(/)

0x03. Python - Data Structures: Lists, Tuples

By Guillaume

Weight: 1

Ongoing project - started Jun 3, 2022, must end by Jun 7, 2022 - you're done with 29% of tasks.

Checker was released at Jun 4, 2022 6:00 AM

An auto review will be launched at the deadline

Resources

Read or watch:

- 3.1.3. Lists (/rltoken/VarQbHxfmbnpGnaGp3Nb_A)
- Data structures (/rltoken/2aa8Mp-V2eSieGeX3OX8yQ) (*until `5.3. Tuples and Sequences` included*)
- Learn to Program 6 : Lists (/rltoken/BX2_CuHj1sq4eYGiXbCYSg)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/qZrNhvUqi5zcqE4cMFGU6Q), **without the help of Google**:

General

- Why Python programming is awesome
- What are lists and how to use them
- What are the differences and similarities between strings and lists
- What are the most common methods of lists and how to use them
- How to use lists as stacks and queues
- What are list comprehensions and how to use them
- What are tuples and how to use them
- When to use tuples versus lists
- What is a sequence
- What is tuple packing
- What is sequence unpacking
- What is the `del` statement and how to use it

Copyright - Plagiarism

(/)

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

Python Scripts

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using python3 (version 3.8.5)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/python3`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the pycodestyle (version `2.8.*`)
- All your files must be executable
- The length of your files will be tested using `wc`

C

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using python3 (version 3.8.5)
- All your files should end with a new line
- Your code should use the `Betty` style. It will be checked using betty-style.pl (https://github.com/holbertonschool/Betty/blob/master/betty-style.pl) and betty-doc.pl (https://github.com/holbertonschool/Betty/blob/master/betty-doc.pl)
- You are not allowed to use global variables
- No more than 5 functions per file
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions should be included in your header file called `lists.h`
- Don't forget to push your header file
- All your header files should be include guarded

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Print a list of integers

mandatory

Write a function that prints all integers of a list.

- Prototype: `def print_list_integer(my_list=[]):`
- Format: one integer per line. See example
- You are not allowed to import any module
- You can assume that the list only contains integers
- You are not allowed to cast integers into strings
- You have to use `str.format()` to print integers

```
guillaume@ubuntu:~/0x03$ cat 0-main.py
#!/usr/bin/python3
print_list_integer = __import__('0-print_list_integer').print_list_integer

my_list = [1, 2, 3, 4, 5]
print_list_integer(my_list)

guillaume@ubuntu:~/0x03$ ./0-main.py
1
2
3
4
5
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `0-print_list_integer.py`

Done! | Help | Check your code | >_ Get a sandbox

1. Secure access to an element in a list

mandatory

Write a function that retrieves an element from a list like in C.

- Prototype: `def element_at(my_list, idx):`
- If `idx` is negative, the function should return `None`
- If `idx` is out of range (> of number of element in `my_list`), the function should return `None`
- You are not allowed to import any module
- You are not allowed to use `try/except`

```
guillaume@ubuntu:~/0x03$ cat 1-main.py
#!/usr/bin/python3
element_at = __import__('1-element_at').element_at

my_list = [1, 2, 3, 4, 5]
idx = 3
print("Element at index {:d} is {}".format(idx, element_at(my_list, idx)))

guillaume@ubuntu:~/0x03$ ./1-main.py
Element at index 3 is 4
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `1-element_at.py`

☑ Done! | Help | Check your code | >_ Get a sandbox

2. Replace element

mandatory

Write a function that replaces an element of a list at a specific position (like in C).

- Prototype: `def replace_in_list(my_list, idx, element):`
- If `idx` is negative, the function should not modify anything, and returns the original list
- If `idx` is out of range (> of number of element in `my_list`), the function should not modify anything, and returns the original list
- You are not allowed to import any module
- You are not allowed to use `try/except`

```
guillaume@ubuntu:~/0x03$ cat 2-main.py
#!/usr/bin/python3
replace_in_list = __import__('2-replace_in_list').replace_in_list

my_list = [1, 2, 3, 4, 5]
idx = 3
new_element = 9
new_list = replace_in_list(my_list, idx, new_element)

print(new_list)
print(my_list)

guillaume@ubuntu:~/0x03$ ./2-main.py
[1, 2, 3, 9, 5]
[1, 2, 3, 9, 5]
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `2-replace_in_list.py`

Done! | Help | Check your code | >_ Get a sandbox

3. Print a list of integers... in reverse!

mandatory

Write a function that prints all integers of a list, in reverse order.

- Prototype: `def print_reversed_list_integer(my_list=[]):`
- Format: one integer per line. See example
- You are not allowed to import any module
- You can assume that the list only contains integers
- You are not allowed to cast integers into strings
- You have to use `str.format()` to print integers

```
guillaume@ubuntu:~/0x03$ cat 3-main.py
#!/usr/bin/python3
print_reversed_list_integer = __import__('3-print_reversed_list_integer').print_reversed
_list_integer

my_list = [1, 2, 3, 4, 5]
print_reversed_list_integer(my_list)

guillaume@ubuntu:~/0x03$ ./3-main.py
5
4
3
2
1
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `3-print_reversed_list_integer.py`

Done! | Help | Check your code | >_ Get a sandbox

4. Replace in a copy

mandatory

Write a function that replaces an element in a list at a specific position without modifying the original list (like in C).

- Prototype: `def new_in_list(my_list, idx, element):`
- If `idx` is negative, the function should return a copy of the original `list`
- If `idx` is out of range (> of number of element in `my_list`), the function should return a copy of the original `list`
- You are not allowed to import any module
- You are not allowed to use `try/except`

```
guillaume@ubuntu:~/0x03$ cat 4-main.py
#!/usr/bin/python3
new_in_list = __import__('4-new_in_list').new_in_list

my_list = [1, 2, 3, 4, 5]
idx = 3
new_element = 9
new_list = new_in_list(my_list, idx, new_element)

print(new_list)
print(my_list)

guillaume@ubuntu:~/0x03$ ./4-main.py
[1, 2, 3, 9, 5]
[1, 2, 3, 4, 5]
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `4-new_in_list.py`

Done? | Help | Check your code | >_ Get a sandbox

5. Can you C me now?

mandatory

Write a function that removes all characters `c` and `C` from a string.

- Prototype: `def no_c(my_string):`
- The function should return the new string
- You are not allowed to import any module
- You are not allowed to use `str.replace()`

```
guillaume@ubuntu:~/0x03$ cat 5-main.py
#!/usr/bin/env python3
no_c = __import__('5-no_c').no_c

print(no_c("Best School"))
print(no_c("Chicago"))
print(no_c("C is fun!"))

guillaume@ubuntu:~/0x03$ ./5-main.py
Best Shool
hiago
 is fun!
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `5-no_c.py`

Done? | Help | Check your code | >_ Get a sandbox

6. Lists of lists = Matrix

mandatory

Write a function that prints a matrix of integers.

- Prototype: `def print_matrix_integer(matrix=[[]]):`
- Format: see example
- You are not allowed to import any module
- You can assume that the list only contains integers
- You are not allowed to cast integers into strings
- You have to use `str.format()` to print integers

```
guillaume@ubuntu:~/0x03$ cat 6-main.py
#!/usr/bin/python3
print_matrix_integer = __import__('6-print_matrix_integer').print_matrix_integer

matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

print_matrix_integer(matrix)
print("--")
print_matrix_integer()

guillaume@ubuntu:~/0x03$ ./6-main.py | cat -e
1 2 3$
4 5 6$
7 8 9$
--$
$
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `6-print_matrix_integer.py`

Done? | Help | Check your code | >_ Get a sandbox

7. Tuples addition

mandatory

Write a function that adds 2 tuples.

- Prototype: `def add_tuple(tuple_a=(), tuple_b=()):`
- Returns a tuple with 2 integers:
  - The first element should be the addition of the first element of each argument
  - The second element should be the addition of the second element of each argument
- You are not allowed to import any module
- You can assume that the two tuples will only contain integers
- If a tuple is smaller than 2, use the value `0` for each missing integer
- If a tuple is bigger than 2, use only the first 2 integers

```
guillaume@ubuntu:~/0x03$ cat 7-main.py
#!/usr/bin/python3
add_tuple = __import__('7-add_tuple').add_tuple

tuple_a = (1, 89)
tuple_b = (88, 11)
new_tuple = add_tuple(tuple_a, tuple_b)
print(new_tuple)

print(add_tuple(tuple_a, (1, )))
print(add_tuple(tuple_a, ()))

guillaume@ubuntu:~/0x03$ ./7-main.py
(89, 100)
(2, 89)
(1, 89)
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `7-add_tuple.py`

☐ Done? | Help | Check your code | >_ Get a sandbox

8. More returns!

mandatory

Write a function that returns a tuple with the length of a string and its first character.

- Prototype: `def multiple_returns(sentence):`
- If the sentence is empty, the first character should be equal to `None`
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x03$ cat 8-main.py
#!/usr/bin/python3
multiple_returns = __import__('8-multiple_returns').multiple_returns

sentence = "At school, I learnt C!"
length, first = multiple_returns(sentence)
print("Length: {:d} - First character: {}".format(length, first))

guillaume@ubuntu:~/0x03$ ./8-main.py
Length: 22 - First character: A
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`

- Directory: `0x03-python-data_structures`
- File: `8-multiple_returns.py`

Done? | Help | Check your code | >_ Get a sandbox

9. Find the max

mandatory

Write a function that finds the biggest integer of a list.

- Prototype: `def max_integer(my_list=[]):`
- If the list is empty, return `None`
- You can assume that the list only contains integers
- You are not allowed to import any module
- You are not allowed to use the builtin `max()`

```
guillaume@ubuntu:~/0x03$ cat 9-main.py
#!/usr/bin/python3
max_integer = __import__('9-max_integer').max_integer

my_list = [1, 90, 2, 13, 34, 5, -13, 3]
max_value = max_integer(my_list)
print("Max: {}".format(max_value))

guillaume@ubuntu:~/0x03$ ./9-main.py
Max: 90
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `9-max_integer.py`

Done? | Help | Check your code | >_ Get a sandbox

10. Only by 2

mandatory

Write a function that finds all multiples of 2 in a list.

- Prototype: `def divisible_by_2(my_list=[]):`
- Return a new list with `True` or `False`, depending on whether the integer at the same position in the original list is a multiple of 2
- The new list should have the same size as the original list
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x03$ cat 10-main.py
#!/usr/bin/python3
divisible_by_2 = __import__('10-divisible_by_2').divisible_by_2

my_list = [0, 1, 2, 3, 4, 5, 6]
list_result = divisible_by_2(my_list)

i = 0
while i < len(list_result):
    print("{:d} {:s} divisible by 2".format(my_list[i], "is" if list_result[i] else "is not"))
    i += 1

guillaume@ubuntu:~/0x03$ ./10-main.py
0 is divisible by 2
1 is not divisible by 2
2 is divisible by 2
3 is not divisible by 2
4 is divisible by 2
5 is not divisible by 2
6 is divisible by 2
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `10-divisible_by_2.py`

Done? | Help | Check your code | >_ Get a sandbox

11. Delete at

mandatory

Write a function that deletes the item at a specific position in a list.

- Prototype: `def delete_at(my_list=[], idx=0):`
- If `idx` is negative or out of range, nothing change (returns the same list)
- You are not allowed to use `pop()`
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x03$ cat 11-main.py
#!/usr/bin/python3
delete_at = __import__('11-delete_at').delete_at

my_list = [1, 2, 3, 4, 5]
idx = 3
new_list = delete_at(my_list, idx)
print(new_list)
print(my_list)

guillaume@ubuntu:~/0x03$ ./11-main.py
[1, 2, 3, 5]
[1, 2, 3, 5]
guillaume@ubuntu:~/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `11-delete_at.py`

Done? | Help | Check your code | >_ Get a sandbox

12. Switch

mandatory

Complete the source code in order to switch value of `a` and `b`

- You can find the source code here (/rltoken/lwhtw8ZaGLN7TIzodKGnYA)
- Your code should be inserted where the comment is (line 4)
- Your program should be exactly 5 lines long

```
guillaume@ubuntu:~/py/0x03$ ./12-switch.py
a=10 - b=89
guillaume@ubuntu:~/py/0x03$ wc -l 12-switch.py
5 12-switch.py
guillaume@ubuntu:~/py/0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `12-switch.py`

Done? | Help | Check your code | >_ Get a sandbox

13. Linked list palindrome (/)

mandatory

Technical interview preparation:

- You are not allowed to google anything
- Whiteboard first

Write a function in C that checks if a singly linked list is a palindrome.

- Prototype: `int is_palindrome(listint_t **head);`
- Return: `0` if it is not a palindrome, `1` if it is a palindrome
- An empty list is considered a palindrome

```
carrie@ubuntu:0x03$ cat lists.h
#ifndef LISTS_H
#define LISTS_H

/**
 * struct listint_s - singly linked list
 * @n: integer
 * @next: points to the next node
 *
 * Description: singly linked list node structure
 * for project
 */
typedef struct listint_s
{
    int n;
    struct listint_s *next;
} listint_t;

size_t print_listint(const listint_t *h);
listint_t *add_nodeint_end(listint_t **head, const int n);
void free_listint(listint_t *head);

int is_palindrome(listint_t **head);

#endif /* LISTS_H */
carrie@ubuntu:0x03$
```

(7)

```
carrie@ubuntu:0x03$ cat linked_lists.c
#include <stdio.h>
#include <stdlib.h>
#include "lists.h"

/**
 * print_listint - prints all elements of a listint_t list
 * @h: pointer to head of list
 * Return: number of nodes
 */
size_t print_listint(const listint_t *h)
{
    const listint_t *current;
    unsigned int n; /* number of nodes */

    current = h;
    n = 0;
    while (current != NULL)
    {
        printf("%i\n", current->n);
        current = current->next;
        n++;
    }

    return (n);
}

/**
 * add_nodeint_end - adds a new node at the end of a listint_t list
 * @head: pointer to pointer of first node of listint_t list
 * @n: integer to be included in new node
 * Return: address of the new element or NULL if it fails
 */
listint_t *add_nodeint_end(listint_t **head, const int n)
{
    listint_t *new;
    listint_t *current;

    current = *head;

    new = malloc(sizeof(listint_t));
    if (new == NULL)
        return (NULL);

    new->n = n;
    new->next = NULL;

    if (*head == NULL)
        *head = new;
    else
    {
        while (current->next != NULL)
            current = current->next;
        current->next = new;
    }
```

```
    return (new);
}

/**
 * free_listint - frees a listint_t list
 * @head: pointer to list to be freed
 * Return: void
 */
void free_listint(listint_t *head)
{
    listint_t *current;

    while (head != NULL)
    {
        current = head;
        head = head->next;
        free(current);
    }
}
carrie@ubuntu:0x03$
```

(7)

```
carrie@ubuntu:0x03$ cat 13-main.c
#include <stdio.h>
#include <stdlib.h>
#include "lists.h"

/**
 * main - check the code for
 *
 * Return: Always 0.
 */
int main(void)
{
    listint_t *head;

    head = NULL;
    add_nodeint_end(&head, 1);
    add_nodeint_end(&head, 17);
    add_nodeint_end(&head, 972);
    add_nodeint_end(&head, 50);
    add_nodeint_end(&head, 98);
    add_nodeint_end(&head, 98);
    add_nodeint_end(&head, 50);
    add_nodeint_end(&head, 972);
    add_nodeint_end(&head, 17);
    add_nodeint_end(&head, 1);
    print_listint(head);

    if (is_palindrome(&head) == 1)
        printf("Linked list is a palindrome\n");
    else
        printf("Linked list is not a palindrome\n");

    free_listint(head);

    return (0);
}
carrie@ubuntu:0x03$
```

```
carrie@ubuntu:0x03$ gcc -Wall -Werror -Wextra -pedantic 13-main.c linked_lists.c 13-is_p
alindrome.c -o palindrome
carrie@ubuntu:0x03$ ./palindrome
1
17
972
50
98
98
50
972
17
1
Linked list is a palindrome
carrie@ubuntu:0x03$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `13-is_palindrome.c, lists.h`

Done? | Help | Check your code | >_ Get a sandbox

14. CPython #0: Python lists

#advanced

CPython is the reference implementation of the Python programming language. Written in C, CPython is the default and most widely used implementation of the language.
Since we now know a bit of C, we can look at what is happening under the hood of Python. Let's have fun with Python and C, and let's look at what makes Python so easy to use.

- All your files will be interpreted/compiled on Ubuntu 14.04 LTS

Create a C function that prints some basic info about Python lists.

- Prototype: `void print_python_list_info(PyObject *p);`
- Format: see example
- Python version: 3.4
- Your shared library will be compiled with this command line: `gcc -Wall -Werror -Wextra -pedantic -std=c99 -shared -Wl,-soname,PyList -o libPyList.so -fPIC -I/usr/include/python3.4 100-print_python_list_info.c`
- OS: `Ubuntu 14.04 LTS`
- Start by reading:
  - listobject.h
  - object.h
  - Common Object Structures (/rltoken/jmRTk4m1VSzjsu3QTGaC6w)
  - List Objects (/rltoken/7V1HlQRESjCqrKrw_O_Urw)

```
julien@ubuntu:~/CPython$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -shared -Wl,-sonam
e,PyList -o libPyList.so -fPIC -I/usr/include/python3.4 100-print_python_list_info.c
julien@ubuntu:~/CPython$ cat 100-test_lists.py
import ctypes

lib = ctypes.CDLL('./libPyList.so')
lib.print_python_list_info.argtypes = [ctypes.py_object]
l = ['hello', 'World']
lib.print_python_list_info(l)
del l[1]
lib.print_python_list_info(l)
l = l + [4, 5, 6.0, (9, 8), [9, 8, 1024], "My string"]
lib.print_python_list_info(l)
l = []
lib.print_python_list_info(l)
l.append(0)
lib.print_python_list_info(l)
l.append(1)
l.append(2)
l.append(3)
l.append(4)
lib.print_python_list_info(l)
l.pop()
lib.print_python_list_info(l)
julien@ubuntu:~/CPython$ python3 100-test_lists.py
[*] Size of the Python List = 2
[*] Allocated = 2
Element 0: str
Element 1: str
[*] Size of the Python List = 1
[*] Allocated = 2
Element 0: str
[*] Size of the Python List = 7
[*] Allocated = 7
Element 0: str
Element 1: int
Element 2: int
Element 3: float
Element 4: tuple
Element 5: list
Element 6: str
[*] Size of the Python List = 0
[*] Allocated = 0
[*] Size of the Python List = 1
[*] Allocated = 4
Element 0: int
[*] Size of the Python List = 5
[*] Allocated = 8
Element 0: int
Element 1: int
Element 2: int
Element 3: int
Element 4: int
[*] Size of the Python List = 4
[*] Allocated = 8
```

```
Element 0: int
Element 1: int
Element 2: int
Element 3: int
julien@CPython:~/CPython$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x03-python-data_structures`
- File: `100-print_python_list_info.c`

Done? | Help | Check your code | >_ Get a sandbox

Copyright © 2022 ALX, All rights reserved.