

Python Exercises Report

Hussein Hussein
Student ID: 58301

Repository Information

GitHub Repository: <https://github.com/Leo-devv/cloud-programming-labs>

Project Overview

This report documents 50 Python exercises covering Python fundamentals, organized into three stages focusing on different aspects of the language.

Repository Structure

The project structure follows course requirements:

```
python/
    ├── task0/      Setup and Workflow (3 files)
    ├── task1/      Variables and Types (10 files)
    ├── task1b/     Conditional Statements (3 files)
    ├── task1c/     match/case Patterns (3 files)
    ├── task2/      List Operations (6 files)
    ├── task2b/     Loop Constructs (6 files)
    ├── task3a/     Dict Manipulation (6 files)
    ├── task3b/     Lambda Functions (6 files)
    └── task3c/     Pipelines and Composition (7 files)
```

Leo-devv TASK3C: S3_PIPE_07 bonus + Python report and README update		
Name	Last commit message	Last commit date
..		6632839 · 3 days ago
task0	TASK0: S0_SETUP_01-03 - Python setup and workflow	3 days ago
task1	TASK1: S1_VAR_01-10 - Python variables and runtime types	3 days ago
task1b	TASK1B: S1_IF_01-03 - Python if/elif/else branching	3 days ago
task1c	TASK1C: S1_MC_01-03 - Python match/case pattern matching	3 days ago
task2	TASK2: S2_LIST_01-06 - Python list operations	3 days ago
task2b	TASK2B: S2_FOR_01-06 - Python loops	3 days ago
task3a	TASK3A: S3_DICT_01-06 - Python dict operations	3 days ago
task3b	TASK3B: S3_LAM_01-06 - Python lambda functions	3 days ago
task3c	TASK3C: S3_PIPE_07 bonus + Python report and README update	3 days ago

Files follow the naming convention S[stage]_[type]_[number].py.

Task 0: Setup and Workflow

Naming

File	Description
S0_SETUP_01.py	Create project and print message
S0_SETUP_02.py	File structure documentation
S0_SETUP_03.py	eq() test helper function

Stage 1: Variables, Types, and Control Flow

Task 1: Variables and Types

File	Description
S1_VAR_01.py	Variable declarations and type() catalog
S1_VAR_02.py	Rebinding and dynamic typing
S1_VAR_03.py	Mutable vs immutable (list vs tuple)
S1_VAR_04.py	Identity vs equality (is vs ==)
S1_VAR_05.py	Truthiness with isTruthy()
S1_VAR_06.py	Safe conversion with to_int_or_none()
S1_VAR_07.py	NaN and math.isnan
S1_VAR_08.py	Big integers - Python int is unlimited
S1_VAR_09.py	Type hints are not runtime enforcement
S1_VAR_10.py	inspect() returns type info dict

Task 1B: Conditional Statements

File	Description
S1_IF_01.py	shipping_cost with weight rules + member discount
S1_IF_02.py	grade(score) returns A-F or None
S1_IF_03.py	normalize_name handles falsy values

Task 1C: match/case Patterns (Python 3.10+)

File	Description
S1_MC_01.py	day_name(1-7) using match/case

S1_MC_02.py	run_command returns message string
S1_MC_03.py	calc with division by zero handling

Stage 2: Lists and Loops

Task 2: List Operations

File	Description
S2_LIST_01.py	clean_numbers drops invalid values
S2_LIST_02.py	unique() without using set
S2_LIST_03.py	chunk() with size validation
S2_LIST_04.py	flatten1() one level only
S2_LIST_05.py	stats() returns min/max/avg dict
S2_LIST_06.py	Transform user records

Task 2B: Loop Constructs

File	Description
S2_FOR_01.py	FizzBuzz implementation
S2_FOR_02.py	find_first_even
S2_FOR_03.py	sum_until stops at threshold
S2_FOR_04.py	count_occurrences returns counts dict
S2_FOR_05.py	10x10 multiplication table
S2_FOR_06.py	sum_nested for matrix

Stage 3: Dicts, Functions, and Pipelines

Task 3A: Dict Operations

File	Description
S3_DICT_01.py	get(obj, "a.b.c", fallback) safe access
S3_DICT_02.py	merge_defaults, no mutation
S3_DICT_03.py	pick(obj, keys)
S3_DICT_04.py	omit(obj, keys)
S3_DICT_05.py	invert with list for duplicates
S3_DICT_06.py	group_by(items, key)

Task 3B: Functions and Lambdas

File	Description
S3_LAM_01.py	square, is_odd, greet as lambdas
S3_LAM_02.py	Sort by key with lambda
S3_LAM_03.py	make_adder closure factory
S3_LAM_04.py	filter -> map -> reduce chain
S3_LAM_05.py	at_least(min) higher-order predicate
S3_LAM_06.py	map_values(obj, fn)

Task 3C: Pipelines

File	Description
S3_PIPE_01.py	pipe() left-to-right composition
S3_PIPE_02.py	compose() right-to-left composition
S3_PIPE_03.py	String normalization pipeline
S3_PIPE_04.py	Generator-based iterable pipeline
S3_PIPE_05.py	Log lines parser pipeline
S3_PIPE_06.py	pipe_safe with ok/error dict
S3_PIPE_07.py	Bonus: itertools (islice, groupby) + memory efficiency

Git Workflow

Commit History

Development progressed through incremental commits:

TASK3C: S3_PIPE_07 bonus + Python report and README update	6632839		
Leo-devv committed 3 days ago			
TASK3C: S3_PIPE_01-06 - Python pipelines	8045f8e		
Leo-devv committed 3 days ago			
TASK3B: S3_LAM_01-06 - Python lambda functions	f9c3d00		
Leo-devv committed 3 days ago			
TASK3A: S3_DICT_01-06 - Python dict operations	3ea4280		
Leo-devv committed 3 days ago			
TASK2B: S2_FOR_01-06 - Python loops	24bfdbe		
Leo-devv committed 3 days ago			
TASK2: S2_LIST_01-06 - Python list operations	61cd5b1		
Leo-devv committed 3 days ago			
TASK1C: S1_MC_01-03 - Python match/case pattern matching	acc9fa8		
Leo-devv committed 3 days ago			
TASK1B: S1_IF_01-03 - Python if/elif/else branching	d5f5252		
Leo-devv committed 3 days ago			
TASK1: S1_VAR_01-10 - Python variables and runtime types	bdb4e30		
Leo-devv committed 3 days ago			
TASK0: S0_SETUP_01-03 - Python setup and workflow	f8276e2		
Leo-devv committed 3 days ago			

Tags

Tag
py-task1-done
py-task1b-done
py-task1c-done
py-task2-done
py-task2b-done
py-task3a-done
py-task3b-done
py-task3c-done
py-all-done

Testing

All exercises were tested using Python 3.12. Each file contains executable code with output verification.

To run any exercise:

```
python python/task1/S1_VAR_01.py
```

The repository can be tested using Python 3.12+ locally or in Google Colab/PythonAnywhere

Screenshots

Exercise Output

Name	Value	type(x)	type(x).__name__
my_int	42	<class 'int'>	int
my_float	3.14	<class 'float'>	float
my_str	'hello'	<class 'str'>	str
my_bool	True	<class 'bool'>	bool
my_none	None	<class 'NoneType'>	NoneType
my_list	[1, 2, 3]	<class 'list'>	list
my_tuple	(1, 2, 3)	<class 'tuple'>	tuple
my_dict	{'a': 1}	<class 'dict'>	dict
my_set	{1, 2, 3}	<class 'set'>	set
my_func	<function>	<class 'function'>	function

```
>> python S0_SETUP_02.py
>> python S0_SETUP_03.py
It works
Setup complete - ready for exercises!
Project structure:
python/task0/
python/task1/
python/task1b/
python/task1c/
python/task2/
python/task2b/
python/task3a/
python/task3b/
python/task3c/

Files will be named like: S1_VAR_01.py, S2_LIST_01.py, etc.
All eq() tests passed!
```



```
Problems Output Debug Console Terminal Ports Query Results (Preview) DevDb GitLens ... -  
==== TASK1B: if/elif/else ===  
weight member cost  
-----  
0.5 False 10  
1 False 10  
1 True 8.0  
1.1 False 20  
5 False 20  
5 True 16.0  
5.1 False 30  
10 True 24.0  
-1 False None  
abc False None  
score grade  
-----  
59 F  
60 D  
69 D  
70 C  
79 C  
80 B  
89 B  
90 A  
100 A  
101 None  
-1 None  
0 F  
==== TASK1C: match/case ===  
day_name(0): None  
day_name(1): Monday  
day_name(2): Tuesday  
day_name(3): Wednesday  
day_name(4): Thursday  
day_name(5): Friday  
day_name(6): Saturday  
day_name(7): Sunday  
day_name(8): None  
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.
```

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win3
2
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
== TASK2B: Loops ===

1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
FizzBuzz
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win3
2
Type "help", "copyright", "credits" or "license" for more information.
```

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win3
2
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
== TASK2B: Loops ===

1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
FizzBuzz
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win3
2
Type "help", "copyright", "credits" or "license" for more information.
```

```
0
>>> print("== TASK3A: Dictionaries =="); exec(open('python/task3a/S3_DICT_01.py').read());
== TASK3A: Dictionaries ==
Leo
Warsaw
None
default
```

[x] node
[x] python

```

==== TASK3B: Lambda ====
1. Filter evens: [2, 4, 6, 8, 10]
2. Square them: [4, 16, 36, 64, 100]
3. Sum: 220

One-liner result: 220
one-liner result: 220
>>> print("== TASK3C: Pipelines =="); exec(open('python/task3c/S3_PIPE_06.py').read());

== TASK3C: Pipelines ==
safe_math(5): {'ok': True, 'value': 11}
safe_math(-5): {'ok': False, 'error': 'Value must be positive'}

safe_parse('10'): {'ok': True, 'value': 20}
safe_parse('abc'): {'ok': False, 'error': "invalid literal for int() with base 10: 'abc'"}
>>> exec(open('python/task3c/S3_PIPE_07.py').read())
  File "<stdin>", line 1
    exec(open('python/task3c/S3_PIPE_07.py').read())
IndentationError: unexpected indent
>>> exec(open('python/task3c/S3_PIPE_07.py').read())
Logs grouped by level:
  DEBUG: 1 entries
  ERROR: 2 entries
  INFO: 4 entries

First 3 logs using islice:
  {'level': 'INFO', 'msg': 'Started'}
  {'level': 'INFO', 'msg': 'Processing'}
  {'level': 'ERROR', 'msg': 'Failed'}

Items 100-105 from infinite generator:
[100, 101, 102, 103, 104, 105]

```

Git Tags



py-task3a-done

(⌚ 3 days ago -O- 3ea4280

py-task2b-done

(⌚ 3 days ago -O- 24bfdbc

py-task2-done

(⌚ 3 days ago -O- 61cd5b1

py-task1c-done

(⌚ 3 days ago -O- acc9fa8

py-task1b-done

(⌚ 3 days ago -O- d5f5252

py-task1-done

(⌚ 3 days ago -O- bdb4e30

py-task0-done

(⌚ 3 days ago -O- f8276e2

py-all-done

(⌚ 3 days ago -O- 8045f8e

Summary

All exercises are complete and tested. The project follows course guidelines for structure, file naming, and version control. Tags enable easy navigation during evaluation.