level. Spaces are preferred instead of tabs.
* Maximum line length should be limited to 79 characters
* Being consistent with breaks next to binary operators. One can break before or after a binary operator, but the same convention should be followed across the file.
* Avoid trailing whitespace anywhere.

Adapted from the following source(s): [peps.python.org/pep-0008/](file:///C:\Users\auror\Downloads\peps.python.org\pep-0008\)

1. In computing / computer science what is a program?

Programs are sequences of instructions for a computer to perform. They are created to implement algorithms

Adapted from the following source(s): <https://www.bbc.co.uk/bitesize/guides/zts8d2p/revision/1>

1. In computing / computer science what is a process?

A process is a program that is running on your computer. It can include smaller tasks or entire applications. All processes are composed of one or more threads.

Computers usually have more processes running than programs because programs are formed of processes. In Windows, processes can be accessed using the Task Manager (Ctrl-Alt-Delete).

Adapted from the following source(s): <https://techterms.com/definition/process>

1. In computing / computer science what is cache?

Cache saves recently used information so that it can be accessed quickly later. Computers use various types of caching to help them run more efficiently and thus improve performance. There are different types of cache, such as browser cache, disk cache, memory cache, and processor cache.

Adapted from the following source(s): <https://techterms.com/definition/cache>

1. In computing / computer science what is a thread and what do we mean by multithreading?

Thanks to threads, a program can execute actions in a sequence or many actions at the same time. Each thread in a program identifies a process that runs when the program asks it to.

Threads are given different priority levels, so that some threads are more important than others.

Once the CPU is finished processing one thread, it can run the next thread waiting in line.

Therefore, multithreading occurs when a program can run several threads at once. The majority of operating systems support multi-threading.

Adapted from the following source(s): <https://techterms.com/definition/thread>

1. In computing / computer science what is concurrency and parallelism and what are the differences?

**Concurrency:**

A programme handles multiple tasks at once is concurrent. By using a single processing unit, concurrency is a method for reducing the system's reaction time. While concurrency gives the appearance of parallel processing, multiple tasks are really being executed simultaneously inside the application rather than in discrete chunks. One task doesn't finish completely before the next one starts.Diagram

Description automatically generated with low confidence

**Parallelism:**   
Parallelism is related to an application where  tasks are divided into smaller sub-tasks that are processed seemingly simultaneously or. It is used to increase the throughput and computational speed of the system by using multiple processors. It enables single sequential CPUs to do lot of things “seemingly” simultaneously.

Diagram

Description automatically generated

Adapted from the following source(s): [https://www.geeksforgeeks.org/difference-between-concurrency-and-parallelism/#:~:text=Concurrency%20is%20the%20task%20of,of%20running%20multiple%20computations%20simultaneously.](https://www.geeksforgeeks.org/difference-between-concurrency-and-parallelism/%23:~:text=Concurrency%20is%20the%20task%20of,of%20running%20multiple%20computations%20simultaneously.)

1. What is GIL in Python and how does it work?

When working with processes, Python employs a form of process lock called the Global Interpreter Lock (GIL). Python typically executes a collection of typed statements using just one thread. This indicates that only one thread at a time will be performed in Python. Because of GIL, the performance of a single-threaded process and a multi-threaded process will be the same. Python's global interpreter lock, which limits the number of threads and acts as a single thread, prevents us from achieving multithreading.

Adapted from the following source(s): https://www.geeksforgeeks.org/what-is-the-python-global-interpreter-lock-gil/

1. What do these software development principles mean: DRY, KISS, BDUF

DRY, which stands for 'don't repeat yourself’. It tries to avoid code repetition/redundancy.

Adapted from the following source(s): <https://www.digitalocean.com/community/tutorials/what-is-dry-development>

Keep it simple, stupid (**KISS)** is a design principle which states that designs and/or systems should be as simple as possible. Therefore, complexity should be avoided

Adapted from the following source(s): <https://www.interaction-design.org/literature/topics/keep-it-simple-stupid>

**What is BDUF?**

Big Design Up Front is an approach where a website, app, or software design is completed and perfected up-front, before its implementation is started. It is associated with the term waterfall process. BDUF was the way to do things before Agile.

Adapted from the following source(s): https://www.freecodecamp.org/news/the-pros-and-cons-of-big-design-up-front-and-what-i-do-instead-375f00542dec/

1. What is a Garbage Collector in Python and how does it work?

It ensures that the RAM does not fill up. The Garbage Collector (GC) has 3 generations.

There is a threshold for the number of objects that can be collected in each generation. If one generation is “full”, its objects move to an older generation.

The GC and its generations can be accessed typing “import gc” and “gc.get\_threshold”.

Adapted from the following source(s):

https://stackify.com/python-garbage-collection/

1. How is memory managed in Python?

Thanks to the Garbage Collector, the user does not need to do manual garbage collection. Python takes care of deleting data that is no longer needed.

However, on a more theoretical aspect,

1. What is a Python module?

A Python module is a .py file. It can be imported and referenced in other files.

1. What is docstring in Python?

It is a comment inserted by the programmer to document code. Tiple quotes should be used at the beginning and end of a docstring.

1. What is pickling and unpickling in Python? Example usage.

Pickle is a module that converts Python objects into byte streams (0 and 1s). Unpickling does the opposite (byte streams are converted into Python objects). The advantage of pickling/unpickling is that using intermediate data formats is not necessary.

However, there arguments against the use of Pickle (concerns about security when unpickling data from unknown sources, for example. Also, JSON is faster than Pickle)

These are two examples showing how to pickle and unpickle, respectively:

**Pickle a simple list**: Pickle\_list1.py

import pickle

mylist = ['a', 'b', 'c', 'd']

with open('datafile.txt', 'wb') as fh:

   pickle.dump(mylist, fh)

**Unpickle a simple list**: unpickle\_list1.py

import pickle

pickle\_off = open ("datafile.txt", "rb")

emp = pickle.load(pickle\_off)

print(emp)

Source: https://www.tutorialspoint.com/python-pickling

1. What are the tools that help to find bugs or perform static analysis?

Pychecker and Pylint respectively. Both are opensource and they can be installed by using the pip package on Python.

Adapted from the following source(s): <https://www.tutorialspoint.com/what-are-the-tools-that-help-to-find-bugs-or-perform-static-analysis-in-python>

1. How are arguments passed in Python by value or by reference? Give an example.

By reference. Passing an argument by reference means that the argument passed given to a function refers to a variable that already exists. Example:

fruit\_amount = {'Bananas': 4, 'Pears': 4, 'Kiwis': 3}

def test(fruit\_amount):

new = {'Apples':7}

fruit\_amount.update(new)

print("Inside the function", fruit\_amount)

return

test(fruit\_amount)

print("Outside the function:", fruit\_amount)

Result:

Inside the function {'Bananas': 4, 'Pears': 4, 'Kiwis': 3, 'Apples': 7}

Outside the function: {'Bananas': 4, 'Pears': 4, 'Kiwis': 3, 'Apples': 7}

This means that the changes that have been made inside the function still have an effect on fruit\_amount outside the function.

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

List and dictionary comprehensions are ways to create new lists/dictionaries based on values from an already existing list/dictionary. Comprehensions are used because they offer a shorter syntax.

Example (without list comprehension and with list comprehension):

**LIST without comprehension. We are trying to create a new list with only clothes that contain the letter “I” ([‘skirts’, ‘shirts’])**

clothes = ["shorts", "skirts", "shirts", "trousers"]

newlist = []

for x in clothes:

if "i" in x:

newlist.append(x)

print(newlist)

**LIST comprehension**

clothes = ["shorts", "skirts", "shirts", "trousers"]

newlist = [x for x in clothes if "i" in x]

print(newlist)

**GENERAL SYNTAX for list comprehension** (source: https://www.w3schools.com/python/python\_lists\_comprehension.asp)

newlist = [expression for item in iterable if condition == True]

**GENERAL SYNTAX for dictionary comprehension** (source: https://www.programiz.com/python-programming/dictionary-comprehension)

dictionary = {key: value for vars in iterable}

1. What is namespace in Python?

A name in Python is an identifier we use for variables.

Example:

number\_of\_cats = 333

Number\_of\_cats would be a name, and 333 an object.

A namespace would be a collection of all the names (including built-in data types, functions…) and their objects in our module.

1. What is pass in Python? It is a placeholder for future code used in functions. Nothing happens when we execute it.
2. What is unit test in Python?

A unit test checks if a smaller block of code works. Example:

def money\_out(a, b):  
 return(a - b)  
  
class TestMoneyOut(unittest.TestCase):  
 def test\_money\_out(self):  
 self.assertEqual(money\_out(10, 5), 5) #This would return a OK test  
 self.assertEqual(money\_out(10, 5), 4) #This would return a FAILED test

1. In Python what is slicing? Slicing is used to return specific characters from a variable. Example:

slicing\_example = "Hello!"

print(slicing\_example[2:5])

Result: llo

1. What is a negative index in Python? In Python, one can start indexing from the left side of an object (0, 1, 2..), but also from the right side. Therefore, the last value would be -1, the second to last, -2…
2. How can the ternary operators be used in python? Give an example.

They are conditional expressions (if, elif, else). Example:

# rain = input("Is it raining?")  
# if rain == ("y"):  
# print("You need an umbrella")  
# elif rain == ("n"):  
# print("You don't need an umbrella")

1. What does this mean: \*args, \*\*kwargs? And why would we use it?

Thanks to \*args, \*\*kwargs we can pass as many arguments as we want to our function. \*args allows to pass non-keyworded arguments, while \*\*kwargs is limited to keyworded arguments.

1. How are range and xrange different from one another? Xrange() no longer exists on Python 3, although it is faster than range(). Range() returns a sequence of numbers.

Example:

for i in range(0, 20):

print(i)

The result would be all numbers from 0 until 19 (both included).

1. What is Flask and what can we use it for? Flask is a micro web framework. It is used to build web applications with Python. Thanks to Flask, users can connect to
2. What are clustered and non-clustered index in a relational database?

A clustered index can only appear in tables with no repeated values. Therefore, they can only be created in a table containing a primary key. When a table has a clustered index, the table is called a clustered table.

Non-clustered indexes are separate from the data in the table.

1. What is a ‘deadlock’ a relational database? A deadlock occurs when two or more transactions are waiting for each other to end but none can do so because they are waiting for the other to release its locks.
2. What is a ‘livelock’ a relational database? A livelock happens when a request for an exclusive lock is continually turned down because there are too many overlapping shared locks. The processes' constant status changes make it impossible for them to finish the task.

**29 points**

**2. Python string methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| **capitalize()** | It converts the first character to upper case | example = "hello"  capital\_first\_letter = example.capitalize()  print(capital\_first\_letter)  Result: Hello |
| **casefold()** | Converts string into lower case (all characters) | example = "Hello"  lower\_case = example.casefold()  print(lower\_case)  Result: hello |
| **center()** | This method will align the string to the center, using a specified character (space is default) as the fill character. | example = "Hello"  centered\_string = example.center(10)  print(centered\_string)  Result:    Hello |
| **count()** | It returns how many times a specified value appears in a string | example = "My dog is the prettiest dog in town"  number\_of\_dogs = example.count("dog")  print(number\_of\_dogs)  Result: 2 |
| **endswith()** | This method returns True if the string ends with the chosen value. If the string does not end with the specified value, it returns False. Endswith() returns Boolean values | example = "Hello"  end\_of\_sentence = example.endswith("o")  print(end\_of\_sentence)  Result: True  Conversely,  example = "Hello"  end\_of\_sentence = example.endswith("l")  print(end\_of\_sentence)  Result: False |
| **find()** | This method returns the position of a specified value. If the string does not contain this value, it returns -1 | example = "Hello, my name is Aurora"  finding\_name = example.find("name")  print(finding\_name)  Result: 10 |
| **format()** | The format() method formats the specified value(s) and inserts them inside the string's placeholder.  The placeholder is defined using curly brackets: {}.  The format() method returns the formatted string. | example = "Hello, my name is Aurora"  next\_sentence = "I live in the south of England"  complete\_sentence = "{} and {}".format(example, next\_sentence)  print(complete\_sentence)  Result:  Hello, my name is Aurora and I live in the south of England |
| **index()** | This method is almost the same as find. The only difference is that find returns -1 if the value is not found, while index() raises and exception | example = "Hello, my name is Aurora"  finding\_name = example.find("name")  print(finding\_name)  Result: 10 |
| **isalnum()** | This method returns true if the characters in a string are alphanumeric (a-z and numbers 0-9). | example = "MynameisAurora"  alphanumeric\_characters = example.isalnum()  print(alphanumeric\_characters)  Result: True  Conversely:  example = "My name is Aurora"  alphanumeric\_characters = example.isalnum()  print(alphanumeric\_characters)  Result: False 🡨 this happens because spaces are not alphanumeric characters |

|  |  |  |
| --- | --- | --- |
| **isalpha()** | This method returns true if the characters in a string are in the alphabet (a-z). | example = "MynameisAurora"  alphabet\_characters = example.isalpha()  print(alphabet\_characters)  Result: True |
| **isdigit()** | This method returns true if the string is formed of digits. This includes exponents too! If the string is not formed of digits, we get a traceback. | example = "1900"  digits = example.isdigit()  print(digits)  Result: True |
| **islower()** | This method returns true if all words in the string are lower case. | example = "my name is aurora"  lower\_case\_sentence = example.islower()  print(lower\_case\_sentence)  Result: True |
| **isnumeric()** | This method returns True if all characters in the string are numeric. This DOES NOT include floating numbers or negative numbers. | example = "1900"  numbers = example.isnumeric()  print(numbers)  Result: True  Conversely:  example = "-1900"  numbers = example.isnumeric()  print(numbers)  Result: False |
| **isspace()** | This method returns True if all characters in a string are whitespace. If not, False. | example = " "  spaces = example.isspace()  print(spaces) |
| **istitle()** | This method returns True if the first character in every word is capitalised. | example = "Pretty Little Liars"  title\_or\_not = example.istitle()  print(title\_or\_not)  Result: True |
| **isupper()** | This method returns True if all letters in a string are uppercase. | example = "PRETTY LITTLE LIARS"  capitals = example.isupper()  print(capitals)  Result: true |
| **join()** | This method puts items in tuples or dictionaries into one single string with a separator of our choice | cute\_tuple = ("Hello Kitty", "Pusheen")  friends = " and ".join(cute\_tuple)  print(friends)  Result:  Hello Kitty and Pusheen |
| **lower()** | This method makes every letter in a string lowercase | example = "My name is Aurora"  all\_lowercase = example.lower()  print(all\_lowercase)  Result: my name is aurora |
| **lstrip()** | This method removes leading characters in a string. Spaces are the main leading character. | example = " lots of spaces on the left side"  no\_leading\_characters = example.lstrip()  print(no\_leading\_characters, "not there anymore")  Result:  lots of spaces on the left side not there anymore |
| **replace()** | This method replaces a given phrase with a phrase of our choice. | example = "My dog is prettier than your dog"  replaced\_words = example.replace("dog", "cat")  print(replaced\_words)  Result:  My cat is prettier than your cat |
| **rsplit()** | This method converts a string into a list.  The first parameter inside of the () is the separator, which is optional. The default value is whitespace. This is optional.  The second parameter inside of the () is the maxplit. This tells us how many splits to do. The default value is -1 (it splits all occurrences) | example = "My dog is prettier than your dog"  splitting\_a\_string = example.rsplit()  print(splitting\_a\_string)  Result:  ['My', 'dog', 'is', 'prettier', 'than', 'your', 'dog'] |
| **rstrip()** | This method removes characters such as spaces or commas at the end of a string (its right side) | example = "lots of spaces on the right side "  right\_side = example.rstrip()  print(right\_side, "not there anymore")  Result: lots of spaces on the right side not there anymore |
| **split()** | Very similar to rsplit. It converts a string into a list. The only difference between rsplit() and split() is when the maxplit happens to be specified. In rsplit, the method starts splitting from the right. In split, the method starts splitting from the left side. | example = "My dog is prettier than your dog"  splitting\_a\_string = example.split()  print(splitting\_a\_string)  Result:  ['My', 'dog', 'is', 'prettier', 'than', 'your', 'dog'] |
| **splitlines()** | This method returns a list with after having eliminated \n characters. | example = "this is the first line \n this is the second line"  removed\_characters = example.splitlines()  print(removed\_characters)  Result:  ['this is the first line ', ' this is the second line'] |
| **startswith()** | This method returns True if our string starts with a specific prefix. | example = "Hello, my name is Aurora"  result = example.startswith("Hello, my name")  print(result)  Result: True |
| **strip()** | This method removes characters such as spaces or commas at the beginning AND the end of a string | example = " lots of spaces on left and right sides "  no\_spaces = example.strip()  print(no\_spaces)  Result: lots of spaces on the left and right sides |
| **swapcase()** | This method converts all uppercase to lowercase and lowercase to uppercase. | example = "Pretty Little Liars"  tvseries = example.swapcase()  print(tvseries)  Result: pRETTY lITTLE lIARS |

|  |  |  |
| --- | --- | --- |
| **title()** | This method capitalises the first letter in every word in a string. | example = "pretty little liars"  tvseries = example.title()  print(tvseries)  Result: Pretty Little Liars |
| **upper()** | This method capitalises every letter in a string. | example = "pretty little liars"  tvseries = example.upper()  print(tvseries)  Result: PRETTY LITTLE LIARS |

**11 points**

**3. Python list methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | This method adds an element to the end of a list. | student\_name = [‘Stephen’, ‘Maria’, ‘Greg’, ‘Stephen’]  student\_name.append(‘Louise’)  Result: [‘Stephen’, ‘Maria’, ‘Greg’, ‘Stephen’, ‘Louise’] |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | This method gets rid of all elements in a list. | student\_name = ["Stephen", "Maria", "Greg", "Stephen"]  student\_name.clear()  print(student\_name)  Result: [] |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | This method copies a given list. | student\_name = ["Stephen", "Maria", "Greg", "Stephen"]  student\_name.copy()  print(student\_name)  Result: ["Stephen", "Maria", "Greg", "Stephen"] |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | This method counts the number of times an item in the list appears. | student\_name = ["Stephen", "Maria", "Greg", "Stephen"]  repeated\_student = student\_name.count("Stephen")  print(repeated\_student)Result: 2 |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Extend() adds all the elements inside of a list to the already existing list. | student\_name = ["Stephen", "Maria", "Greg", "Stephen"]  student\_grades=[8, 7, 6, 5]  student\_name.extend(student\_grades)  print(student\_name)  Result: ['Stephen', 'Maria', 'Greg', 'Stephen', 8, 7, 6, 5] |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | This method returns the position of a specified element in a list (0 for the first element, 1 for the second, and so on). | student\_name = ["Stephen", "Maria", "Greg", "Stephen"]  indexed\_student = student\_name.index("Greg")  print(indexed\_student)  Result: 2 |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | This method adds an element defined by us in a given position. The position will be the first parameter in brackets, and the element to be inserted will be the second parameter in brackets. | students = ["Mark", "Sarah", "Lloyd"]  students.insert(2, "Michael")  print(students)  Result: ["Mark", "Sarah", “Michael”, "Lloyd"] |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | This method removes an element from our list. We need to specify the position of the element that needs to be removed in brackets. | students = ["Mark", "Sarah", "Lloyd"]  students.pop(2)  print(students)  Result: ['Mark', 'Sarah'] |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Remove() is very similar to pop(). The only difference is that in brackets we need to write the element that needs to be removed from the list, not its position. | students = ["Mark", "Sarah", "Lloyd"]  students.remove("Lloyd")  print(students)  Result: ['Mark', 'Sarah'] |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverse() reverses the order of a string. The last element appears first. | students = ["Mark", "Sarah", "Lloyd"]  students.reverse()  print(students)  Result:  ['Lloyd', 'Sarah', 'Mark'] |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | It sorts the elements in the list. It can sort the elements in descending order (reverse=True) or in ascending order (reverse=False). Ascending order: a-z /0-9. Descending order: z-a/9-0. | students = ["Mark", "Sarah", "Lloyd"]  students.sort(reverse=False)  print(students)  Result:  ['Lloyd', 'Mark', 'Sarah']  Conversely:  students = ["Mark", "Sarah", "Lloyd"]  students.sort(reverse=True)  print(students)  Result: ['Sarah', 'Mark', 'Lloyd'] |

**2 points**

**4. Python tuple methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**count()**](https://www.w3schools.com/python/ref_tuple_count.asp) | This method counts the number of times an item in the tuple appears. | my\_tuple = ("hi", "hello", "hi", "hola", "bonjour")  counting\_words = my\_tuple.count("hi")  print(counting\_words)  Result: 2 |
| [**index()**](https://www.w3schools.com/python/ref_tuple_index.asp) | This method returns the position of a specified element in a tuple (0 for the first element, 1 for the second, and so on). | my\_tuple = ("hi", "hello", "hola", "bonjour")  indexed\_words = my\_tuple.index("hello")  print(indexed\_words)  Result: 1 |

**11 points**

**5. Python dictionary methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [clear()](https://www.w3schools.com/python/ref_dictionary_clear.asp) | This method gets rid of all elements in a dictionary. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  wardrobe.clear()  print(wardrobe)  Result: {} |
| [copy()](https://www.w3schools.com/python/ref_dictionary_copy.asp) | This method copies the dictionary we are applying the method to. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  wardrobe.copy()  print(wardrobe)  Result: {'shoes': 4, 'skirts': 7, 'trousers': 9} |
| [fromkeys()](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp) | This method returns a new dictionary with the keys we have specified in previous variable. Optionally, we can also create a variable to specify a single value we are passing to all the keys | keys = ("key 1", "key 2", "key 3")  value = 10  mydict = dict.fromkeys(keys, value)  print(mydict)  Result: {'key 1': 10, 'key 2': 10, 'key 3': 10} |
| [get()](https://www.w3schools.com/python/ref_dictionary_get.asp) | This method returns the value of a specified key. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  number\_of\_shoes = wardrobe.get("shoes")  print(number\_of\_shoes)  Result: 4 |
| [items()](https://www.w3schools.com/python/ref_dictionary_items.asp) | This method returns a view object with the key-value pairs in the dictionary. It basically converts the dictionary into a list. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  key\_value\_pairs = wardrobe.items()  print(key\_value\_pairs)  Result:  dict\_items([('shoes', 4), ('skirts', 7), ('trousers', 9)]) |
| [keys()](https://www.w3schools.com/python/ref_dictionary_keys.asp) | This method is similar to items() because it also returns a view object (dictionary as a list). It returns the keys from the dictionary. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  just\_keys = wardrobe.keys()  print(just\_keys)  Result: dict\_keys(['shoes', 'skirts', 'trousers']) |
| [pop()](https://www.w3schools.com/python/ref_dictionary_pop.asp) | It removes a specified item from our dictionary. We need to specify the key name of the item that will be removed. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  wardrobe.pop("shoes")  print(wardrobe)  Result:  {'skirts': 7, 'trousers': 9} |
| [popitem()](https://www.w3schools.com/python/ref_dictionary_popitem.asp) | This method removes the last key and value from the dictionary. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  wardrobe.popitem()  print(wardrobe)  Result: {'shoes': 4, 'skirts': 7} |

|  |  |  |
| --- | --- | --- |
| [setdefault()](https://www.w3schools.com/python/ref_dictionary_setdefault.asp) | This method is similar to get(), as both return the value of a specified key. However, setdefault() offers a value to be returned if the key is not found. | Example when key exists on the dictionary  wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  x = wardrobe.setdefault("skirts")  print(x)  Result: 7  Example when key does NOT exist in the dictionary  wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  x = wardrobe.setdefault("dresses", "fab!")  print(x)  Result: fab! |
| [**update()**](https://www.w3schools.com/python/ref_dictionary_update.asp) | This method updates the dictionary. We can pass a new key and a new value. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  wardrobe.update({"dresses": 10})  print(wardrobe)  Result: {'shoes': 4, 'skirts': 7, 'trousers': 9, 'dresses': 10} |
| [**values()**](https://www.w3schools.com/python/ref_dictionary_values.asp) | It returns the values from the dictionary. It returns a list. | wardrobe = {  "shoes": 4,  "skirts": 7,  "trousers": 9  }  just\_values = wardrobe.values()  print(just\_values)  Result:  dict\_values([4, 7, 9]) |

**12 points**

**6. Python set methods:**

**describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) | It adds a new item to the set. | set = {"tennis", "squash", "volleyball"}  set.add("football")  print(set)  Result: {'football', 'tennis', 'volleyball', 'squash'} |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | It clears all elements in a set. | set = {"tennis", "squash", "volleyball"}  set.clear()  print(set)  Result: set() |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | It duplicates the set | set = {"tennis", "squash", "volleyball"}  set.copy()  print(set)  Result: {'tennis', 'volleyball', 'squash'}  \*\* Process exited - Return Code: 0 \*\*  Press Enter to exit terminal |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | It creates a new set with the items that appear in the first set but not in the second one. | set1 = {"tennis", "squash", "volleyball"}  set2 = {"tennis", "rubgy", "netball"}  set3 = set.difference(set1, set2)  print(set3)  Result: {'squash', 'volleyball'}  \*\* Process exited - Return Code: 0 \*\*  Press Enter to exit terminal |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | It is the opposite of difference(). It returns the items that appear both in the first set and the second set. | set1 = {"tennis", "squash", "volleyball"}  set2 = {"tennis", "rubgy", "netball"}  set3 = set.intersection(set1, set2)  print(set3)  Result: {'tennis'}  \*\* Process exited - Return Code: 0 \*\*  Press Enter to exit terminal |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns True if ALL items in one set appear in another set. | set1 = {"dogs", "cats", "birds", "lizards"}  set2 = {"dogs", "cats", "birds", "lizards", "snakes", "rabbits"}  check = set1.issubset(set2)  print(check)  Result: True |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns True if one set contains ALL items in the other set. | set1 = {"dogs", "cats", "birds", "lizards"}  set2 = {"dogs", "cats", "birds", "lizards", "snakes", "rabbits"}  check = set2.issuperset(set1)  print(check)  Result: True |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | It removes an element in a set. | set = {"tennis", "squash", "volleyball"}  set.pop()  print(set)  Result: {'tennis', 'volleyball'} |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | It removes a specified item in a set. | set = {"tennis", "squash", "volleyball"}  set.remove("tennis")  print(set)  Result: {'squash', 'volleyball'} |
| [symmetric\_differ](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) [ence()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | This method creates a new set with the items that DO NOT appear in both sets. | set1 = {"dogs", "cats", "birds", "lizards"}  set2 = {"dogs", "cats", "birds", "lizards", "snakes", "rabbits"}  check = set1.symmetric\_difference(set2)  print(check)  Result: {‘rabbits’, ‘snakes’} |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | It creates a third set with all elements from the first and second sets. It removes duplicates. | set1 = {"dogs", "cats", "birds", "lizards"}  set2 = {"dogs", "cats", "birds", "lizards", "snakes", "rabbits"}  check = set1.union(set2)  print(check)  Result: {'lizards', 'rabbits', 'snakes', 'cats', 'dogs', 'birds'} |

|  |  |  |
| --- | --- | --- |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | This method updates one set by adding the non-repeated items from another set. | set1 = {"dogs", "cats", "birds", "lizards"}  set2 = {"dogs", "cats", "birds", "lizards", "snakes", "rabbits"}  set1.update(set2)  print(set1)  Result: {'rabbits', 'snakes', 'lizards', 'birds', 'dogs', 'cats'} |

**5 points**

**7. Python file methods:**

**describe each method and provide an example**

For these last five questions, let’s imagine we have a file called **theory\_questions.txt**

This is the text we find in theory\_questions.txt in our project folder:

*Almost finished with these exercises.*

*It took a long time, but I learnt a lot.*

*I will have to study a lot!*

*Bye!*

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [**read()**](https://www.w3schools.com/python/ref_file_read.asp) | This method reads a specified file. | f = open(“theory\_questions.txt”, “r”)  print(f.read())  Result:  Almost finished with these exercises.  It took a long time, but I learnt a lot.  I will have to study a lot!  Bye! |
| [**readline()**](https://www.w3schools.com/python/ref_file_readline.asp) | This method read ONE line from our file. | f = open(“theory\_questions.txt”, “r”)  print(f.readline())  Result:  Almost finished with these exercises. |
| [**readlines()**](https://www.w3schools.com/python/ref_file_readlines.asp) | This method returns all the lines in a file as if they were a list. Each element of this list is a line in the file | f = open("theory\_questions.txt", "r")  print(f.readlines())  Result:  ['Almost finished with these exercises. \n', 'It took a long time, but I learnt a lot.\n', 'I will have to study a lot!\n', 'Bye! \n'] |
| [**write()**](https://www.w3schools.com/python/ref_file_write.asp) | This method writes text to the file. If we choose “a” in the file mode, the text will be added at the end of the file. If we choose “w”, the previous text in the previous file will be deleted and replace by the new text. | f = open("theory\_questions.txt ", "w") f.write("So much typing!")  Result: So much typing! |
| [**writelines()**](https://www.w3schools.com/python/ref_file_writelines.asp) | Writelines() writes the items of a list to the file. Same logic with “a” and “w”. | f = open("theory\_questions.txt ", "w") f.writelines("\nSo much typing!")  Result: So much typing! |