Context

Almost everyday in you day to day work you'll have internet access.
Today you won't have any. and you can't use your phone.
You'll be provided with the standard python documentation in pdf format.

You are not allowed to use an IDE (no vscode, no atom).
You'll have to use vi or nano.

Part 0

Read every exercise and question before starting. Then Read Again before starting any exercise.

Create a folder with your name. your whole work will be done in this folder.

```
~ $ mkdir lumy ; cd lumy
~/lumy $
```

Write your name (or nickname) with a period at the end in a file named `rendu.txt`.
ex:

```
$ cat rendu.txt
lumy.
```

Optional : Write another file named `why.txt`.
In one or two sentences, tell me what you expected from IT (job, money, personal project, domotique, new environemnt, working with google or working on IOT).

From now on each part of the test will be in its own folder:

```
$ ls
part1/ part2/ part3/ ...
$ ls part1/
part1/myfuncs.py
```

Part 1

For the first part **you are NOT allowed to use ANY builtins functions not explicitely listed below, or other module (no import). Calling methods of `str` class is not allowed.**

"Builtins" are functions already available in Python without any imports : like `map`, `zip`, `all`, `any`...

You are only allowed to use `len()`, `range()`, `ord()`, `chr()`.

Operators and keywords are **allowed** : `and`, `or`, `not`, `>`, `for`, `while` ...

`str` class methods are functions you can call on a string : for instance "abc".isalpha() is calling the "method" `isalpha` on the string "abc".

Refer to bonus question for handling of incorrect argument types or content.

In a file `myfuncs.py` write these 5 functions:

- isalpha:
  - Must return True if all characters are letters (uppercase or lowercase), , False otherwise
- isdigit: take a string instance as argument, check is it's made of digits
  - Returns True is all characters in the string are digits (between 0 and 9), False otherwise
- isalnum: take a string instance as argument, check if it's alphanumerical
  - Returns True if all characters are letters (upper or lowercase), or digits, False otherwise
- tolower : converts a string to lowercase
  - Takes a string as argument, returns the same string with all uppercase letters converted to lowercase (leaving the rest unchanged)
    - Example : tolower("AbCd1234#") => "abcd1234#"
- lencmp: compares length of two strings
  - take 2 string instances as arguments, return 0 if they are of equal length, 1 if the first one is longer, -1 if the first one is shorter
  - lencmp("a", "b") returns 0
  - lencmp("aa", "z") returns 1
  - lencmp("z", "aa") returns -1
- strcmp: take 2 string instances as arguments, return 0 if they are equal, 1 if the first one is greater (in lexicographical order), -1 if the first one is lower
  - This function is case-insensitive : strcmp("ABC", "abc") must return 0
  - "Lexicographical order" is the way words are sorted in a dictionary, also called "alphabetical order"
    - First compare each character in each string starting from the left
    - If a string has a character that is higher in the alphabetical order : this string is the greater one
    - If both strings have the same characters starting from the left : the longest string is the greater one
    - If both string contain the same characters in the same order : they're equal (obviously)
    - Examples :
      - "b" is greater than "abcd"
      - "abcd" is greater than "abc"
      - "z" is greater than "ywfrgeser"

Bonuses:

- raise a ValueError when the type of the arguments is not the expected one.
- Implement the function strlen: take a string instance as argument return the length of the string.
  - Without using len()
  - Must return an integer : either the length of the string or -1 if the argument isn't a str

function name	parameters	return value
isalpha	string	boolean
isdigit	string	boolean
isalnum	string	boolean

function name	parameters	return value
tolower	string	string
lencmp	string, string	int
strcmp	string, string	int

Part 2

You can now use builtins. **Still no import or method calls.**

For each of those functions create a python file in which you'll implement the funciton.
A table is provided with the signature of each function, use the "filename" from that table as your filename.
You must also name your function as indicated in the table.

- Three Words:
  - Take a string as parameter
  - Return True if a string contains a three consecutive space-separated "words" containing only letters
  - Examples :
    - 'abc abc dce 1 cde' => True
    - 'abc 1 bcd 1 abc' => False

- Reverse Each Word:
  - Takes as input a string of space separated "words"
  - 'hello world' => 'olleh dlrow'

- Acceptable Password:
  - Takes one string parameter
  - Returns True if the parameter contains all of the following :
    - includes both numbers and letters
      - contains at least a "special" character (that isn't a letter or number)
      - length is more than 8 characters

- Digits Multiplication:
  - Takes an int as input
  - Take each digit from if and multiply them together
  - If the input number is negative : result is negative
  - Ignore any zeroes
  - Must return a `str`
  - Examples :
    - 123045 = 1 * 2 *3 * 4 * 5 => '120'
      - -2035 = -2 * 3 * 5 => '-30'

- Syracuse problem :
  - Take a integer `N` greater than 1 as input
  - Then repeat the following operations until `N` equals 1 : if `N` is even : divide it by 2, else : multiply it by 3 and add 1
  - Return all the numbers reached during that process as a list (except the very first one)
  - Examples :
    - Start with 7 the sequence is : "22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1"
    - Start with 10 the sequence is : "5, 16, 8, 4, 2, 1"

        - Context : there"s an unproven mathematical theory, the "Collatz conjecture", that the Syracuse Sequence of any positive integer won't be infinite, but experimental data shows that any number your computes can handle won't have an infinite sequence

- Fizz Buzz:

    - Take an int as parameters.
    - return a string:
        - `Fizz Buzz` if the number is divisible by 3 and by 5
        - `Fizz` if the number is divisible by 3
        - `Buzz` if the number is divisible by 5
        - The number as a string for other cases.

  Bonus:
  None, bonus points will be counted if builtins are used wisely in code.

filename	function name	parameters	return value
three_word	three_word	string	boolean
reverse_word	reverse_word	string	string
valid_password	valid_password	string	boolean
multiply_digit	multiply_digit	int	string
syracuse	syracuse	integer	list
fizzbuzz	fizzbuzz	integer	string

Part 3

You can import and use methods calls as well.

Write this function(s) in file named `cpwd.py`

- check_password using SHA256 algorithm:
    - take two string as parameters.
    - the first one is a password in clear
    - the second is a hash (password that we protected)
    - returns True is the password matches the hash, False otherwise
    - example:
        - check_password(b"test", "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08") => True
        - check_password(b"test", "915b0f00a08") => False

Hint:

hashlib can be very usefull.

- Bonus: Take a third parameters, optional with default value of `sha256` but accept also `blake` and use the right algorithm in function of this string
    - examples:
        - check_password(b"test", "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08", "sha256") => True
        - check_password(b"test", "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08", "blake") => False

Part 4

- create an python script `mydb`
  - it must take command line arguments (from sys.argv)
    - first one is always `add` or `get`
    - second is always a key
    - third one (for add) is the value
  - create and update a file `mydb.db` which will contains your database.
  - In case of `add` the key/value should be recorded
  - In case of `get` the value corresponding to the key will be printed

```
$ mydb add favorite_teacher lumy
$ mydb add favorite_day monday
$ mydb get favorite_teacher
lumy
$ mydb add favorite_teacher romain
$ mydb get favorite_day
monday
$ mydb get favorite_teacher
romain
```

Bonus: crypt (any cryptographic method is ok, even rot13) content of value in my.db.

> ROT13 ("rotate by 13 places", sometimes hyphenated ROT-13) is a simple letter substitution cipher that replaces a letter with the 13th letter after it in the alphabet. ROT13 is a special case of the Caesar cipher which was developed in ancient Rome
> Example: hello => uryyb

Bonus part

Only start this part if you completed everything else.

- Create a quizz game with Tkinter
- Offer your user multiple choices for each question
- Your quizz game must take its input from any kind of structured text file you wish (.yml, .json, .xml, .csv…).
- This file will contain the questions, the possible answers, and the right ones
- Score your user in the end (number of good answers, number of wrong answers…)
- Bonus : imitate Kahoot and offer scoring based on how fast the user answers