**NOTE: This is an official document by Indexademics. Unless otherwise stated, this document may not be accredited to individuals or groups other than the club IDX, nor should this document be distributed, sold, or modified for personal use in any way.**

**IDX G9 COMPUTER SCIENCE H STUDY GUIDE**

**ISSUE 4**

**By Gavin**

**Note: For the final review, Lesson 12-15 and all previous lessons would also be covered. It is strongly suggested to view CS H Review Guide ISSUE 3 also.**

**Lesson 16 Dictionaries**

A dictionary in Python is a collection of key-value pairs that allows for the storage and retrieval of data in an efficient manner.

A dictionary is defined using curly braces   **{}**

Each key-value pair is separated by a colon **:**

Pairs are separated by commas  **,**

Example Usage | Defining a dictionary

my\_dict = {

"name": "Ivan",

"age": 114514,

"city": "Shanghai",

}

The information in the bracket is structured in the format “key”: “value”.

To retrieve from a dictionary the value of a certain key, the following grammar is used:

Dictionary\_name[“key”]

Example Usage | Retrieving a value

print(my\_dict["name"]) # Output: Ivan

To modify the value of a certain key in a dictionary, or to add value to a new key, use the grammar:

Dictionary\_name[“key”] = new\_value

To remove a key, use the grammar:

del Dictionary\_name[“key”]

A dictionary can be iterated with the following grammar:

for key, value in Dictionary\_name.items():

print(f"{key}: {value}")

**Lesson 17 Comments**

Comments are used to annotate code, providing notes not executed by the Python Interpreter.

There are two kinds of comments: Single-line and Multiple-line.

**Single-line:**

#*comments*

**Mutiple-line:**

' ' '

*comments*

' ' '

Example Usage | Single Line Comment

# This is a single-line comment

print("Hello, World!") # This prints a message but this comment won’t have any impact

Example Usage | Multiple Line Comment

' ' '

This is a multi-line comment.

It can span multiple lines.

' ' '

print("Hello again!")

**Lesson 18 File Paths**

Lets start with a quick review of opening and reading files in python:

Opening a file

file = open('filename', mode)

Reading a file

with open('example.txt', 'r') as file:

content = file.read()

print(content)

Line Stripping

with open('example.txt', 'r') as file:

for line in file:

print(line.strip())

Readlines() method

with open('example.txt', 'r') as file:

lines = file.readlines()

print(lines)

Writing Data

with open('example.txt', 'w') as file:

file.write("Hello, World!")

Appending Data

with open('example.txt', 'a') as file:

file.write("\nHello, World!")

You can open a file for both reading and writing using the 'r+' mode:

with open('example.txt', 'r+') as file:

content = file.read()

file.write("\nNew content added.")

When using open(), always close the file after use to free system resources:

file = open('example.txt', 'r')

# Perform operations

file.close()

Alternatively, use the with statement, which automatically closes the file:

with open('example.txt', 'r') as file:

content = file.read()

To handle errors (e.g., file not found), use try-except blocks:

try:

with open('nonexistent.txt', 'r') as file:

content = file.read()

except FileNotFoundError:

print("File not found!")

Here’s the grammar structure of try-except blocks:

try:

# Code that may raise an error

risky\_operation()

except SomeException as e:

# Code that runs if an exception occurs

print(f"An error occurred: {e}")

# e is the error that occured