Statistical Computing
Lesson 3

Gabriele Rovigatti

Bank of Italy

March 15, 2021

This Course

- Python basics
  - General Framework
  - Syntax
  - Modules / Libraries
  - Functions
- Webscraping
  - Data Storage
  - HTML Syntax
  - Data Collection
  - `Requests` Package
  - `Webdriver` Package
- Machine Learning?

Today

- Python basics
  - General Framework
  - Syntax
  - Modules / Libraries
  - Functions
- **Webscraping**
  - **Data Storage**
  - **HTML Syntax**
  - **Data Collection**
  - **`Requests` Package**
  - `Webdriver` Package
- Machine Learning?

Arithmetic and Comparison Operators

```
1  # numeric objects
2  a = 10 # 10
3  a += 1 # 11 --
4  a -= 1 # 10
5  a % 3 # 1 -- modulo operator:
6  a ** 2 # 100
7
8  # string objects
9  animals = 'Cat ' + 'Dog '
10 animals += 'Monkey' # 'Cat Dog Monkey'
11 animals_list = ['Cat', 'Dog', 'Monkey']
12 ' and '.join(animals_list) # 'Cat and Dog and Monkey'
13
14 # Arithmetic Comparison
15 a > 10 # False
16 a >= 10 # True
17 a == 10 # True
18 a != 10 # False
```

Comparison

```
1  # Logical
2  A and B
3  A or B
4  not A
5  (A and (B or C))
6
7  # Identity Comparison
8  1 is 1 # True
9  1 is not '1' # True
10 bool(1) # Boolean Logical
11 bool(True) # This works with nearly every object
12 True is bool(1) # True
```

Functions

Functions in Python are defined according to simple rules:

- The function block begins with **def** followed by `functname` and parameters within brackets
- The code block within every function starts with a colon (:) and is indented
- The statement `return(object)` exits a function, optionally passing back an object to the caller

Check whether input is string

```
def str_check(parlist): # here we define the function: def, name, list parameters
    array = list(isinstance(x, str) for x in parlist) #this is indented!
    return(array)

print(str_check(['this',2,'string'])) # call the function with a parameter list
```

Fibonacci Series: Define a Function

```
import sys # not necessary, just to print the error
### FUNCTION DEFINITION ###s
def fibonacci_series(series_length, double_ones=True):
    if series_length <= 2: # check whether n > 2. If it is not the case, throw an er
        raise ValueError('You must input n > 2!')
        sys.exit()
    # for loop solution
    if double_ones == True:
        fibonacci_list = [1, 1] # initiate the list with the predetermined elements
    else:
        fibonacci_list = [0,1]
    for x in range(2,series_length+1):
        fibonacci_list.append(fibonacci_list[x-1] + fibonacci_list[x-2])
        # append the sum of previous two values to the list
    return fibonacci_list # return the list as output of the function
### ROUTINE ###
n = int(input("Input a number greater than 2 ")) # read the input number
d1 = fibonacci_series(n)
print(d1) # print the resulting list
d2 = fibonacci_series(n,0)
print(d2)
```

Import Modules

```
1           # Imports the datetime module into the current namespace
2 import datetime
3 datetime.date.today()
4 # Import datetime and adds date and timedelta to the current namespace
5 from datetime import date, timedelta
6 date.today()
7 # Rename imports
8 import date as my_date
9 # This is sometimes used, but it is not suggested
10 from datetime import *
```

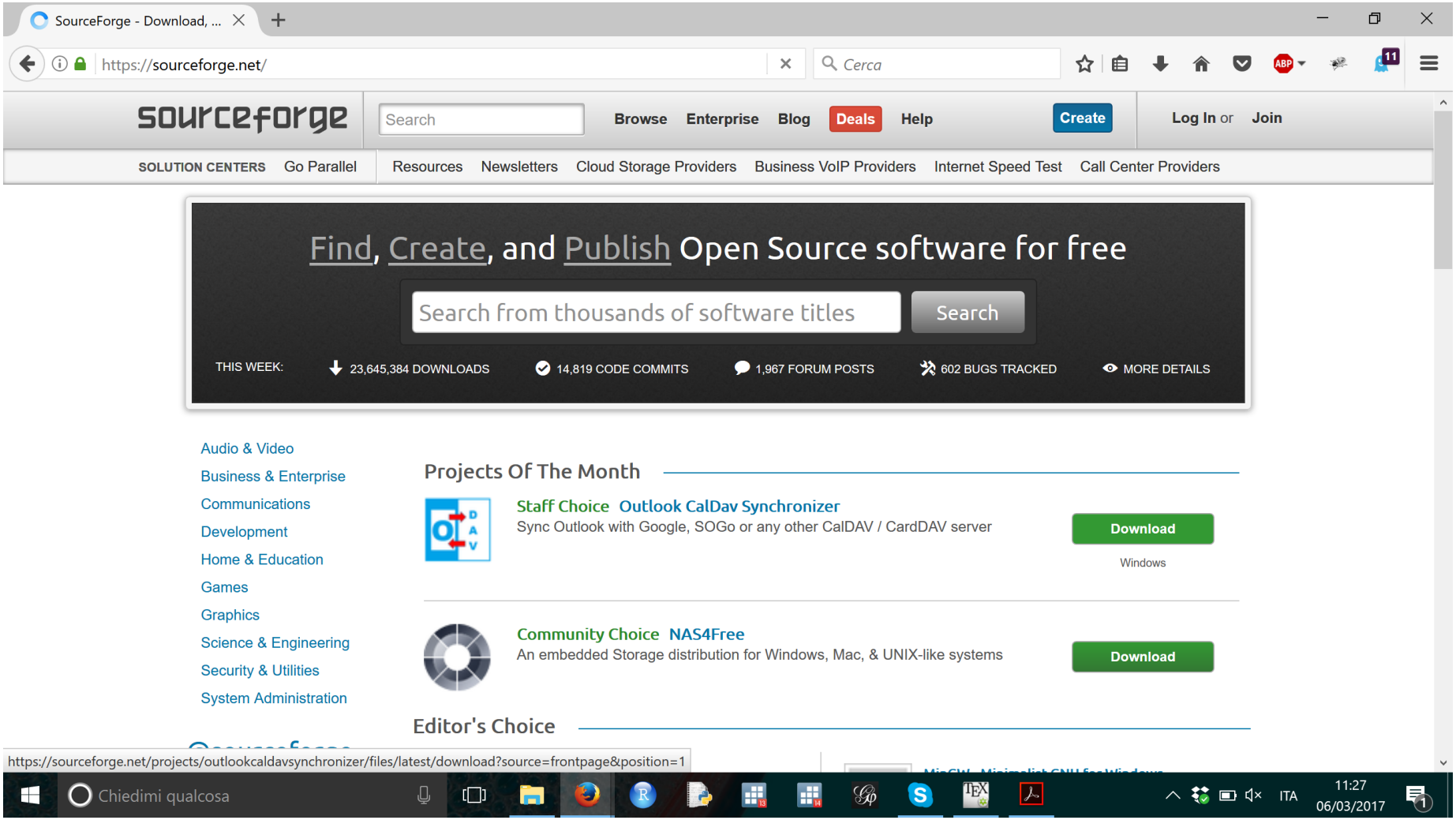
Data Storage

On the web, data are typically stored in:

- **Repositories (servers, GitHub)**
- FTP-type repositories
- Webpages

In turn, these can be public - i.e. freely accessible through urls - or not - e.g. domains accessible through username/password.

Github repository



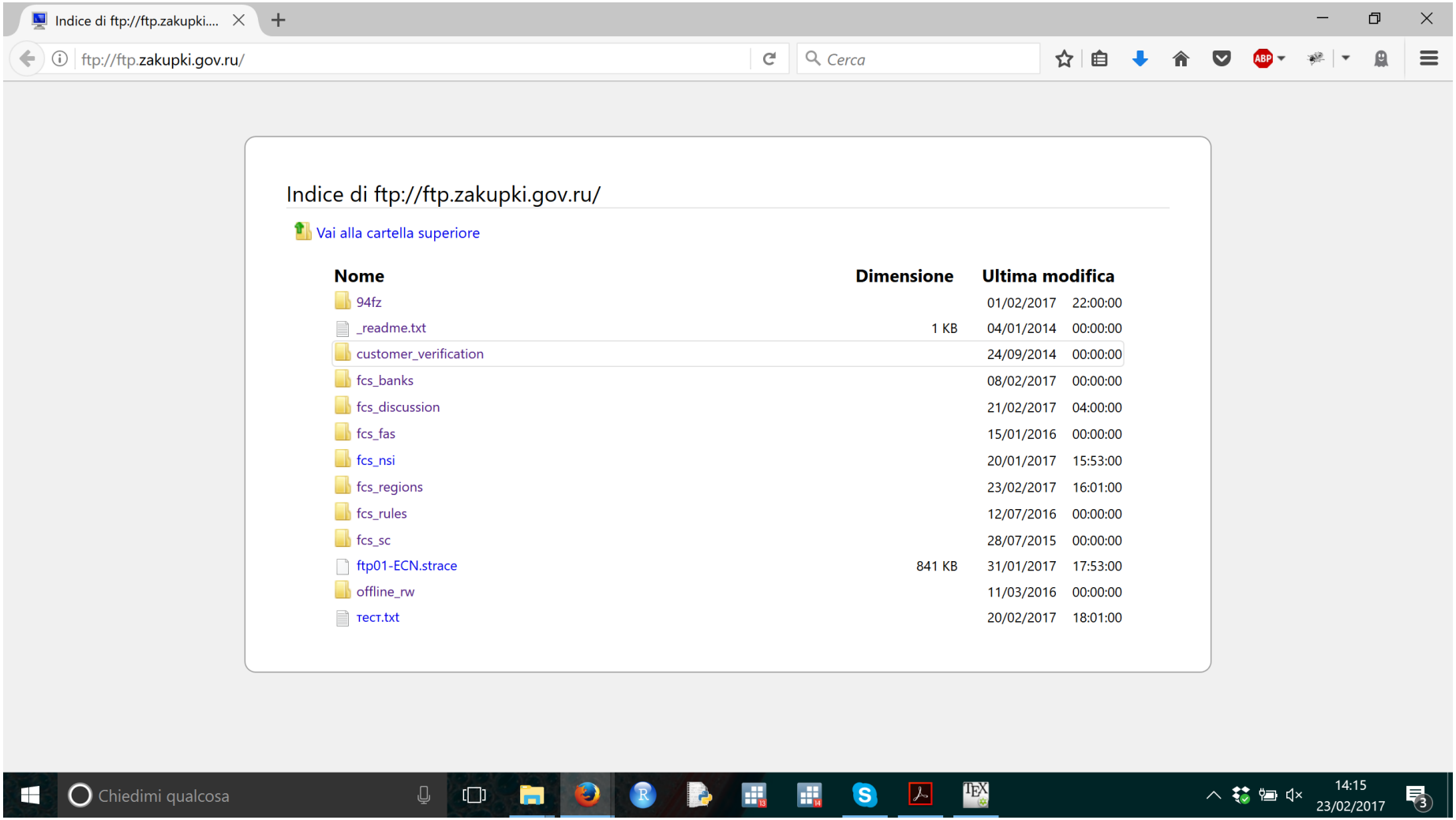
Data Storage

On the web, data are typically stored in:

- Repositories (servers, GitHub)
- **FTP-type repositories**
- Webpages

In turn, these can be public - i.e. freely accessible through urls - or not - e.g. domains accessible through username/password.

FTP server


Indice di ftp://ftp.zakupki....
ftp://ftp.zakupki.gov.ru/
Cerca
Indice di ftp://ftp.zakupki.gov.ru/
Vai alla cartella superiore
Nome
Dimensione
Ultima modifica
94fz 01/02/2017 22:00:00
_readme.txt 1 KB 04/01/2014 00:00:00
customer_verification 24/09/2014 00:00:00
fcs_banks 08/02/2017 00:00:00
fcs_discussion 21/02/2017 04:00:00
fcs_fas 15/01/2016 00:00:00
fcs_nsi 20/01/2017 15:53:00
fcs_regions 23/02/2017 16:01:00
fcs_rules 12/07/2016 00:00:00
fcs_sc 28/07/2015 00:00:00
ftp01-ECN.strace 841 KB 31/01/2017 17:53:00
offline_rw 11/03/2016 00:00:00
тест.txt 20/02/2017 18:01:00
Chiedimi qualcosa
14:15
23/02/2017

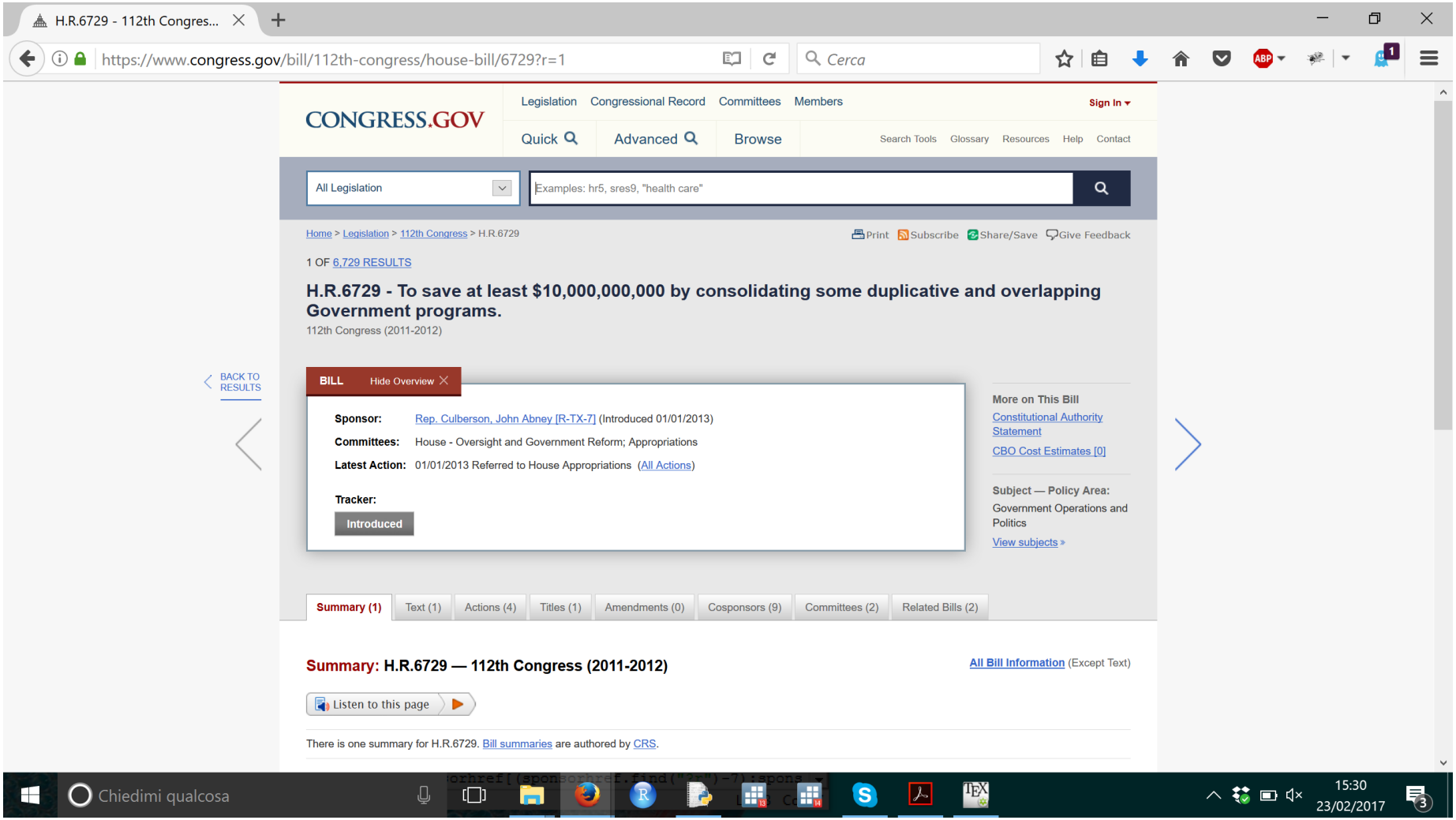
Data Storage

On the web, data are typically stored in:

- Repositories (servers, GitHub)
- FTP-type repositories
- **Webpages**

In turn, these can be public - i.e. freely accessible through urls - or not - e.g. domains accessible through username/password.

Website - text information



Extract data from the internet

There are several ways to extract data from websites:

- Download structured databases (*.csv*,*.xlsx*,*.xml*) export report
- API (Application Programming Interface) API
- Scrape webpages:
  - Tabular data tab data
  - Text text data

All these methods can be implemented by hand, or **automated through codes in Python**.

Exercises

- Write a Python script to concatenate following dictionaries to create a new one solution
  - dict1 = {1:10, 2:20}
  - dict2 = {3:30, 4:40}
  - dict3 = {5:50, 6:60}
- Write a Python script to generate and print a dictionary that contains the square of numbers between 1 and n (`n=int(input("Input a number "))` allows you to input integers). n = 10 solution
- Write a Python program to sum all the items in the previous dictionary solution
- Write a Python program to map two lists into a dictionary solution
  - colors = ['red', 'green', 'blue']
  - value = ['#FF0000','#008000', '#0000FF']
- Write a Python program to store all elements of an iterable (e.g., the string "python for webscraping") as keys, and as values the number of times they appear. solution

"Bonus" exercise: the Fibonacci Series

Write a Python script to generate and print a list that contains the Fibonacci series of numbers between 1 and n, where n is given as input when the code starts. Features of Fibonacci series:

1. the first two numbers of the series are [1,1] and are predetermined;[1]
2. the sequence F_n is defined by the recurrence equation: $F_n = F_{n-1} + F_{n-2}$

The solution is straightforward: we follow the rule with an exception!

[1] Some use [0,1], it is a matter of definition.

Fibonacci Series: (One of the) Solutions

```
import sys # not necessary, just to print the error
n = int(input("Input a number greater than 2 ")) # read the input n
if n <= 2: # check whether n > 2. If it is not the case, throw an e
    raise ValueError('You must input n > 2!')
    sys.exit()
# for loop solution
d1 = [1, 1] # initiate the list with the predetermined elements
for x in range(2,n+1):
    d1.append(d1[x-1] + d1[x-2]) # append the sum of previous two v
print(d1) # print the resulting list
```

Exercises - **Mandatory**

- Write a function called **CipCiop** that takes a number and
  - If the number is divisible by 3, it should return "Cip";
  - If it is divisible by 5, it should return "Ciop";
  - If it is divisible by both 3 and 5, it should return "CipCiop";
  - Otherwise, it should return the original number.
- Write a function that takes as input an integer (called *limit*) and prints all the prime numbers between 0 and limit.
- Program a Rock-Paper-Scissors game, with the input() method (twice!), comparing the plays and printing the winner. (Yes, it is the same exercise for which you have the solutions, I will check your code. Try not to copy!)

Exercises - **Complimentary**

- Write a program that takes two lists as input and returns the common elements between the two lists. Make sure your program works on two lists of different sizes. Solution
- Write a one-line python code that takes a numeric list as input and returns the even elements of the list; in **one** line. Solution

Solution 2 - 1

```python
# define the lists to be compared
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
# for loop, if, and, or solution
import sys
def ls_overlap(A,B):
    if not isinstance(A,list) or not isinstance(B, list): # check that both inputs are lists
        sys.exit("both inputs must be lists!")
    res = []
    for e in set(A): # for every unique elment in A
        if e in B and B not in res: # if A element is in B and not already in res
            res.append(e)
    return(res)
res = ls_overlap(a,b)
print(res)
# use "set()" and "intersection()"
res = list(set(a).intersection(set(b)))
print(res)
```

back

Solution 2 - 2

```
# define the list
ls_a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
# one-line solution: list comprehension, for, in and modulus
res = [x for x in ls_a if x % 2 == 0]
print(res)
# for loop, in, if and append
res = []
for a in ls_a:
    if a % 2 == 0:
        res.append(a)
print(res)
```

back

Solution 1

```
# initiate the dictionaries
dict1 = {1:10, 2:20}
dict2 = {3:30, 4:40}
dict3 = {5:50,6:60}
# not really the expected output! #
new_dict = {1 : dict1, 2:dict2, 3:dict3}
print(new_dict) # {1: {1: 10, 2: 20}, 2: {3: 30, 4: 40}, 3: {5: 50, 6: 60}}
# Good job, Flavio and Elizaveta!
dict0 = {**dict1,**dict2,**dict3}
print(dict0)
# list comprehension
solution1 = {}
{solution1.update(x) for x in [dict1,dict2,dict3]}
print(solution1) # {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
# for loop
solution2 = {}
for d in (dict1, dict2, dict3): solution2.update(d)
print(solution2) # {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

back

Solution 2

```python
# read the input number
n = int(input("Input a number "))
# dictionary comprehension solution
dc = {x : x*x for x in range(1,n+1)}
print(dc)
# for loop solution
d1 = dict()
for x in range(1,n+1):
    d1[x] = x*x
print(d1)
```

back

Solution 3

```python
dc = {x : x*x for x in range(1,11)}
# for loop solution
total = 0
for x in dc.values():
    total += x
print(total)
# sum() method
total_sum = sum(dc.values())
print(total_sum)
```

back

Solution 4

```
colors = ['red', 'green', 'blue']
values = ['#FF0000','#008000', '#0000FF']
# for solution
d = {}
for i in range(len(colors)):
    d[colors[i]] = values[i]
print(d)
# dict() and zip() solution
color_dict = dict(zip(colors, values))
print(color_dict)
```

back

Solution 5

```
sample = 'python for webscraping'
string_dict = {}
for letter in sample:
    string_dict[letter] = string_dict.get(letter, 0) + 1
print(string_dict)
# input your own sentence
sentence = str(input("Write your sentence"))
string_dict = {}
for letter in sentence:
    string_dict[letter] = string_dict.get(letter, 0) + 1
print(string_dict)
```

back