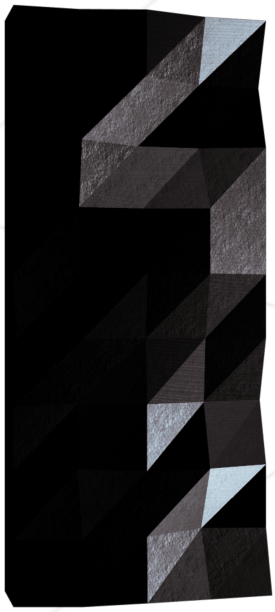

Pre-open Piscine

New Language 00

Summary: This document is the subject for the New Language 00 module of the Pre-open Piscine @ 42Tokyo.

Contents

Chapter I

Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for every exercise.
- Your exercises will be checked and graded by your fellow classmates.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Exercises in Shell must be executable with /bin/sh.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / man / the Internet / ...`.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- If no other explicit information is displayed, you must assume the following versions of languages : `Python - python3.8.2`, `Ruby - 2.3.7`

Chapter II

Foreword

There are 10 types of people in the world: those who understand binary, and those who don't.

Chapter III

Exercise 00 : Variables

	Exercise 00
Variables	
Turn-in directory : $ex00/$	
Files to turn in : `var.py or var.rb or both`	
Allowed functions : `None`	

- Create a file `var.py` or `var.rb` which you will put a function called `my_var`.

  You have to put all the possible type of variable inside the file and print out those values as below.

```
?>python3 var.py
42 is type of <class 'int'>
42 is type of <class 'str'>
quarante-deux is type of <class 'str'>
42.0 is type of <class 'float'>
True is type of <class 'bool'>
[42] is type of <class 'list'>
{42: 42} is type of <class 'dict'>
(42,) is type of <class 'tuple'>
set() is type of <class 'set'>
?>
```

```
?>ruby var.rb
42 is type of Fixnum
42 is type of String
is type of NilClass
42.0 is type of Float
[42, 42] is type of Array
{"42"=>42} is type of Hash
42..42 is type of Range
?>
```

  It is forbidden to copy paste this output and submit the file.

Chapter IV

Exercise 01 : Numbers

	Exercise 01
Numbers	
Turn-in directory : $ex01/$	
Files to turn in : `numbers.py or numbers.rb or both`	
Allowed functions : `None`	

- For this exercise, you can create function as much as you wish and you can name those function as you like.

  Download the `resources.tar.gz` which is on the main project page. open the tar and inside the directory `ex01`, you will see a file `numbers.txt` which contains numbers from 1 to 100 separeted with commas.

  Create a script which will open and read the content of the file and then print out those numbers, one number per line without commas.

Chapter V

Exercise 02 : My first dictionary/hash

	Exercise 02
My first dictionary/hash	
Turn-in directory : $ex02/$	
Files to turn in : `dict.rb or dict.py or both`	
Allowed functions : `None`	

For this exercise, you can create function as much as you wish and you can name those function as you like.

Create a script which you will need to copy paste this list/Array.

```
data = [['Caleb' , 24],
['Calixte' , 84],
['Calliste', 65],
['Calvin' , 12],
['Cameron' , 54],
['Camil' , 32],
['Camille' , 5],
['Can' , 52],
['Caner' , 56],
['Cantin' , 4],
['Carl' , 1],
['Carlito' , 23],
['Carlo' , 19],
['Carlos' , 26],
['Carter' , 54],
['Casey' , 2]]
```

- Your script needs to transform that variable `d` into a dictionary/hash.
- Key is the number, and other one will be the value.
- Your script will also need to print out those keys and values as below.

```
?>ruby dict.rb
24 : Caleb
84 : Calixte
65 : Calliste
12 : Calvin
[...]
```

```
?>python3 dict.py
24 : Caleb
84 : Calixte
65 : Calliste
12 : Calvin
[...]
```

Chapter VI

Exercise 03 : Find by Key

	Exercise 03
Find by Key	
Turn-in directory : $ex03/$	
Files to turn in : `capital_city.py or capital_city.rb or both`	
Allowed functions : `None`	

Copy paste the dictionary below inside your script.

```
states = {
    "Oregon" : "OR",
    "Alabama" : "AL",
    "New Jersey": "NJ",
    "Colorado" : "CO"
    }
    capital_cities = {
        "OR": "Salem",
        "AL": "Montgomery",
        "NJ": "Trenton",
        "CO": "Denver"
        }
```

- Create a program which will take one argument State (ex: Oregon) and the program will output the name of the capital city(ex: Salem) related to that state.
- If there is no related capital city for a given argument, the program should print: `Unknown state`.
- If there is no or too many argument, your script wont do anything.

```
$> python3 capital_city.py Oregon
Salem
$> python3 capital_city.py Ile-De-France
Unknown state
$> python3 capital_city.py
$> python3 capital_city.py Oregon Alabama
$> python3 capital_city.py Oregon Alabama Ile-De-France
$>
```

```
$> ruby capital_city.rb Oregon
Salem
$> ruby capital_city.rb Ile-De-France
Unknown state
$> ruby capital_city.rb
$> ruby capital_city.rb Oregon Alabama
$> ruby capital_city.rb Oregon Alabama Ile-De-France
$>
```

Chapter VII

Exercise 04 : Find by Value

	Exercise 04
Find by Value	
Turn-in directory : $ex04/$	
Files to turn in : `state.py or state.rb or both`	
Allowed functions : `None`	

- Copy the previous dictionary inside your file once again.
- Create a program which will take one argument capital city(ex: Salem) and the program will output the name of the State (ex:Oregon) related to that capital city.
- If there is no related state for a given argument, the program should print: `Unknown capital city`.
- If there is no or too many argument, your script wont do anything.

```
$> python3 state.py Salem
Oregon
$> python3 state.py Paris
Unknown capital city
$> python3 state.py
$>
```

```
$> ruby state.rb Salem
Oregon
$> ruby state.rb Paris
Unknown capital city
$> ruby state.rb
$>
```

Chapter VIII

Exercise 05 : Find by value or key

	Exercise 05
Find by value or key	
Turn-in directory : $ex05/$	
Files to turn in : `all_in.py or all_in.rb or both`	
Allowed functions : `None`	

- Copy the previous dictionary inside your file once again.
- Your program should accept one argument which will contain as much expression as they want separeted with commas.
- For each expression, your program have to detect if it is a capital city, state or neither of them.
- Your program is not case sensitive and your program accept multiple whitespaces.
- If there is no argument or if there is too many argument, your program doesn't print anything out.
- If there is two consecutive commas, your program doesn't print anything out.
- If there is two consecutive commas, your program doesn't print anything out.

```
$> python3 all_in.py "New jersey, Tren ton, NewJersey, Trenton, toto, , sAlem"
Trenton is the capital of New Jersey
Tren ton is neither a capital city nor a state
NewJersey is neither a capital city nor a state
Trenton is the capital of New Jersey
toto is neither a capital city nor a state
Salem is the capital of Oregon
$>
```

```
$> ruby all_in.rb "New jersey, Tren ton, NewJersey, Trenton, toto, , sAlem"
Trenton is the capital of New Jersey
Tren ton is neither a capital city nor a state
NewJersey is neither a capital city nor a state
Trenton is the capital of New Jersey
```

```
toto is neither a capital city nor a state
Salem is the capital of Oregon
$>
```

Chapter IX

Exercise 06 : Sort this dictionary/hash

	Exercise 06
Sort this dictionary/hash	
Turn-in directory : *ex*06/	
Files to turn in : `my_sort.py or my_sort.rb or both`	
Allowed functions : `None`	

Add this dictionary/hash inside your file.

```
d = {
    'Hendrix':'1942',
    'Allman':'1946',
    'King':'1925',
    'Clapton':'1945',
    'Johnson':'1911',
    'Berry':'1926',
    'Vaughan':'1954',
    'Cooder':'1947',
    'Page':'1944',
    'Richards':'1943',
    'Hammett':'1962',
    'Cobain':'1967',
    'Garcia':'1942',
    'Beck':'1944',
    'Santana':'1947',
    'Ramone':'1948',
    'White':'1975',
    'Frusciante':'1970',
    'Thompson':'1949',
    'Burton':'1939',
    }
```

- Create a program which will first sort this dictionary/hash by year and then sort it by alphabetical order if the year is same. You will print out the name of the artist line by line without printing out the year.

Chapter X

Exercise 07 : Peiodic Table

	Exercise 07
Peiodic Table	
Turn-in directory : $ex07/$	
Files to turn in : `periodic_table.py or periodic_table.rb or both`	
Allowed functions : `None`	

Download the `resources.tar.gz` which is on the main project page. open the tar and inside the directory `ex07`, you will see a file `periodic_table.txt` which contains necessary information of elements.

Create a program which will use this file and create a HTML page which describe about the elements inside periodic table.

- Each element should be inside a 'box' of HTML table
- The name of the element should be inside h4.
- Each Attributs of elements should be inside a list. It should contain at least, atomic number, symbol and the atomic weight.
- The periodic table should be displayed as the Mendeleev table which you can easily find by googling. Don't forget that there is some empty boxes and you should follow the same output !

Your program must create a file `periodic_table.html`. This file should be readable from any kind of browser. This file must pass the validation of `W3C`.

You are free to add inline css to make the visual looks better. If you don't like inline css, you can also create a file called `periodic_table.css`.

Below is the example of one element:

```
[...]
<table>
<tr>
<td style="border: 1px solid black; padding:10px">
<h4>Hydrogen</h4>
<ul>
<li>No 42</li>
<li>H</li>
```

```
<li>1.00794</li>
<li>1 electron</li>
<ul>
</td>
[...]
```

https://validator.w3.org/#validate_by_upload