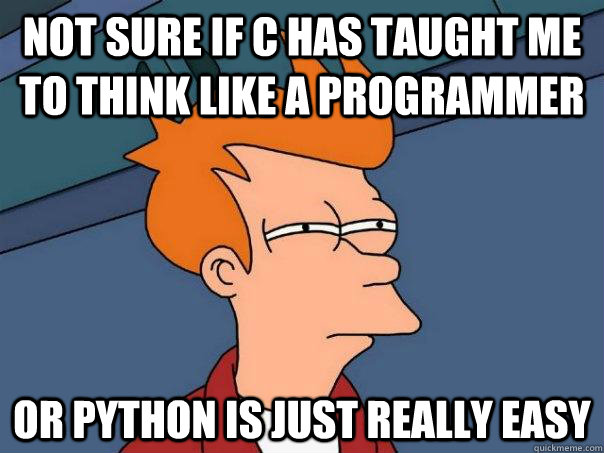
# 0x00. Python - Hello, World



## Author’s disclaimer

Welcome to the Python world!  
  
The first projects are more "C-oriented" - no tricks, no funky syntax - simple!  
If you've already played with Python, don't worry, fun things will come.  
You'll soon find that with Python (and the majority of higher level languages), there are ten different ways to do the same thing. Some tasks will expect only one implementation, while other tasks will have multiple possible implementations.  
Like C, Python also has a linter / style guide like Betty, called PEP8, also now known as PyCode.  
  
Enjoy!  
  
- Guillaume

## Resources

**Read or watch**:

* [The Python tutorial](https://intranet.alxswe.com/rltoken/JsFCs_NBzMAR7-XPAZ9BoA) (*only the first three chapters below*)
* [Whetting Your Appetite](https://intranet.alxswe.com/rltoken/kifRlLG2iMX5AZiW8lrCMg)
* [Using the Python Interpreter](https://intranet.alxswe.com/rltoken/RVpfAuagCo9SdfYeoHd6jg)
* [An Informal Introduction to Python](https://intranet.alxswe.com/rltoken/bVps0ZPWq7qVZ7vc-eJGTw) (*Read up until “3.1.2. Strings” included*)
* [How To Use String Formatters in Python 3](https://intranet.alxswe.com/rltoken/Ju0J8BxkuPX5yKZctyKfsQ)
* [Learn to Program](https://intranet.alxswe.com/rltoken/szBsJ-Qyig_RrImN7RGlOg)
* [Pycodestyle – Style Guide for Python Code](https://intranet.alxswe.com/rltoken/tgYt-0zVy1T4sDlE9ohxnA)

## Learning Objectives

At the end of this project, you are expected to be able to [explain to anyone](https://intranet.alxswe.com/rltoken/TYWTMEj3W1HhTHqMKu8kWA), **without the help of Google**:

### General

* Why Python programming is awesome
* Who created Python
* Who is Guido van Rossum
* Where does the name ‘Python’ come from
* What is the Zen of Python
* How to use the Python interpreter
* How to print text and variables using print
* How to use strings
* What are indexing and slicing in Python
* What is the official Python coding style and how to check your code with pycodestyle

### 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 repo, containing a description of the repository
* A README.md file, at the root of the folder of *this* 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

### Shell Scripts

* Allowed editors: vi, vim, emacs
* All your scripts will be tested on Ubuntu 20.04 LTS
* All your scripts should be exactly two lines long (wc -l file should print 2)
* All your files should end with a new line
* The first line of all your files should be exactly #!/bin/bash
* All your files must be executable

### C Scripts

* Allowed editors: vi, vim, emacs
* All your files will be compiled on Ubuntu 20.04 LTS using gcc, using the options -Wall -Werror -Wextra -pedantic -std=gnu89
* 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/alx-tools/Betty/blob/master/betty-style.pl) and [betty-doc.pl](https://github.com/alx-tools/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

## More Info

### Zen

The Zen of Python, by Tim Peters  
  
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than \*right\* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!

### Pycodestyle

Pycodestyle is now the [new standard of Python style code](https://intranet.alxswe.com/rltoken/UQ25jC6sA5XqZl6ZZIdAaw)



### Quiz questions

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

## Tasks

### 0. Run Python file

**mandatory**

Write a Shell script that runs a Python script.

The Python file name will be saved in the environment variable $PYFILE

guillaume@ubuntu:~/py/0x00$ cat main.py   
#!/usr/bin/python3  
print("Best School")  
  
guillaume@ubuntu:~/py/0x00$ export PYFILE=main.py  
guillaume@ubuntu:~/py/0x00$ ./0-run  
Best School  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 0-run

Done? Help Get a sandbox

### 1. Run inline

**mandatory**

Write a Shell script that runs Python code.

The Python code will be saved in the environment variable $PYCODE

guillaume@ubuntu:~/py/0x00$ export PYCODE='print(f"Best School: {88+10}")'  
guillaume@ubuntu:~/py/0x00$ ./1-run\_inline   
Best School: 98  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 1-run\_inline

Done? Help Get a sandbox

### 2. Hello, print

**mandatory**

Write a Python script that prints exactly "Programming is like building a multilingual puzzle, followed by a new line.

* Use the function print

guillaume@ubuntu:~/py/0x00$ ./2-print.py   
"Programming is like building a multilingual puzzle  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 2-print.py

Done? Help Get a sandbox

### 3. Print integer

**mandatory**

Complete this [source code](https://github.com/alx-tools/0x00.py/blob/master/3-print_number.py) in order to print the integer stored in the variable number, followed by Battery street, followed by a new line.

* You can find the source code [here](https://github.com/alx-tools/0x00.py/blob/master/3-print_number.py)
* The output of the script should be:
* the number, followed by Battery street,
* followed by a new line
* You are not allowed to cast the variable number into a string
* Your code must be 3 lines long
* You have to use f-strings [tips](https://intranet.alxswe.com/rltoken/Ju0J8BxkuPX5yKZctyKfsQ)

guillaume@ubuntu:~/py/0x00$ ./3-print\_number.py  
98 Battery street  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 3-print\_number.py

Done? Help Get a sandbox

### 4. Print float

**mandatory**

Complete the source code in order to print the float stored in the variable number with a precision of 2 digits.

* You can find the source code [here](https://github.com/alx-tools/0x00.py/blob/master/4-print_float.py)
* The output of the program should be:
* Float:, followed by the float with only 2 digits
* followed by a new line
* You are not allowed to cast number to string
* You have to use f-strings

guillaume@ubuntu:~/py/0x00$ ./4-print\_float.py  
Float: 3.14  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 4-print\_float.py

Done? Help Get a sandbox

### 5. Print string

**mandatory**

Complete this [source code](https://github.com/alx-tools/0x00.py/blob/master/5-print_string.py) in order to print 3 times a string stored in the variable str, followed by its first 9 characters.

* You can find the source code [here](https://github.com/alx-tools/0x00.py/blob/master/5-print_string.py)
* The output of the program should be:
* 3 times the value of str
* followed by a new line
* followed by the 9 first characters of str
* followed by a new line
* You are not allowed to use any loops or conditional statement
* Your program should be maximum 5 lines long

guillaume@ubuntu:~/py/0x00$ ./5-print\_string.py   
Holberton SchoolHolberton SchoolHolberton School  
Holberton  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 5-print\_string.py

Done? Help Get a sandbox

### 6. Play with strings

**mandatory**

Complete this [source code](https://github.com/alx-tools/0x00.py/blob/master/6-concat.py) to print Welcome to Holberton School!

* You can find the source code [here](https://github.com/alx-tools/0x00.py/blob/master/6-concat.py)
* You are not allowed to use any loops or conditional statements.
* You have to use the variables str1 and str2 in your new line of code
* Your program should be exactly 5 lines long

guillaume@ubuntu:~/py/0x00$ ./6-concat.py  
Welcome to Holberton School!  
guillaume@ubuntu:~/py/0x00$ wc -l 6-concat.py  
5 6-concat.py  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 6-concat.py

Done? Help Get a sandbox

### 7. Copy - Cut - Paste

**mandatory**

Complete this [source code](https://github.com/alx-tools/0x00.py/blob/master/7-edges.py)

* You can find the source code [here](https://github.com/alx-tools/0x00.py/blob/master/7-edges.py)
* You are not allowed to use any loops or conditional statements
* Your program should be exactly 8 lines long
* word\_first\_3 should contain the first 3 letters of the variable word
* word\_last\_2 should contain the last 2 letters of the variable word
* middle\_word should contain the value of the variable word without the first and last letters

guillaume@ubuntu:~/py/0x00$ ./7-edges.py  
First 3 letters: Hol  
Last 2 letters: on  
Middle word: olberto  
guillaume@ubuntu:~/py/0x00$ wc -l 7-edges.py  
8 7-edges.py  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 7-edges.py

Done? Help Get a sandbox

### 8. Create a new sentence

**mandatory**

Complete this [source code](https://github.com/alx-tools/0x00.py/blob/master/8-concat_edges.py) to print object-oriented programming with Python, followed by a new line.

* You can find the source code [here](https://github.com/alx-tools/0x00.py/blob/master/8-concat_edges.py)
* You are not allowed to use any loops or conditional statements
* Your program should be exactly 5 lines long
* You are not allowed to create new variables
* You are not allowed to use string literals

guillaume@ubuntu:~/py/0x00$ ./8-concat\_edges.py  
object-oriented programming with Python  
guillaume@ubuntu:~/py/0x00$ wc -l 8-concat\_edges.py  
5 8-concat\_edges.py  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 8-concat\_edges.py

Done? Help Get a sandbox

### 9. Easter Egg

**mandatory**

Write a Python script that prints “The Zen of Python”, by TimPeters, followed by a new line.

* Your script should be maximum 98 characters long (please check with wc -m 9-easter\_egg.py)

guillaume@ubuntu:~/py/0x00$ ./9-easter\_egg.py  
The Zen of Python, by Tim Peters  
  
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than \*right\* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!  
guillaume@ubuntu:~/py/0x00$

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 9-easter\_egg.py

Done? Help Get a sandbox

### 10. Linked list cycle

**mandatory**

**Technical interview preparation**:

* You are not allowed to google anything
* Whiteboard first
* This task and all future technical interview prep tasks will include checks for the efficiency of your solution, i.e. is your solution’s runtime fast enough, does your solution require extra memory usage / mallocs, etc.

Write a function in C that checks if a singly linked list has a cycle in it.

* Prototype: int check\_cycle(listint\_t \*list);
* Return: 0 if there is no cycle, 1 if there is a cycle

Requirements:

* Only these functions are allowed: write, printf, putchar, puts, malloc, free

carrie@ubuntu:~/0x00$ cat lists.h  
#ifndef LISTS\_H  
#define LISTS\_H  
  
#include <stdlib.h>  
  
/\*\*  
 \* struct listint\_s - singly linked list  
 \* @n: integer  
 \* @next: points to the next node  
 \*  
 \* Description: singly linked list node structure  
 \*   
 \*/  
typedef struct listint\_s  
{  
 int n;  
 struct listint\_s \*next;  
} listint\_t;  
  
size\_t print\_listint(const listint\_t \*h);  
listint\_t \*add\_nodeint(listint\_t \*\*head, const int n);  
void free\_listint(listint\_t \*head);  
int check\_cycle(listint\_t \*list);  
  
#endif /\* LISTS\_H \*/  
carrie@ubuntu:~/0x00$ cat 10-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 - adds a new node at the beginning of a listint\_t list  
 \* @head: pointer to a pointer of the start of the list  
 \* @n: integer to be included in node  
 \* Return: address of the new element or NULL if it fails  
 \*/  
listint\_t \*add\_nodeint(listint\_t \*\*head, const int n)  
{  
 listint\_t \*new;  
  
 new = malloc(sizeof(listint\_t));  
 if (new == NULL)  
 return (NULL);  
  
 new->n = n;  
 new->next = \*head;  
 \*head = 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:~/0x00$ cat 10-main.c  
#include <stdlib.h>  
#include <string.h>  
#include <stdio.h>  
#include "lists.h"  
  
/\*\*  
 \* main - check the code  
 \*  
 \* Return: Always 0.  
 \*/  
int main(void)  
{  
 listint\_t \*head;  
 listint\_t \*current;  
 listint\_t \*temp;  
 int i;  
  
 head = NULL;  
 add\_nodeint(&head, 0);  
 add\_nodeint(&head, 1);  
 add\_nodeint(&head, 2);  
 add\_nodeint(&head, 3);  
 add\_nodeint(&head, 4);  
 add\_nodeint(&head, 98);  
 add\_nodeint(&head, 402);  
 add\_nodeint(&head, 1024);  
 print\_listint(head);  
  
 if (check\_cycle(head) == 0)  
 printf("Linked list has no cycle\n");  
 else if (check\_cycle(head) == 1)  
 printf("Linked list has a cycle\n");  
  
 current = head;  
 for (i = 0; i < 4; i++)  
 current = current->next;  
 temp = current->next;  
 current->next = head;  
  
 if (check\_cycle(head) == 0)  
 printf("Linked list has no cycle\n");  
 else if (check\_cycle(head) == 1)  
 printf("Linked list has a cycle\n");  
  
 current = head;  
 for (i = 0; i < 4; i++)  
 current = current->next;  
 current->next = temp;  
  
 free\_listint(head);  
  
 return (0);  
}  
carrie@ubuntu:~/0x00$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 10-main.c 10-check\_cycle.c 10-linked\_lists.c -o cycle  
carrie@ubuntu:~/0x00$$ ./cycle   
1024  
402  
98  
4  
3  
2  
1  
0  
Linked list has no cycle  
Linked list has a cycle  
carrie@ubuntu:~/0x00$

Solving a problem is already a big win! but finding the best and optimal way to solve it, it’s way better! Think about the most optimal / fastest way to do it.

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x00-python-hello\_world
* File: 10-check\_cycle.c, lists.h

Done? Help Get a sandbox

Copyright © 2023 ALX, All rights reserved.