Bootcamp Python



Module00 Basic stuff

Module00 - Basic stuff - Eleven Commandments

The goal of the module is to get started with the Python language.

Notions of the module

Basic setup, variables, types, functions, ...

General rules

- The version of Python to use is 3.7, you can check the version of Python with the following command: python -V
- The norm: during this bootcamp you will follow the PEP 8 standards. You can install pycodestyle which is a tool to check your Python code.
- The function eval is never allowed.
- The exercises are ordered from the easiest to the hardest.
- Your exercises are going to be evaluated by someone else, so make sure that your variable names and function names are appropriate and civil.
- Your manual is the internet.
- You can also ask questions in the dedicated channel in the 42 AI Slack: 42-ai.slack.com.
- If you find any issue or mistake in the subject please create an issue on our dedicated repository on Github.

Helper

How do you install and link Python in the \$PATH? That's the first exercise!

Ensure that you have the right Python version.

```
> which python
/goinfre/miniconda/bin/python
> python -V
Python 3.7.*
> which pip
/goinfre/miniconda/bin/pip
```

Exercise 00 - \$PATH

Exercise 01 - Rev Alpha

Exercise 02 - The Odd, the Even and the Zero

Exercise 03 - Functional file

Exercise 04 - Elementary

Exercise 05 - The right format

Exercise 06 - A recipe

Exercise 07 - Shorter, faster, pythonest

Exercise 08 - S.O.S

Exercise 09 - Secret number

Exercise 10 - Loading bar!

\clearpage

Exercise 00 - \$PATH

Turn-in directory: ex00/

Files to turn in: answers.txt, requirements.txt

Forbidden functions: None Remarks: n/a

The first thing you need to do is install Python.

Conda manual install

If you want a fully automated install go to **Conda automated install** section. The automated part will allow you to reinstall everything more easily in case you use another computer. Below is a step by step installation.

1. Download conda install with the following command (MacOS version):

curl -LO "https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh"

2. Install conda using the script (we advise you to install it with this path /goinfre/miniconda3).

sh Miniconda3-latest-MacOSX-x86_64.sh -b -p <path>

The goinfre will change depending on your desktop location in cluster, so you will need to reinstall everything.

3. Add export to your .zshrc file.

export PATH=\$MINICONDA_PATH:\$PATH

4. Source your .zshrc file.

source ~/.zshrc

5. Check your Python environment.

which python

6. Install needed requirements.

```
conda install -y "jupyter" "numpy" "pandas"
```

Your Python should now be the one corresponding to the miniconda environment!

Conda automated install

A way to install the entire environment is to define a bash function in your ~/.zshrc.

1. Copy paste the following code into your ~/.zshrc.

```
function set_conda {
   HOME=$(echo ~)
   INSTALL_PATH="/goinfre"
   MINICONDA_PATH=$INSTALL_PATH"/miniconda3/bin"
   PYTHON_PATH=$(which python)
   SCRIPT="Miniconda3-latest-MacOSX-x86_64.sh"
   REQUIREMENTS="jupyter numpy pandas"
   DL_LINK="https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh"
   if echo $PYTHON_PATH | grep -q $INSTALL_PATH; then
       echo "good python version :)"
   else
   cd
   if [ ! -f $SCRIPT ]; then
       curl -LO $DL LINK
       fi
       if [ ! -d $MINICONDA_PATH ]; then
           sh $SCRIPT -b -p $INSTALL_PATH"/miniconda3"
   fi
   conda install -y $(echo $REQUIREMENTS)
   clear
   echo "Which python:"
   which python
   if grep -q "^export PATH=$MINICONDA_PATH" ~/.zshrc
   then
       echo "export already in .zshrc";
   else
       echo "adding export to .zshrc ...";
       echo "export PATH=$MINICONDA_PATH:\$PATH" >> ~/.zshrc
   fi
   source ~/.zshrc
   fi
```

By default, conda will be installed in the goinfre (look at the INSTALL_PATH variable). Feel free to change that path if you want to.

The function can be used whenever we want and will carry out the installation of miniconda and all needed librairies for the module (feel free to add more of them). It will also add a line to export miniconda environment.

2. Source your .zshrc with the following command:

```
source ~/.zshrc
```

3. Use the function set_conda:

```
set_conda
```

When the installation is done rerun the set_conda function.

4. Check your Python path.

```
which python
```

Your Python should now be the one corresponding to the miniconda environment!

Getting started

As an introduction, complete the following questionnaire using Python and pip, save your answers in a file answers.txt (write an answer per line in the text file), and check them with your peers.

Find the commands to:

- 1. Output a list of installed packages.
- 2. Output a list of installed packages and their versions.
- 3. Show the package metadata of numpy.
- 4. Search for PyPI packages whose name or summary contains "tesseract".
- 5. Freeze the packages and their current versions in a requirements.txt file you have to turn-in.

\clearpage

Exercise 01 - Rev Alpha

Turn-in directory: ex01/Files to turn in: exec.py Forbidden functions: None Remarks: n/a

Instructions:

You have to make a program that reverses the order of a string and the case of its words. If we have more than one argument we have to merge them into a single string and separate each arg by a '' (space char).

Examples:

```
> python exec.py "Hello World\!" | cat -e
!DLROw OLLEh$
> python exec.py "Hello" "my Friend" | cat -e
DNEIRf YM OLLEh$
> python exec.py
>
```

\clearpage

Exercise 02 - The Odd, the Even and the Zero

Turn-in directory: ex02/ Files to turn in: whois.py Forbidden functions: None Remarks: n/a

Instructions:

You have to make a program that checks if a number is odd, even or zero. The program will accept only one parameter, an integer.

Examples:

```
> python whois.py 12
I'm Even.

> python whois.py 3
I'm Odd.

> python whois.py

> python whois.py 0
I'm Zero.

> python whois.py Hello
ERROR

> python whois.py 12 3
ERROR
```

\clearpage

Exercise 03 - Functional file

Turn-in directory: ex03/Files to turn in: count.py Forbidden functions: None Remarks: n/a

Instructions:

Create a function called text_analyzer that displays the sums of upper-case characters, lower-case characters, punctuation characters and spaces in a given text.

text_analyzer will take only one parameter: the text to analyze. You have to handle the case where the text is empty (maybe by setting a default value). If there is no text passed to the function, the user is prompted to give one.

Test it in the Python console.

Examples:

```
> python
>>> from count import text_analyzer
>>> text_analyzer("Python 2.0, released 2000, introduced
features like List comprehensions and a garbage collection
```

```
system capable of collecting reference cycles.")
The text contains 143 characters:
- 2 upper letters
- 113 lower letters
- 4 punctuation marks
- 18 spaces
>>> text_analyzer("Python is an interpreted, high-level,
general-purpose programming language. Created by Guido van
Rossum and first released in 1991, Python's design philosophy
emphasizes code readability with its notable use of significant
whitespace.")
The text contains 234 characters:
- 5 upper letters
- 187 lower letters
- 8 punctuation marks
- 30 spaces
>>> text_analyzer()
What is the text to analyse?
>> Python is an interpreted, high-level, general-purpose
programming language. Created by Guido van Rossum and first
released in 1991, Python's design philosophy emphasizes code
readability with its notable use of significant whitespace.
The text contains 234 characters:
- 5 upper letters
- 187 lower letters
- 8 punctuation marks
 30 spaces
```

Handle the case when more than one parameter is given to text_analyzer:

```
>>> from count import text_analyzer
>>> text_analyzer("Python", "2.0")
ERROR
```

You're free to write your docstring and format it the way you want.

```
>>> print(text_analyzer.__doc__)

This function counts the number of upper characters, lower characters, punctuation and spaces in a given text.
```

\clearpage

Exercise 04 - Elementary

Turn-in directory: ex04/
Files to turn in: operations.py
Forbidden functions: None
Remarks: n/a

Instructions:

You have to make a program that prints the results of the four elementary mathematical operations of arithmetic (addition, subtraction, multiplication, division) and the modulo operation. This should be accomplished by

writing a function that takes 2 numbers as parameters and returns 5 values, as formatted in the console output below.

Examples:

```
> python operations.py 10 3
Sum:
            13
Difference: 7
           30
Product:
Quotient: 3.3333333333333333
Remainder:
> python operations.py 42 10
Difference: 32
Product:
           420
Quotient:
            4.2
Remainder:
> python operations.py 1 0
Sum:
Difference: 1
Product:
Quotient: ERROR (div by zero)
Remainder: ERROR (modulo by zero)
> python operations.py
Usage: python operations.py <number1> <number2>
Example:
   python operations.py 10 3
> python operations.py 12 10 5
InputError: too many arguments
Usage: python operations.py <number1> <number2>
Example:
    python operations.py 10 3
> python operations.py "one" "two"
InputError: only numbers
Usage: python operations.py <number1> <number2>
Example:
    python operations.py 10 3
> python operations.py "512" "63.1"
InputError: only numbers
Usage: python operations.py <number1> <number2>
Example:
    python operations.py 10 3
```

\clearpage

Exercise 05 - The right format

```
Turn-in directory: ex05/
Files to turn in: kata00.py, kata01.py, kata02.py, kata03.py, kata04.py
Forbidden functions: None
Remarks: n/a
```

Instructions:

Let's get familiar with the useful concept of **string formatting** through a kata series.

kata00

```
t = (19,42,21)
```

Including the tuple above in your file, write a program that dynamically builds up a formatted string like the following:

```
> python kata00.py
The 3 numbers are: 19, 42, 21
```

kata01

```
languages = {
    'Python': 'Guido van Rossum',
    'Ruby': 'Yukihiro Matsumoto',
    'PHP': 'Rasmus Lerdorf',
}
```

Using the languages dictionary above, write a program similar to the previous kata:

```
> python kata01.py
Python was created by Guido van Rossum
Ruby was created by Yukihiro Matsumoto
PHP was created by Rasmus Lerdorf
```

kata02

```
(3,30,2019,9,25)
```

Given the tuple above, whose values stand for: (hour, minutes, year, month, day), write a program that displays it in the following format:

```
> python kata02.py
09/25/2019 03:30
```

kata03

```
phrase = "The right format"
```

Write a program to display the string above right-aligned with '-' padding and a total length of 42 characters:

```
> python kata03.py | cat -e
-----The right format%
> python kata03.py | wc -c
42
```

kata04

```
(0, 4, 132.42222, 10000, 12345.67)
```

Given the tuple above, write a program which displays a formatted string:

```
> python kata04.py
module_00, ex_04 : 132.42, 1.00e+04, 1.23e+04
```

\clearpage

Exercise 06 - A recipe

Turn-in directory: ex06/Files to turn in: recipe.py Forbidden functions: None Remarks: n/a

Objective:

It is time to discover Python dictionaries. Dictionaries are collections that contain mappings of unique keys to values.

Hints: check what is a nested dictionary in Python.

Instructions:

First, you have to create a cookbook dictionary called cookbook.

cookbook will store 3 recipes:

- sandwich
- cake
- salad

Each recipe will store 3 values:

- ingredients: a list of ingredients
- meal: type of meal
- prep time: preparation time in minutes

Sandwich's ingredients are ham, bread, cheese and tomatoes. It is a lunch and it takes 10 minutes of preparation. Cake's ingredients are flour, sugar and eggs. It is a dessert and it takes 60 minutes of preparation. Salad's ingredients are avocado, arugula, tomatoes and spinach. It is a lunch and it takes 15 minutes of preparation.

- 1. Get to know dictionaries. In the first place, try to print only the keys of the dictionary. Then only the values. And to conclude, all the items.
- 2. Write a function to print a recipe from cookbook dictionary. The function parameter will be the name of the recipe.
- 3. Write a function to delete a recipe from cookbook dictionary. The function parameter will be the name of the recipe.
- 4. Write a function to add a new recipe to cookbook with its ingredients, its meal type and its preparation time. The function parameters will be the name of recipe, the ingredients, the meal type and preparation time.

- 5. Write a function to print all recipe names from cookbook dictionary. Think about formatting the output.
- 6. Last but not least, make a program using the four functions you just created.

The program will prompt the user to make a choice between printing the cookbook, printing only one recipe, adding a recipe, deleting a recipe or quitting the cookbook.

It could look like the example below but feel free to organize it the way you want to:

```
> python recipe.py
Please select an option by typing the corresponding number:
1: Add a recipe
2: Delete a recipe
3: Print a recipe
4: Print the cookbook
5: Quit
>> 3
Please enter the recipe's name to get its details:
>> cake
Recipe for cake:
Ingredients list: ['flour', 'sugar', 'eggs']
To be eaten for dessert.
Takes 60 minutes of cooking.
```

Your program must continue running until the user exits it (option 5):

```
> python recipe.py
Please select an option by typing the corresponding number:
1: Add a recipe
2: Delete a recipe
3: Print a recipe
4: Print the cookbook
5: Quit
>> 5
Cookbook closed.
```

The program will also continue running if the user enters a wrong value. It will prompt the user again until the value is correct:

```
> python recipe.py
Please select an option by typing the corresponding number:
1: Add a recipe
2: Delete a recipe
3: Print a recipe
4: Print the cookbook
5: Quit
>> test
This option does not exist, please type the corresponding number.
To exit, enter 5.
>>
```

\clearpage

Exercise 07 - Shorter, faster, pythonest

Turn-in directory: ex07/

Files to turn in: filterwords.py

Forbidden functions: filter Remarks: n/a

Instructions:

Using list comprehensions, you have to make a program which removes all the words in a string that are shorter than or equal to n letters, and returns the filtered list with no punctuation.

The program will accept only two parameters: a string, and an integer n.

Examples:

\clearpage

Exercise 08 - S.O.S

 $\begin{array}{ccc} \text{Turn-in directory:} & \text{ex}08/\\ \text{Files to turn in:} & \text{sos.py} \\ \text{Forbidden functions:} & \text{None} \\ \text{Remarks:} & \text{n/a} \end{array}$

Instructions:

You have to make a function which encodes strings into Morse code. All alphanumeric characters are accepted by the encoder.

Examples:

@ref: https://morsecode.world/international/morse2.html

\clearpage

Exercise 09 - Secret number

 $\begin{array}{ccc} \text{Turn-in directory:} & \text{ex09/} \\ \text{Files to turn in:} & \text{guess.py} \\ \text{Forbidden functions:} & \text{None} \\ \text{Remarks:} & \text{n/a} \\ \end{array}$

Instructions:

You have to make a program that will be an interactive guessing game. It will ask the user to guess a number between 1 and 99. The program will tell the user if their input is too high or too low. The game ends when the user finds out the secret number or types exit.

You will import the random module with the randint function to get a random number. You have to count the number of trials and print that number when the user wins.

Examples:

```
> python guess.py
This is an interactive guessing game!
You have to enter a number between 1 and 99 to find out the secret number.
Type 'exit' to end the game.
Good luck!
What's your guess between 1 and 99?
>> 54
Too high!
What's your guess between 1 and 99?
Too low!
What's your guess between 1 and 99?
>> 45
Too high!
What's your guess between 1 and 99?
>> A
That's not a number.
What's your guess between 1 and 99?
Congratulations, you've got it!
You won in 5 attempts!
```

If the user discovers the secret number on the first try, tell them. If the secret number is 42, make a reference to Douglas Adams.

```
> python guess.py
This is an interactive guessing game!
You have to enter a number between 1 and 99 to find out the secret number.
Type 'exit' to end the game.
Good luck!

What's your guess between 1 and 99?
>> 42
The answer to the ultimate question of life, the universe and everything is 42.
Congratulations! You got it on your first try!
```

Other example:

```
> python guess.py
This is an interactive guessing game!
You have to enter a number between 1 and 99 to find out the secret number.
Type 'exit' to end the game.
Good luck!
What's your guess between 1 and 99?
>> exit
Goodbye!
```

\clearpage

Exercise 10 - Loading bar!

 $\begin{array}{ccc} \text{Turn-in directory:} & \text{ex}10/\\ \text{Files to turn in:} & \text{loading.py} \\ \text{Forbidden functions:} & \text{None} \\ \text{Remarks:} & \text{n/a} \\ \end{array}$

This is a bonus exercise! ## Objective:

You are about to discover the yield operator! So let's create a function called ft_progress(lst).

Instructions:

You have to create a function which displays the progress of a for loop.

Hints: Take a look to the yield statement.

Examples:

```
listy = range(1000)
ret = 0
for elem in ft_progress(listy):
    ret += (elem + 3) % 5
    sleep(0.01)
print()
print(ret)
```

```
> python loading.py
ETA: 8.67s [ 23%] [====> ] 233/1000 | elapsed time 2.33s
...
2000
```

```
listy = range(3333)
ret = 0
for elem in ft_progress(listy):
    ret += elem
    sleep(0.005)
print()
print(ret)
```

```
> python loading.py
ETA: 14.67s [ 9%][=> ] 327/3333 | elapsed time 1.33s
```

5552778		
\clearpage		