AI VIETNAM
All-in-One Course

Advanced Python

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2022

Outline

- Exception
- Iterator
- Generator
- Decorator
- Text Libraries
- Sound Libraries

Try/Catch

❖ Used in functions

```python
# exception

number1 = 5
number2 = 0

result = number1 / number2
print(result)
```

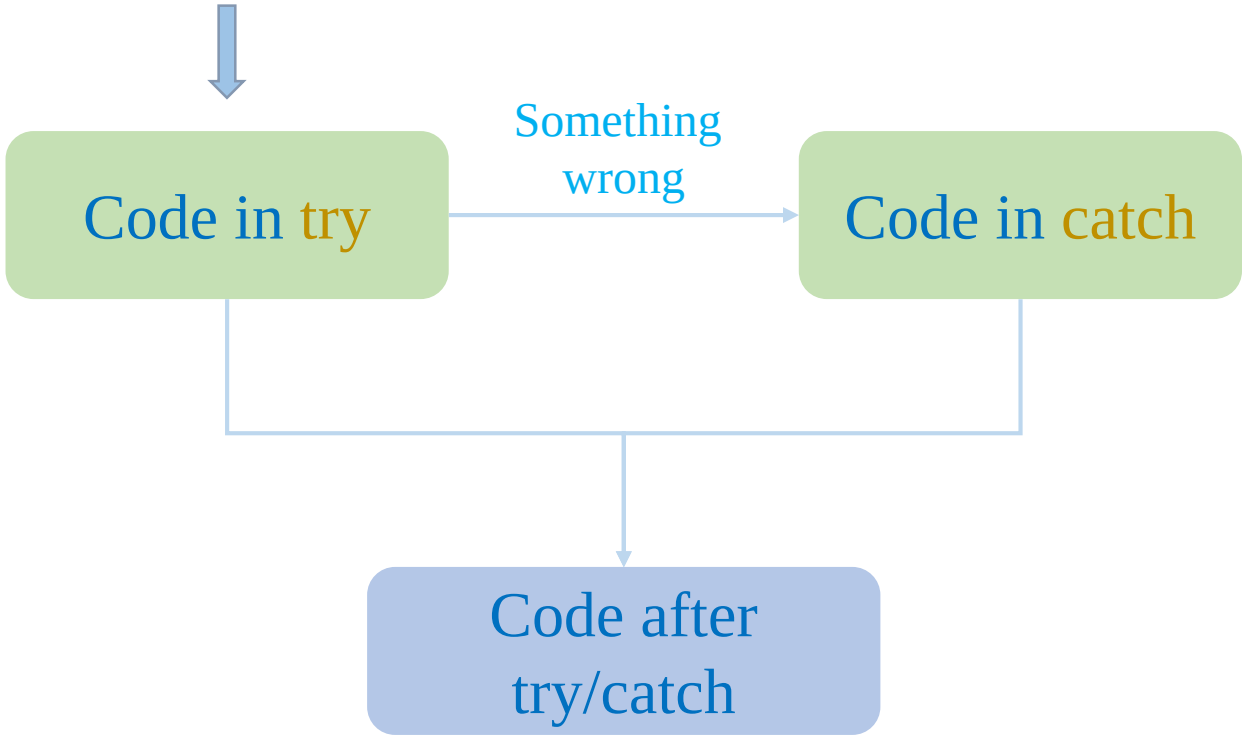
```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback
<ipython-input-1-6495ba73fbbe> in <module>
      4 number2 = 0
      5
----> 6 result = number1 / number2
      7 print(result)

ZeroDivisionError: division by zero
```

```python
# exception

value = 'ai'
number = int(value)
print(number)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most
<ipython-input-2-407689c09ea1> in <module>
      2
      3 value = 'ai'
----> 4 number = int(value)
      5 print(number)

ValueError: invalid literal for int() with base 10: 'ai'
```

Try/Catch

❖ Exception Handling

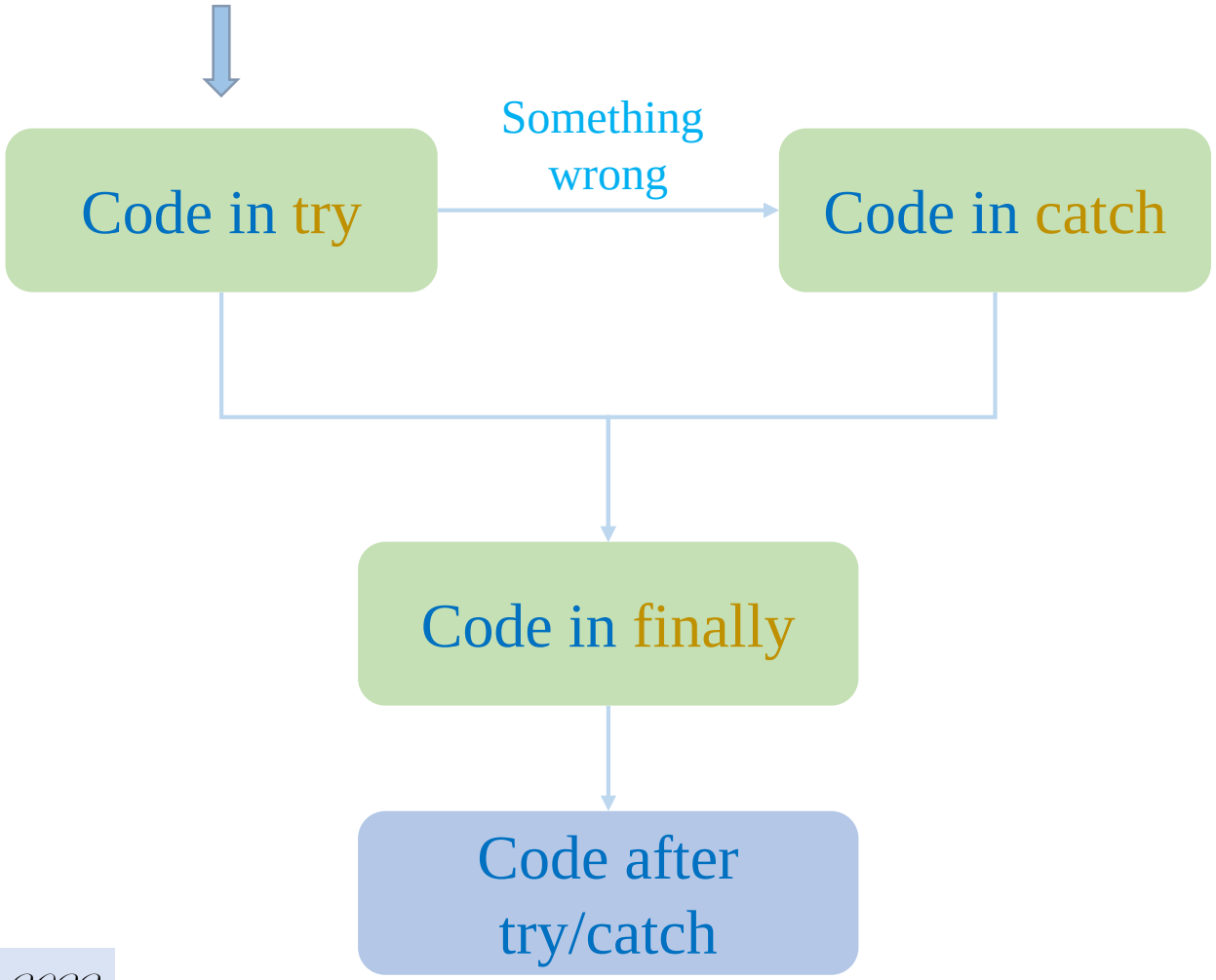



```python
# exception

try:
    value = input()
    number = int(value)
    print(f'The number is {number}')
except:
    print('Something wrong.')
```

```
21
The number is 21
```

Try/Catch

❖ Exception Handling

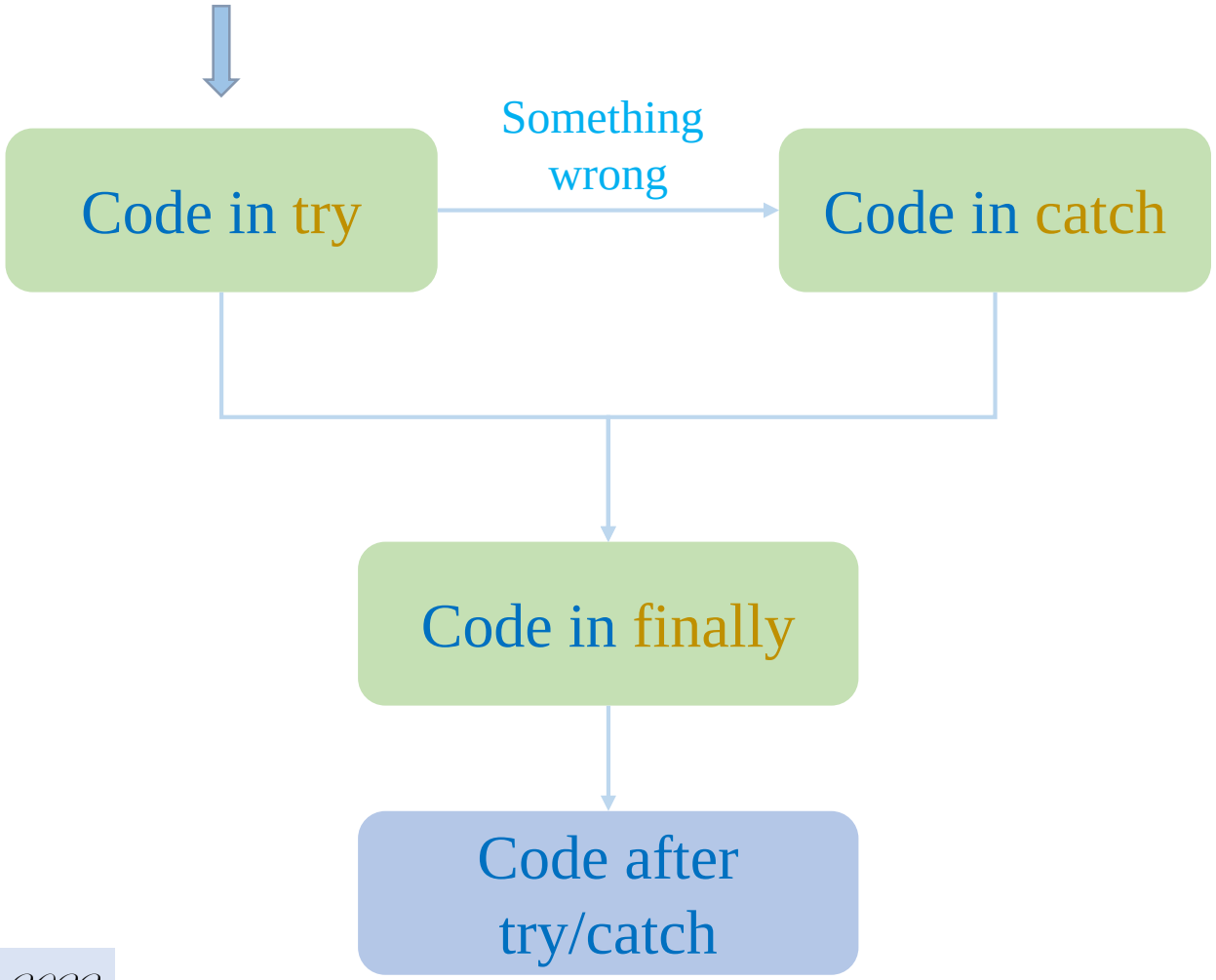



```python
# exception

try:
    value = input()
    number = int(value)
    print(f'The number is {number}')
except:
    print('Something wrong.')
finally:
    print('Do something in finally')
```

```
21
The number is 21
Do something in finally
```

Try/Catch

❖ Exception Handling

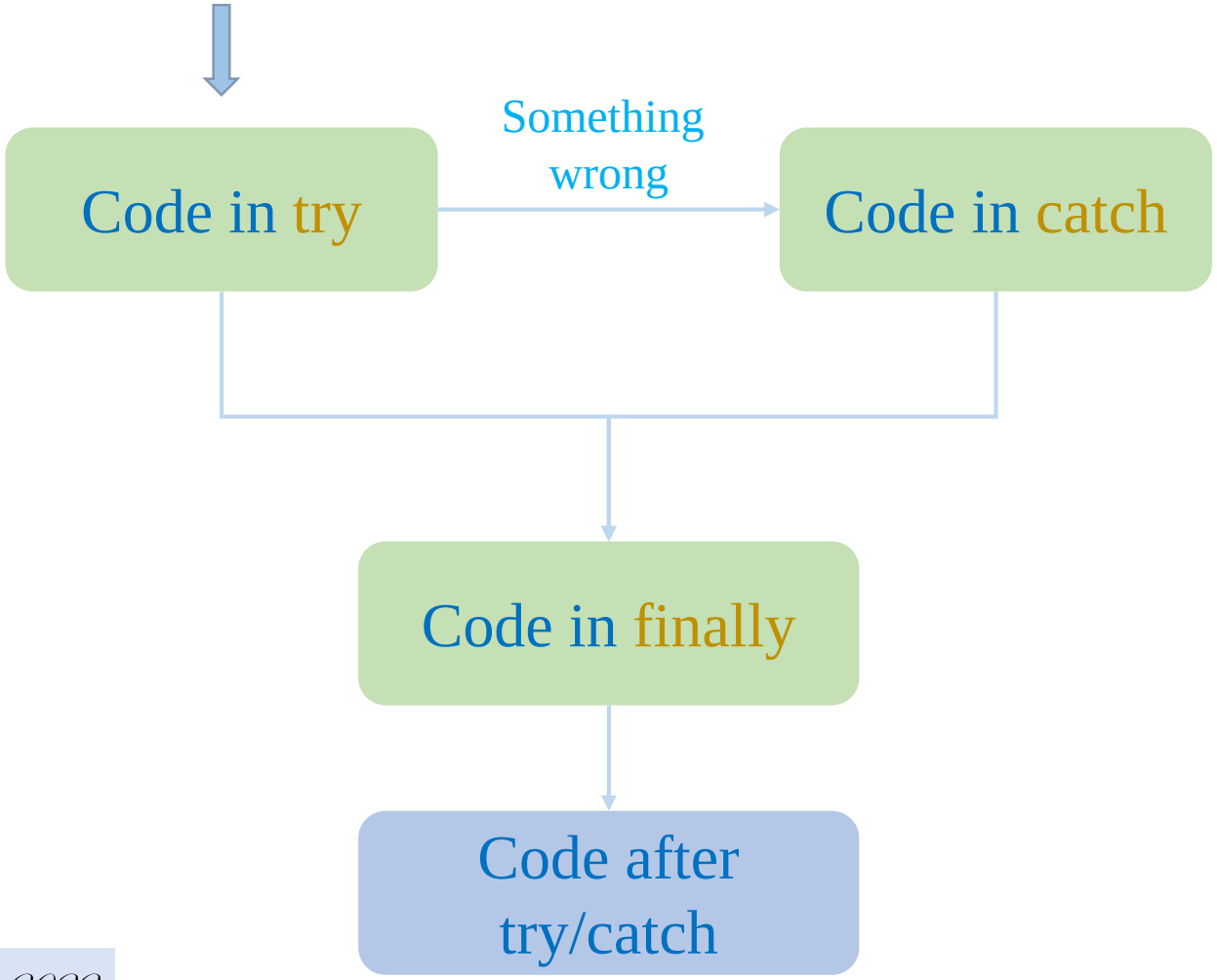



```python
# exception

try:
    value = input()
    number = int(value)
    print(f'The number is {number}')
except ValueError:
    print('Something wrong.')
finally:
    print('Do something in finally')
```

```
d
Something wrong.
Do something in finally
```

Try/Catch

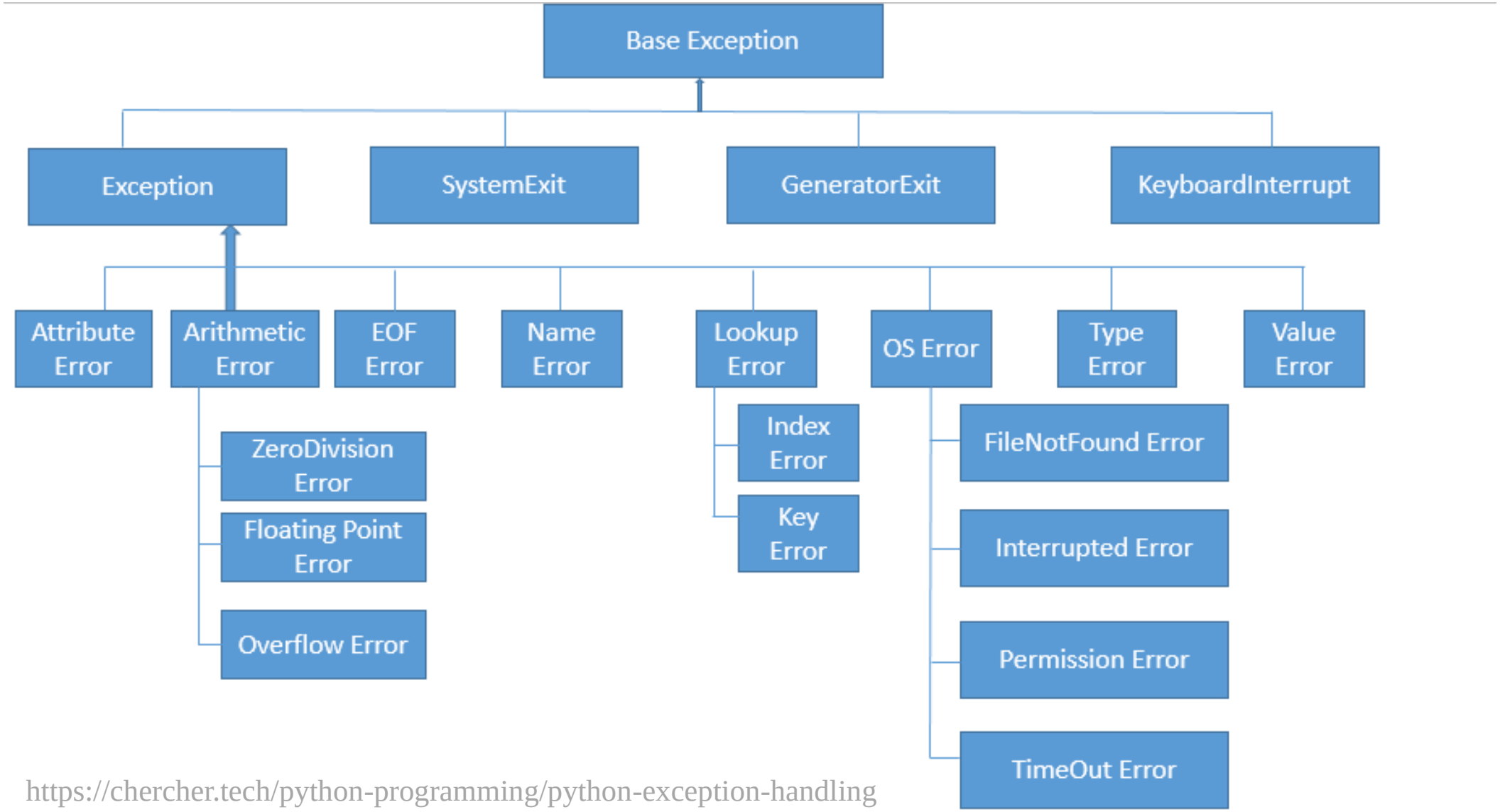
❖ Exception Handling




```python
# exception

try:
    value = input()
    number = int(value)
    print(f'The number is {number}')
except ZeroDivisionError:
    print('Something wrong.')
finally:
    print('Do something in finally')
```

```
d
Do something in finally

---------------------------------------------------------------
ValueError                                Traceback (most
<ipython-input-13-14ecc3112767> in <module>
      3 try:
      4     value = input()
----> 5     number = int(value)
      6     print(f'The number is {number}')
      7 except ZeroDivisionError:

ValueError: invalid literal for int() with base 10: 'd'
```

Exception in Python



https://chercher.tech/python-programming/python-exception-handling

Common Exceptions

❖ KeyError

```python
# KeyError: Khi key không tìm thấy trong dictionary

parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

print(parameters['optimizer'])
print(parameters['loss'])
```

```
Adam
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [2], in <cell line: 8>()
      3 parameters = {'learning_rate': 0.1,
      4               'optimizer': 'Adam',
      5               'metric': 'Accuracy'}
      7 print(parameters['optimizer'])
----> 8 print(parameters['loss'])

KeyError: 'loss'
```

❖ FileNotFoundError

```python
# FileNotFoundError: Lỗi liên quan đến input/output
# Ví dụ mở một file không tồn tại

f = open('non-existing-file.txt', 'r' )
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Input In [4], in <cell line: 4>()
      1 # IOError: raised when an input/output operation fails.
      2 # For example, trying to open a file that does not exist
----> 4 f = open ( "non-existing-file.txt", 'r' )

FileNotFoundError: [Errno 2] No such file or directory: 'non-existing-file.txt'
```

❖ IndexError

```python
# IndexError: khi index không tìm thấy trong một sequence

data = 'AI VIETNAM'
print(data[50])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [3], in <cell line: 4>()
      1 # IndexError
      3 data = 'AI VIETNAM'
----> 4 print(data[50])

IndexError: string index out of range
```

Common Exceptions

❖ ZeroDivisionError

```python
# ZeroDivisionError: Lỗi chia cho 0

a = 5
b = 0
print(a/b)
```

```
---------------------------------------------------------
ZeroDivisionError                         Traceback (most
Input In [6], in <cell line: 5>()
      3 a = 5
      4 b = 0
----> 5 print(a/b)

ZeroDivisionError: division by zero
```

❖ OverflowError

```python
# OverflowError: Lỗi giá trị vượt qua
# khả năng mô tả của biến

import math

value = math.exp(1000)
print(value)
```

```
---------------------------------------------------------
OverflowError                             Traceback (most
Input In [7], in <cell line: 3>()
      1 import math
----> 3 value = math.exp(1000)
      4 print(value)

OverflowError: math range error
```

Common Exceptions

❖ ValueError

```python
# ValueError: Lỗi khi user nhập vào
# một giá trị không hợp lệ

number = int(input('What is your age?'))
print(number)
```

```
What is your age?a
```

```
---------------------------------------------------------------
ValueError                                Traceback (most
Input In [11], in <cell line: 3>()
      1 # ValueError:
----> 3 number = int(input('What is your age?'))
      4 print(number)

ValueError: invalid literal for int() with base 10: 'a'
```

❖ SyntaxError

```python
# SyntaxError: Lỗi sai cú pháp

print(2~)
```

```
  Input In [16]
    print(2~)
           ^
SyntaxError: invalid syntax
```

❖ NameError

```python
# NameError: Lỗi dùng một biến không tồn tại

print(undefined_variable)
```

```
---------------------------------------------------------------
NameError                                 Traceback (most recent
Input In [8], in <cell line: 1>()
----> 1 print(undefined_variable)

NameError: name 'undefined_variable' is not defined
```

Common Exceptions

❖ IndentationError

```
1 # IndentationError: Lỗi về indentation
2
3 a = 5
4  b = 7
5 print(a+b)
```

```
  Input In [17]
    b = 7
    ^
IndentationError: unexpected indent
```

❖ TypeErrors

```
1 # TypeErrors: Lỗi kết hợp các kiểu
2 # dữ liệu không phù hợp
3
4 print('AI VIETNAM' + 2022)
```

```
---------------------------------------------------------------
TypeError                                 Traceback (most
Input In [26], in <cell line: 4>()
      1 # TypeErrors: Lỗi kết hợp các kiểu
      2 # dữ liệu không phù hợp
----> 4 print('AI VIETNAM' + 2022)

TypeError: can only concatenate str (not "int") to str
```

Common Exceptions

❖ StopIteration

```python
# StopIteration: Xảy ra khi tiếp tục gọi hàm next()
# khi đã duyệt hết phần tử của một iterator

data = ['AI', 'VIETNAM']

iterator = iter(data)
print(next(iterator))
print(next(iterator))
print(next(iterator))
```

```
AI
VIETNAM


---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
Input In [20], in <cell line: 7>()
      5 print(next(iterator))
      6 print(next(iterator))
----> 7 print(next(iterator))

StopIteration:
```

Common Exceptions

EOFError

```
>>> data = input('What is your name?')
What is your name?Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
EOFError
>>>
```

hit Ctrl+D while waiting for input from users

PermissionError

```
1 # PermissionError: Lỗi khi không có quyền truy cập vào folders/files
2
3 f = open(r'C:\Windows\System32', 'r' )
```

```
---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
Input In [21], in <cell line: 3>()
      1 #
----> 3 f = open(r'C:\Windows\System32', 'r' )

PermissionError: [Errno 13] Permission denied: 'C:\\Windows\\System32'
```

Iterator

❖ Introduction

```python
# iterate a list

fruits = ['apple', 'banana', 'melon', 'peach']

for fruit in fruits:
    print(fruit)
```

```
apple
banana
melon
peach
```

```python
# iterate a tuple

fruits = ('apple', 'banana', 'melon')

for fruit in fruits:
    print(fruit)
```

```
apple
banana
melon
```

```python
# iterate a dictionary

parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

for key in parameters:
    print(key, parameters.get(key))
```

```
learning_rate 0.1
optimizer Adam
metric Accuracy
```

```python
# use range()

for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

```python
# iterate a string

greeting = 'Hello'

for char in greeting:
    print(char)
```

```
H
e
l
l
o
```

Iterator

❖ Range and Iterator

```python
# range and iterator

for v in range(3):
    print('Value: ', v)
```

```
Value:  0
Value:  1
Value:  2
```

```python
# range and iterator

range_object = range(3)
print(range_object)

iterator = iter(range_object)
print(next(iterator))
print(next(iterator))
print(next(iterator))
```

```
range(0, 3)
0
1
2
```

Iterator

❖ Range and Iterator

```
# range and iterator

for v in range(3):
    print('Value: ', v)
```

```
Value:  0
Value:  1
Value:  2
```

```
# range and iterator

range_object = range(3)
print(range_object)

iterator = iter(range_object)
print(next(iterator))
print(next(iterator))
print(next(iterator))
print(next(iterator))
```

```
range(0, 3)
0
1
2

---------------------------------------------------------------
StopIteration                                 Traceback
<ipython-input-31-38407993d207> in <module>
      8 print(next(iterator))
      9 print(next(iterator))
---> 10 print(next(iterator))

StopIteration:
```

Iterator

❖ Range and Iterator

```python
# range and iterator

for v in range(3):
    print('Value: ', v)
```

```
Value:  0
Value:  1
Value:  2
```

```python
# range and iterator

range_object = range(3)
iterator = iter(range_object)

while True:
    v = next(iterator)
    print('Value ', v)
```

```
Value  0
Value  1
Value  2

---------------------------------------------------------------------------
StopIteration                                             Traceback
<ipython-input-8-9548022965d3> in <module>
      3
      4 while True:
----> 5     v = next(iterator)
      6     print('Value ', v)

StopIteration:
```

Iterator

❖ Range and Iterator

```python
# range and iterator

for v in range(3):
    print('Value: ', v)
```

```
Value:  0
Value:  1
Value:  2
```

```python
# range and iterator

range_object = range(3)
iterator = iter(range_object)

try:
    while True:
        v = next(iterator)
        print('Value: ', v)
except StopIteration:
    pass
finally:
    del iterator
```

```
Value:  0
Value:  1
Value:  2
```

Iterator

❖ Range and Iterator

```
# list and iterator

a_list = [1, 2, 3]
for v in a_list:
    print('Value: ', v)
```

```
Value:  1
Value:  2
Value:  3
```

```
# list and iterator

a_list = [1, 2, 3]

# get an iterator
iterator = iter(a_list)

print(next(iterator))
print(next(iterator))
print(next(iterator))
print(next(iterator))
```

```
1
2
3

---------------------------------------------------------------
StopIteration                                 Traceback
<ipython-input-35-da58936b18b1> in <module>
      9 print(next(iterator))
     10 print(next(iterator))
---> 11 print(next(iterator))

StopIteration:
```

Iterator

❖ List and Iterator

```python
# list and iterator

a_list = [1, 2, 3]
for v in a_list:
    print('Value: ', v)
```

```
Value:  1
Value:  2
Value:  3
```

```python
# list and iterator

a_list = [1, 2, 3]
iterator = iter(a_list)

try:
    while True:
        v = next(iterator)
        print('Value: ', v)
except StopIteration:
    pass
finally:
    del iterator
```

```
Value:  1
Value:  2
Value:  3
```

Iterator

❖ Tuple and Iterator

```python
# tuple and iterator

a_tuple = (1, 2, 3)
for v in a_tuple:
    print('Value ', v)
```

```
Value  1
Value  2
Value  3
```

```python
# tuple and iterator

a_tuple = (1, 2, 3)

# get an iterator
iterator = iter(a_tuple)

print(next(iterator))
print(next(iterator))
print(next(iterator))
print(next(iterator))
```

```
1
2
3

---------------------------------------------------------------------------
StopIteration                             Traceback
<ipython-input-14-df3a94ab29ed> in <module>
      9 print(next(iterator))
     10 print(next(iterator))
---> 11 print(next(iterator))

StopIteration:
```

Iterator

❖ Tuple and Iterator

```python
# tuple and iterator

a_tuple = (1, 2, 3)
for v in a_tuple:
    print('Value ', v)
```

```
Value  1
Value  2
Value  3
```

```python
# tuple and iterator

a_tuple = (1, 2, 3)
iterator = iter(a_tuple)

try:
    while True:
        v = next(iterator)
        print('Value: ', v)
except StopIteration:
    pass
finally:
    del iterator
```

```
Value:  1
Value:  2
Value:  3
```

Iterator

❖ String and Iterator

```
1  # string and iterator
2
3  name = 'ai'
4  for v in name:
5      print('Value: ', v)
```

```
Value:  a
Value:  i
```

```
1  # string and iterator
2
3  name = 'ai'
4
5  # get an iterator
6  iterator = iter(name)
7
8  print(next(iterator))
9  print(next(iterator))
10 print(next(iterator))
```

```
a
i

---------------------------------------------------------------
StopIteration                                Traceback
<ipython-input-18-f93d9638a1a9> in <module>
      8 print(next(iterator))
      9 print(next(iterator))
---> 10 print(next(iterator))

StopIteration:
```

Iterator

❖ String and Iterator

```python
# string and iterator

name = 'ai'
for v in name:
    print('Value: ', v)
```

```
Value:  a
Value:  i
```

```python
# string and iterator

name = 'ai'
iterator = iter(name)

try:
    while True:
        v = next(iterator)
        print('Value: ', v)
except StopIteration:
    pass
finally:
    del iterator
```

```
Value:  a
Value:  i
```

Iterator

❖ Set and Iterator

```
1  # set and iterator
2
3  a_set = {1, 3, 5}
4  for v in a_set:
5      print('Value: ', v)
```

```
Value:  1
Value:  3
Value:  5
```

```
1  # set and iterator
2
3  a_set = {1, 3, 5}
4
5  # get an iterator
6  iterator = iter(a_set)
7
8  print(next(iterator))
9  print(next(iterator))
10 print(next(iterator))
11 print(next(iterator))
```

```
1
3
5

---------------------------------------------------------------
StopIteration                                 Traceback
<ipython-input-41-2a143e840bc9> in <module>
      9 print(next(iterator))
     10 print(next(iterator))
---> 11 print(next(iterator))

StopIteration:
```

Iterator

❖ Set and Iterator

```python
# set and iterator

a_set = {1, 3, 5}
for v in a_set:
    print('Value: ', v)
```

```
Value:  1
Value:  3
Value:  5
```

```python
# set and iterator

a_set = {1, 3, 5}
iterator = iter(a_set)

try:
    while True:
        v = next(iterator)
        print('Value: ', v)
except StopIteration:
    pass
finally:
    del iterator
```

```
Value:  1
Value:  3
Value:  5
```

Iterator

❖ Dictionary and Iterator

```python
# dictionary and iterator

parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

for key in parameters:
    print(key)
```

```
learning_rate
optimizer
metric
```

```python
# dictionary and iterator

parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

# get an iterator
iterator = iter(parameters)

print(next(iterator))
print(next(iterator))
print(next(iterator))
print(next(iterator))
```

```
learning_rate
optimizer
metric

---------------------------------------------------------------
StopIteration                                 Traceback
<ipython-input-43-b68adca850c8> in <module>
     11 print(next(iterator))
     12 print(next(iterator))
---> 13 print(next(iterator))

StopIteration:
```

Iterator

❖ Dictionary and Iterator

```python
# dictionary and iterator

parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

for key in parameters:
    print(key)
```

```
learning_rate
optimizer
metric
```

```python
# dictionary and iterator

parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}
iterator = iter(parameters)

try:
    while True:
        v = next(iterator)
        print('Value ', v)
except StopIteration:
    pass
finally:
    del iterator
```

```
Value  learning_rate
Value  optimizer
Value  metric
```

Iterator

1. Create iterator using iter()
2. Get value using next()

```
# iterator

a_list = [1, 2, 3]

# get an iterator
a_iter = iter(a_list)

print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
```

```
1
2
3
```

```
# iterator

a_list = [1, 2, 3]

# get an iterator
a_iter = iter(a_list)

print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
```

```
1
2
3

---------------------------------------------------------------------------
StopIteration                                             Traceback
<ipython-input-1-251788936f93> in <module>
      9 print(next(a_iter))
     10 print(next(a_iter))
---> 11 print(next(a_iter))

StopIteration:
```

Iterator

❖ Classes and Iterator

```python
class AdditionValueSeries:
    '''
    Comment ...
    '''

    def __init__(self, value, max=0):
        self.value = value
        self.max = max

# create an object
numbers = AdditionValueSeries(5, 2)
# return 5, 6, 7
```

Iterator

```python
class AdditionValueSeries:
    '''
    Comment ...
    '''

    def __init__(self, value, max=0):
        self.value = value
        self.max = max

    def __iter__(self):
        self.n = 0
        return self

    def __next__(self):
        if self.n <= self.max:
            result = self.n + self.value
            self.n += 1

            return result
        else:
            raise StopIteration
```

1. Implement iter()
2. Implement next()
3. Raise StopIteration

```python
# create an object
numbers = AdditionValueSeries(5, 2)

for n in numbers:
    print(n)
```

```
5
6
7
```

Iterator

```python
class AdditionValueSeries:
    '''
    Comment ...
    '''

    def __init__(self, value, max=0):
        self.value = value
        self.max = max

    def __iter__(self):
        self.n = 0
        return self

    def __next__(self):
        if self.n <= self.max:
            result = self.n + self.value
            self.n += 1

            return result
        else:
            raise StopIteration
```

```python
# create an object
numbers = AdditionValueSeries(5, 2)

# create an iterator
a_iter = iter(numbers)

# get iterator elements
print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
```

```
5
6
7

---------------------------------------------------------------------------
StopIteration                                       Traceback
<ipython-input-46-ab07f2303243> in <module>
      9 print(next(a_iter))
     10 print(next(a_iter))
---> 11 print(next(a_iter))

<ipython-input-44-577d77e46614> in __next__(self)
     19             return result
     20         else:
---> 21             raise StopIteration

StopIteration:
```

Iterator

```python
class AdditionValueSeries:
    '''
    Comment ...
    '''

    def __init__(self, value, max=0):
        self.value = value
        self.max = max

    def __iter__(self):
        self.n = 0
        return self

    def __next__(self):
        if self.n <= self.max:
            result = self.n + self.value
            self.n += 1

            return result
        else:
            raise StopIteration
```

```python
# create an object
numbers = AdditionValueSeries(5, 2)

for number in numbers:
    print(number)
```

```
5
6
7
```

Iterator

```python
class AdditionValueSeries:
    '''
    Comment ...
    '''

    def __init__(self, value, max=0):
        self.value = value
        self.max = max

    def __iter__(self):
        self.n = 0
        return self

    def __next__(self):
        if self.n <= self.max:
            result = self.n + self.value
            self.n += 1

            return result
        else:
            raise StopIteration
```

```python
# classes and iterator

numbers = AdditionValueSeries(5, 2)
iterator = iter(numbers)

try:
    while True:
        v = next(iterator)
        print(v)
except StopIteration:
    pass
finally:
    del iterator
```

```
5
6
7
```

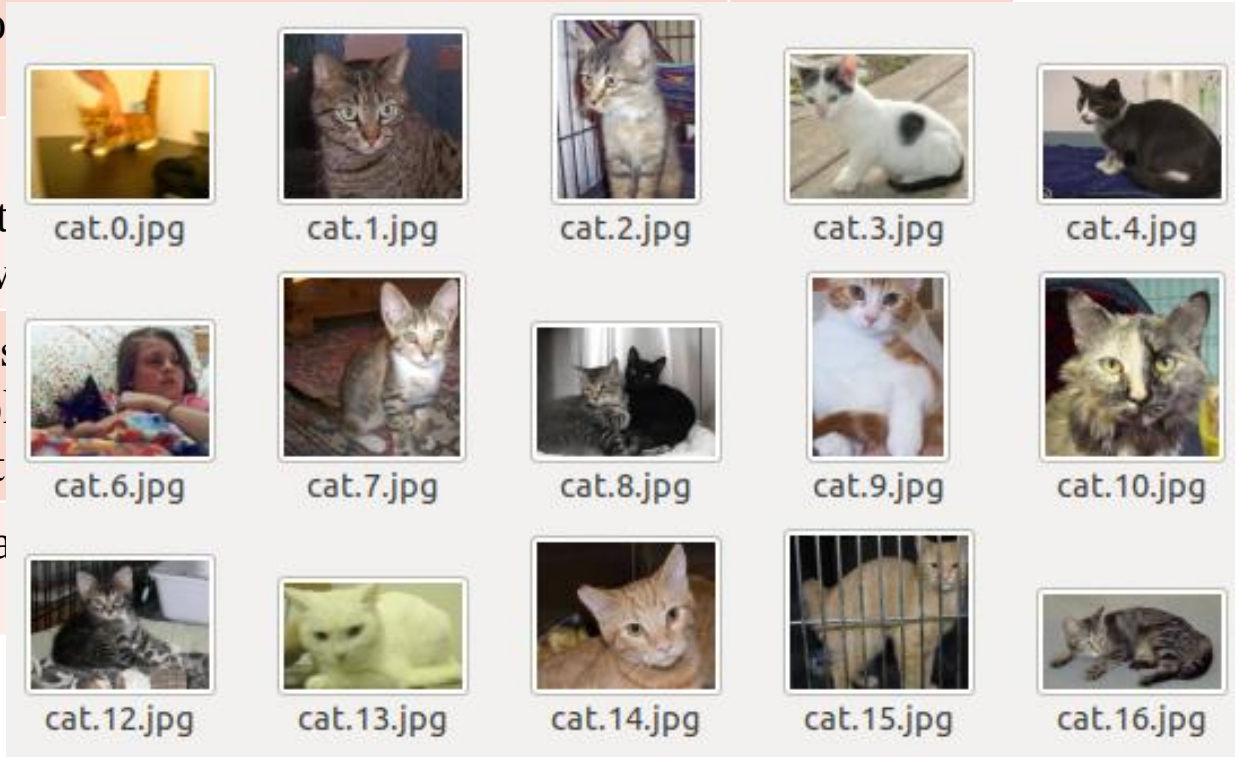
Iterator

- For large datasets

Features | Label

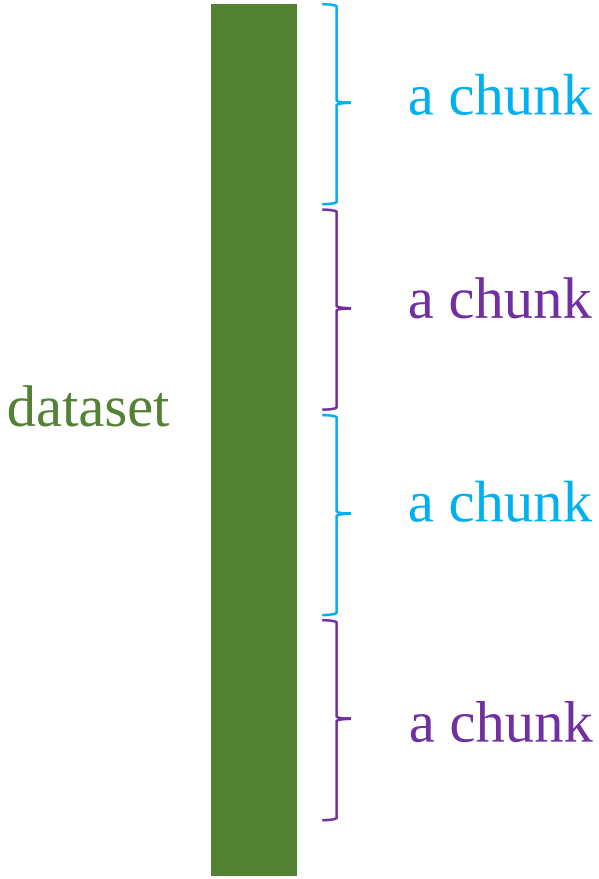
crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9

“A wonderful little production. The filming technique is very unassuming- very old-time-BBC fashion and gives a comforting, and sometimes disco entire piece…..”	positive
“This show was an amazing, fresh & innovative idea in the 70's years were brilliant, but things dropped off after that. By 1990, t anymore, and it's continued its decline further to the complete w	
“I thought this was a wonderful way to spend time on a too hot s conditioned theater and watching a light-hearted comedy. The pl witty and the characters are likable (even the well bread suspect	
“BTW Carver gets a very annoying sidekick who makes you wa minutes he's on screen.”	



Iterator

❖ **For large datasets**



A whole dataset does not fit the memory, but a chunk does

```python
class DataLoading:
    def __init__(self, path, data_size, chunk_size=0):
        self.path = path
        self.data_size  = data_size
        self.chunk_size = chunk_size

    def __iter__(self):
        self.index = 0
        return self

    def __next__(self):
        if self.index*self.chunk_size <= self.data_size:
            # load data from hard disk
            from_index = self.index*self.chunk_size
            to_index = min(self.data_size, (self.index+1)*self.chunk_size)
            data = [i for i in range(from_index, to_index)]

            self.index += 1

            return data
        else:
            raise StopIteration
```

Generator

```python
# Generator

def addValueSeries(value, max=0):
    n = 0
    while n <= max:
        yield n + value
        n += 1
```

```python
numbers = addValueSeries(5, 2)

# create an iterable
a_iter = iter(numbers)

# get iterator elements
print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
```

```
5
6
7
```

```python
# Generator

numbers = addValueSeries(5, 2)

# create an iterable
a_iter = iter(numbers)

# get iterator elements
print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
```

```
5
6
7

---------------------------------------------------------------------------
StopIteration                                             Traceback
<ipython-input-56-48a0c5434dc3> in <module>
     10 print(next(a_iter))
     11 print(next(a_iter))
---> 12 print(next(a_iter))

StopIteration:
```

Generator

```python
# Generator

def addValueSeries(value, max=0):
    n = 0
    while n <= max:
        yield n + value
        n += 1
```

```python
numbers = addValueSeries(5, 2)

# create an iterable
a_iter = iter(numbers)

# get iterator elements
print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
```

```
5
6
7
```

```python
# Generator

numbers = addValueSeries(5, 2)

for number in numbers:
    print(number)
```

```
5
6
7
```

```python
# Generator

numbers = addValueSeries(5, 2)
iterator = iter(numbers)

try:
    while True:
        v = next(iterator)
        print(v)
except StopIteration:
    pass
finally:
    del iterator
```

```
5
6
7
```

Generator

❖ Create a generator

```python
# Generator expression

data = (i*i for i in range(3))
```

```python
print(data)
print(type(data))
```

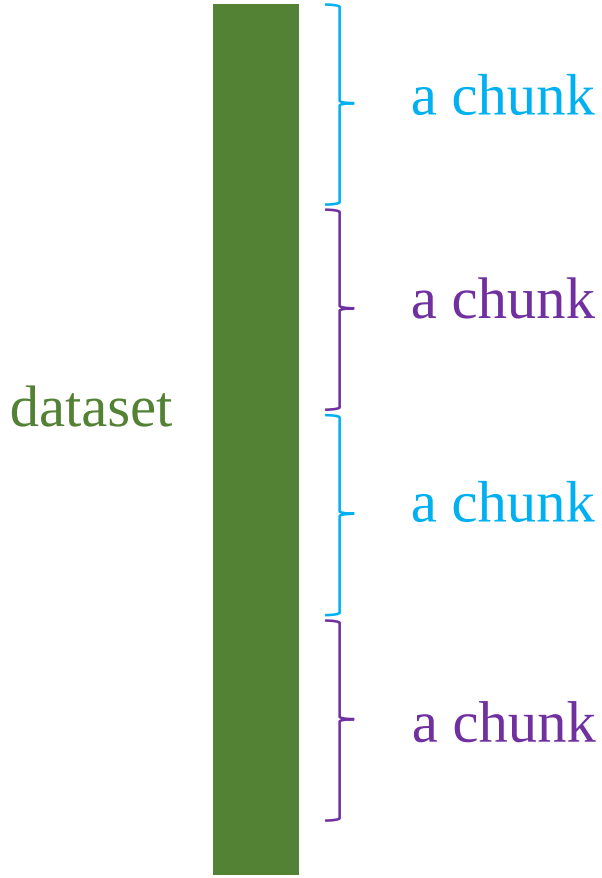
```
<generator object <genexpr> at 0x000
<class 'generator'>
```

```python
# create an iterable
a_iter = iter(data)

# get iterator elements
print(next(a_iter))
print(next(a_iter))
print(next(a_iter))
```

```
0
1
4
```

Iterator

❖ **For large datasets**



A whole dataset does not fit the memory, but a chunk does

```python
def DataLoadingGenerator(path, data_size, chunk_size=0):
    index = 0

    while index*chunk_size <= data_size:
        # load data from harddisk
        from_index = index*chunk_size
        to_index = min(data_size, (index+1)*chunk_size)
        data = [i for i in range(from_index, to_index)]

        index += 1

        yield data
```

```python
generator = DataLoadingGenerator('some path', 22, 5)
```

```python
for chunk in generator:
    print(chunk)
```

```
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]
[10, 11, 12, 13, 14]
[15, 16, 17, 18, 19]
[20, 21]
```

Outline

- Exception
- Iterator
- Generator
- Decorator
- Text Libraries
- Sound Libraries

Inner Functions

❖ Functions as variables

```
1  def power(base):
2      return base**2
3
4  var = power
5  print(var)
6  print(var(3))
```

```
<function power at 0x000001F93C3B2280>
9
```

```
1  def generate_power(exponent):
2      def power(base):
3          return base**exponent
4      return power
```

```
1  var = generate_power(2)
2  print(var)
3  print(var(3))
```

```
<function generate_power.<locals>.power at 0x000001F93C3B2EE0>
9
```

Decorator

❖ Motivation

$$y = ax^2 + bx + c$$

$$\Delta = b^2 - 4ac$$

...

```python
import math

def quadratic_equation(a, b, c):
    result = ()

    # compute delta
    delta = b*b - 4*a*c

    if delta == 0: # one solution
        x = (-b+math.sqrt(delta))/2*a
        result = (x,)
    elif delta > 0: # two solutions
        x1 = (-b+math.sqrt(delta))/(2*a)
        x2 = (-b-math.sqrt(delta))/(2*a)
        result = (x1,x2)

    return result
```

```python
import math

def quadratic_equation(a, b, c):
    result = ()

    # compute delta
    delta = b*b - 4*a*c

    if delta == 0: # one solution
        x = (-b+math.sqrt(delta))/2*a
        result = (x,)
    elif delta > 0: # two solutions
        x1 = (-b+math.sqrt(delta))/(2*a)
        x2 = (-b-math.sqrt(delta))/(2*a)
        result = (x1,x2)

    return result
```

```python
result = quadratic_equation(a=5, b=0, c=1)
print(result)
```

```
()
```

```python
result = quadratic_equation(a=5, b=5, c=-1)
print(result)
```

```
(0.17082039324993695, -1.170820393249937)
```

```python
result = quadratic_equation(a=4, b=4, c=1)
print(result)
```

```
(-8.0,)
```

```python
result = quadratic_equation(a=0, b=3, c=1)
print(result)
```

```
---------------------------------------------------------------
ZeroDivisionError                         Traceback (most rece
```

Decorator

❖ Motivation

$$y = ax^2 + bx + c$$

$$\Delta = b^2 - 4ac$$

...

Constraint

$$a \neq 0$$

How to add the constraint without modifying the existing code?

```python
import math

def quadratic_equation(a, b, c):
    result = ()

    # compute delta
    delta = b*b - 4*a*c

    if delta == 0: # one solution
        x = (-b+math.sqrt(delta))/2*a
        result = (x,)
    elif delta > 0: # two solutions
        x1 = (-b+math.sqrt(delta))/(2*a)
        x2 = (-b-math.sqrt(delta))/(2*a)
        result = (x1,x2)

    return result
```

Decorator

❖ Solution 1

```python
def helper_function(a, b, c):
    if a == 0:
        return None
    else:
        return quadratic_equation(a, b, c)
```

```python
def helper_function(a, b, c):
    def check():
        if a == 0:
            return None
        else:
            return quadratic_equation(a, b, c)
    return check()
```

```python
result = helper_function(a=5, b=0, c=1)
print(result)
```

```
()
```

```python
result = helper_function(a=5, b=5, c=-1)
print(result)
```

```
(0.17082039324993695, -1.170820393249937)
```

```python
result = helper_function(a=4, b=4, c=1)
print(result)
```

```
(-8.0,)
```

```python
result = helper_function(a=0, b=3, c=1)
print(result)
```

```
None
```

Decorator

- Solution 2
  - Using an inner function

```
def helper_function(func):
    def check(a, b, c):
        if a == 0:
            return None
        else:
            return func(a, b, c)
    return check
```

```
check_fn = helper_function(quadratic_equation)
print(check_fn(a=5, b=0, c=1))
```

```
()
```

```
check_fn = helper_function(quadratic_equation)
print(check_fn(a=5, b=5, c=-1))
```

```
(0.17082039324993695, -1.170820393249937)
```

```
check_fn = helper_function(quadratic_equation)
print(check_fn(a=4, b=4, c=1))
```

```
(-8.0,)
```

```
check_fn = helper_function(quadratic_equation)
print(check_fn(a=0, b=3, c=1))
```

```
None
```

Decorator

❖ Solution 3

```
def helper_function(func):
    def check(a, b, c):
        if a == 0:
            return None
        else:
            return func(a, b, c)
    return check
```

```
import math

@helper_function
def quadratic_equation(a, b, c):
    result = ()

    # compute delta
    delta = b*b - 4*a*c

    if delta == 0: # one solution
        x = (-b+math.sqrt(delta))/2*a
        result = (x,)
    elif delta > 0: # two solutions
        x1 = (-b+math.sqrt(delta))/(2*a)
        x2 = (-b-math.sqrt(delta))/(2*a)
        result = (x1,x2)

    return result
```

Decorator

❖ Solution 3

```python
def helper_function(func):
    def check(a, b, c):
        if a == 0:
            return None
        else:
            return func(a, b, c)
    return check
```

```python
result = quadratic_equation(a=5, b=0, c=1)
print(result)
```

```
()
```

```python
result = quadratic_equation(a=5, b=5, c=-1)
print(result)
```

```
(0.17082039324993695, -1.170820393249937)
```

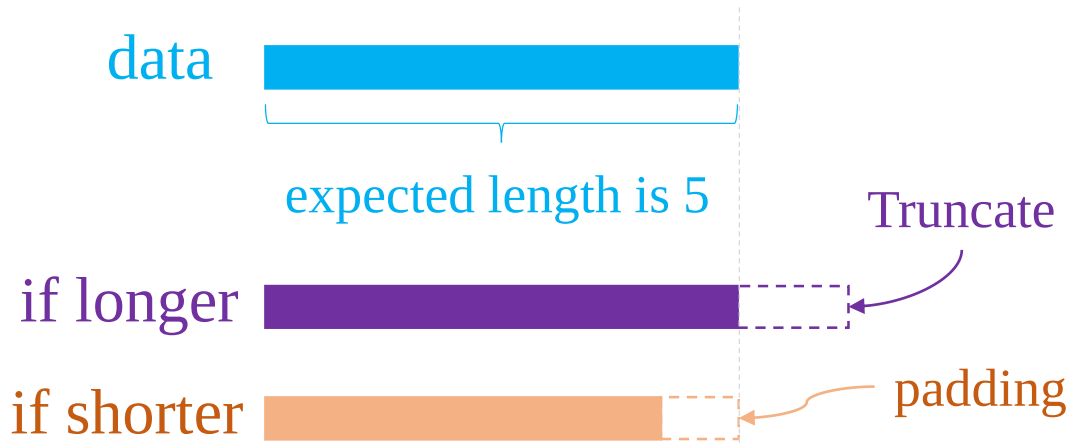
```python
result = quadratic_equation(a=4, b=4, c=1)
print(result)
```

```
(-8.0,)
```

```python
result = quadratic_equation(a=0, b=3, c=1)
print(result)
```

```
None
```

Walrus Operator



❖ Variable assignments inside of larger expressions

```python
# Text processing

def preprocess(data, maxLength):
    dataLength = len(data)

    if dataLength <= maxLength:
        padding = maxLength - dataLength
        print(f'padding={padding}')
    else:
        cutting = dataLength - maxLength
        print(f'cutting={cutting}')
```

```python
data = ['a', 'b', 'c']
maxLength = 5

preprocess(data, maxLength)
```

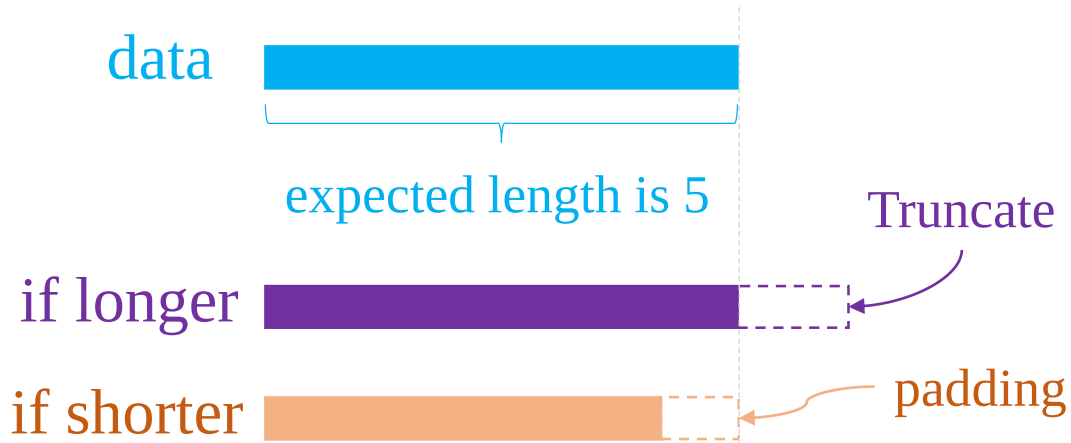
```
padding=2
```

```python
data = ['a', 'b', 'c', 'd', 'e', 'f']
maxLength = 5

preprocess(data, maxLength)
```

```
cutting=1
```

Walrus Operator

❖ Variable assignments inside of larger expressions




```python
# Text processing
# Using Walrus operator

def preprocess(data, maxLength):
    if (dataLength := len(data)) <= maxLength:
        padding = maxLength - dataLength
        print(f'padding={padding}')
    else:
        cutting = dataLength - maxLength
        print(f'cutting={cutting}')
```

```python
data = ['a', 'b', 'c']
maxLength = 5

preprocess(data, maxLength)
```

```
padding=2
```

```python
data = ['a', 'b', 'c', 'd', 'e', 'f']
maxLength = 5

preprocess(data, maxLength)
```

```
cutting=1
```

Walrus Operator

❖ Variable assignments inside of larger expressions

Generate random numbers until getting 5

```python
import random

while True:
    value = random.randint(0, 9)
    print(value)

    if value==5:
        break
```

```
6
8
8
5
```

```python
import random

value = random.randint(0, 9)
print(value)

while value != 5:
    value = random.randint(0, 9)
    print(value)
```

```
6
5
```

Walrus Operator

❖ Variable assignments inside of larger expressions

Generate random numbers until getting 5

```python
# Generate random numbers until getting 5
import random

while (value := random.randint(0, 9)) != 5:
    print(value)
```

```
8
6
8
8
9
1
```

```python
# Generate random numbers until getting 5
import random

while value := random.randint(0, 9) != 5:
    print(value)
```

```
True
True
True
```

Outline

- Exception
- Iterator
- Generator
- Decorator
- Text Libraries
- Sound Libraries

Text Libraries

❖ Underthesea Library

Open-source Vietnamese Natural Language Process Toolkit

📝 **Description**:

- **A Vietnamese NLP toolkit:** API for apply some pretrained NLP models.
- **A Pytorch library:** backed by Pytorch libraray.
- **A open-source software**

🔨 **Installation**:

```
1 !pip install underthesea
```

source: https://github.com/undertheseanlp/underthesea

Text Libraries

❖ Underthesea Library: Usages

✸ POS Tagging

Input	She	sells	seashells	on	the	seashore
Output	PRP	VBZ	NNS	IN	DT	NN

🥹 **Syntax:** `underthesea.pos_tag(`*`sentence`*`)`

Input:

- *sentence*: raw sentence (unicode, str)

Output:

- *tokens*: list of tuple with word, pos tag

```
1 #1.Gán nhãn từ loại (POS tagging)
2 from underthesea import pos_tag
3 pos_tag("học sinh đang học")
```

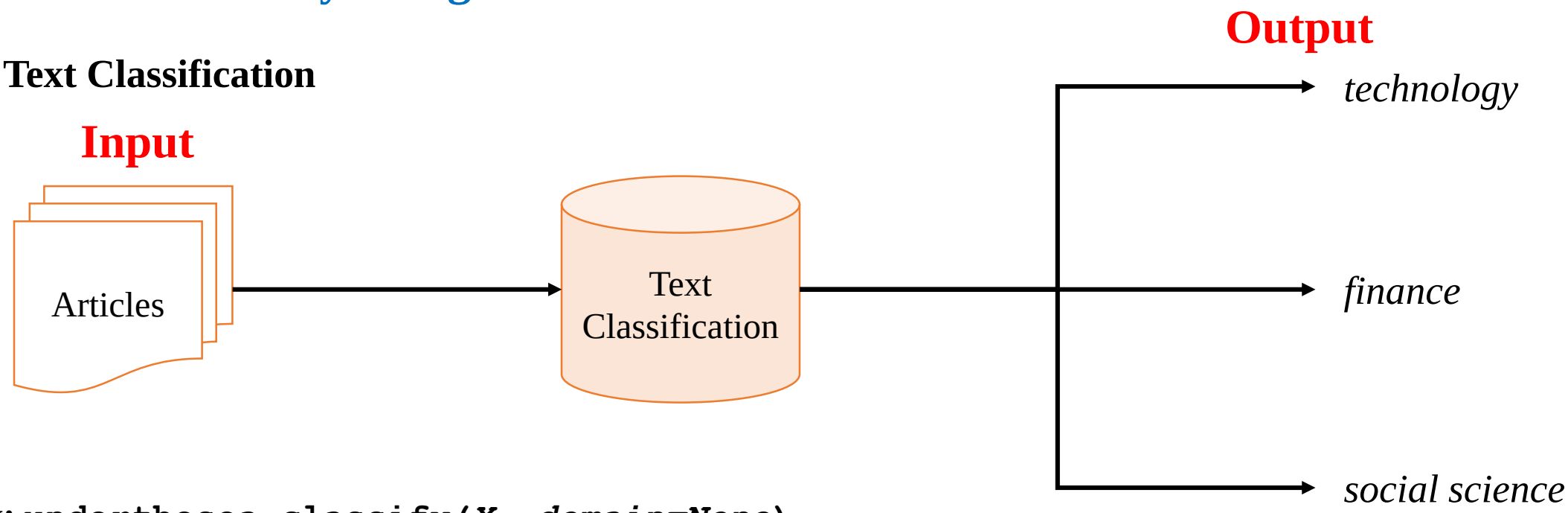
```
[('học sinh', 'N'), ('đang', 'R'), ('học', 'V')]
```

source: https://github.com/undertheseanlp/underthesea

Text Libraries

❖ Underthesea Library: Usages

Text Classification



😲 **Syntax:** `underthesea.classify(X, domain=None)`

Input:

- *X*: raw sentence (unicode, str)
- *domain*: domain of text

Output:

- *tokens*: list categories of sentence

```
1 #2.Phân loại văn bản (Text classification)
2 from underthesea import classify
3 classify("giá cổ phiếu đang có nhiều biến động trong thời gian qua")
```

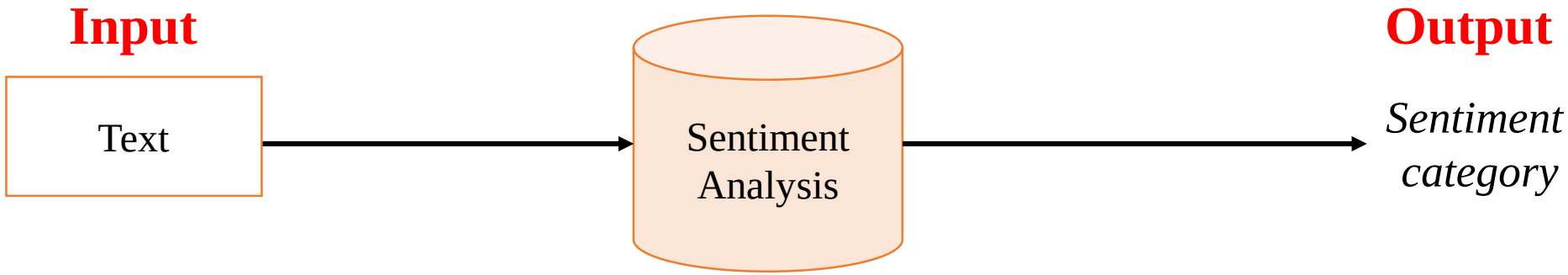
```
['kinh_doanh']
```

source: https://github.com/undertheseanlp/underthesea

Text Libraries

❖ Underthesea Library: Usages

Sentiment Analysis



😢 **Syntax:** `underthesea.sentiment(X, domain='general')`

Input:

- *X*: raw sentence (unicode, str)
- *domain:* domain of text

Output:

- *labels: sentiment of sentence*

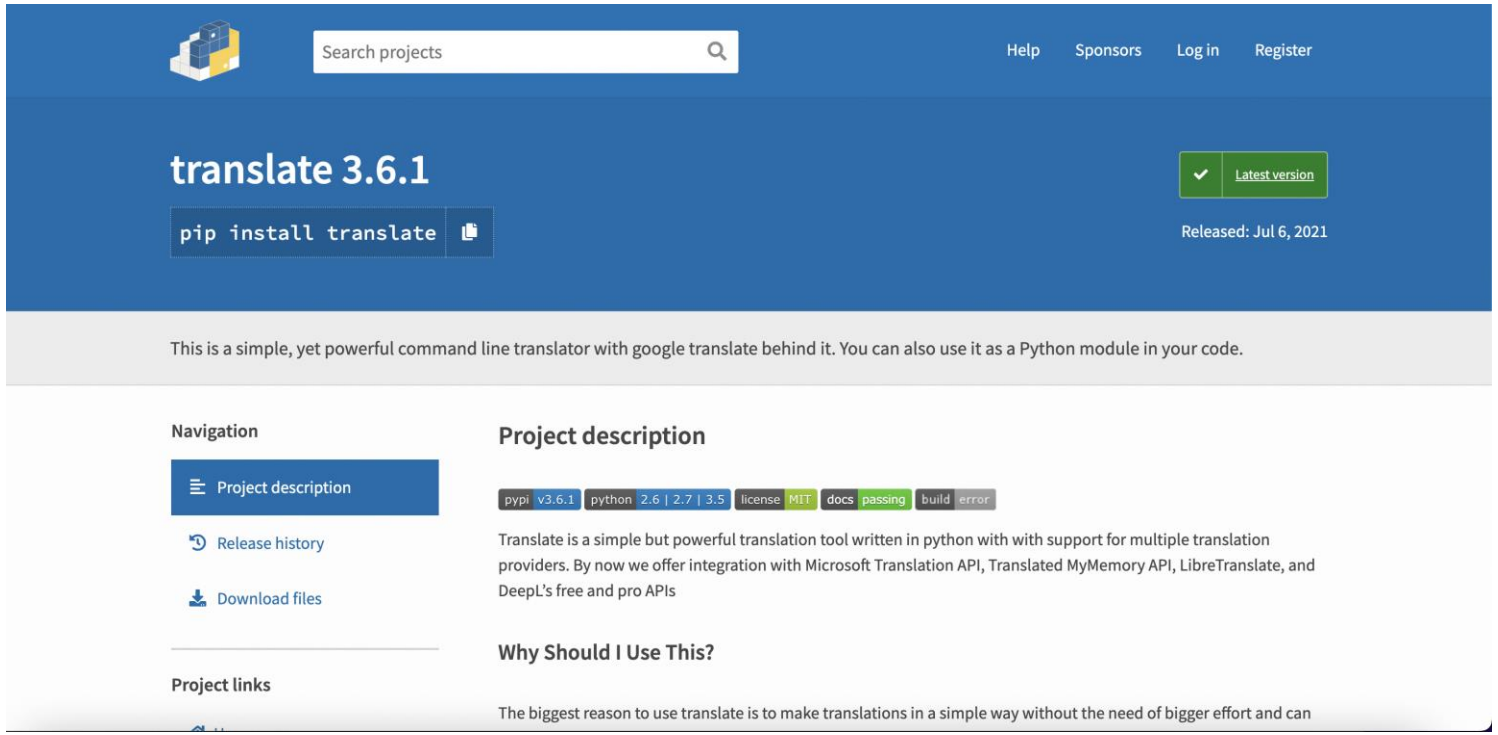
```
1 #3.Phân tích cảm xúc (Sentiment Analysis)
2 from underthesea import sentiment
3 sentiment("Sản phẩm mình đặt về không như quảng cáo")
```

```
'negative'
```

source: https://github.com/undertheseanlp/underthesea

Text Libraries

❖ translate Library



A simple yet powerful command line translator with google translate behind it

🔨 **Installation**:

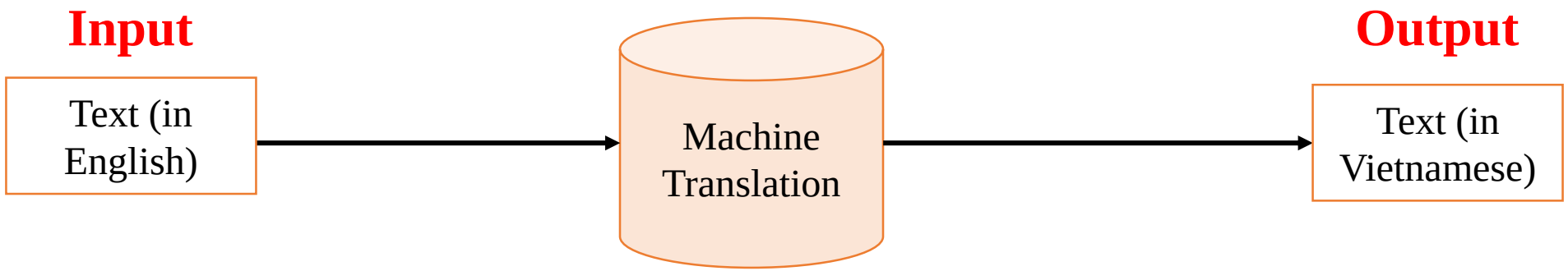
```
1 !pip install translate
```

source: https://pypi.org/project/translate/

Text Libraries

❖ translate Library

Machine Translation



😢 **Syntax: `translate.Translator.translate(X)`**

Input:

- *X*: raw sentence (unicode, str)

Output:

- *y: translated sentence*

```
#5.Dịch máy anh => việt (Machine Translate)
from translate import Translator

translator= Translator(to_lang="vi")
translation = translator.translate("Don't cry because it's over, smile because it happened.")
translation
```

```
'Đừng khóc vì nó đã kết thúc, hãy mỉm cười vì nó đã xảy ra.'
```

source: https://pypi.org/project/translate/

Text Libraries

❖ googletrans Library

A free and unlimited python library that implemented Google Translate API

🔨 **Installation**:

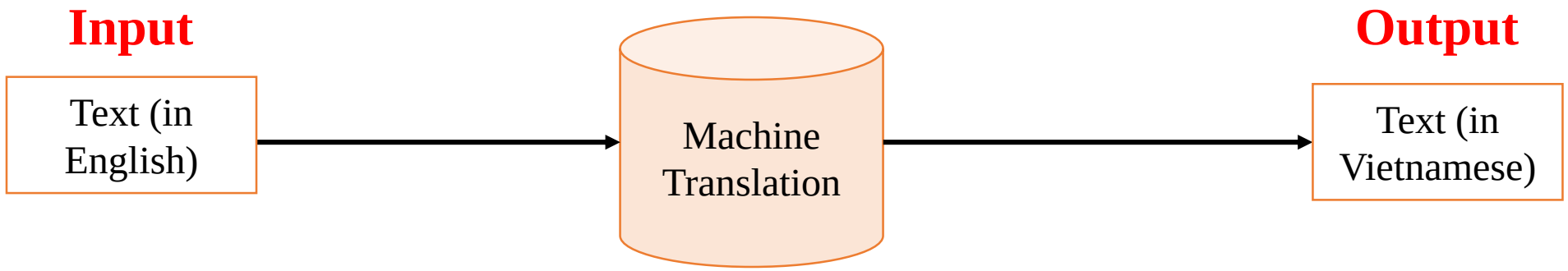
```
!pip install googletrans==4.0.0rc1
```

source: https://pypi.org/project/googletrans/

Text Libraries

❖ googletrans Library

Machine Translation



Syntax: `googletrans.Translator.translate(text)`

Input:

- *text*: raw text (unicode, str)

Output:

- <class 'googletrans.models.Translated'>

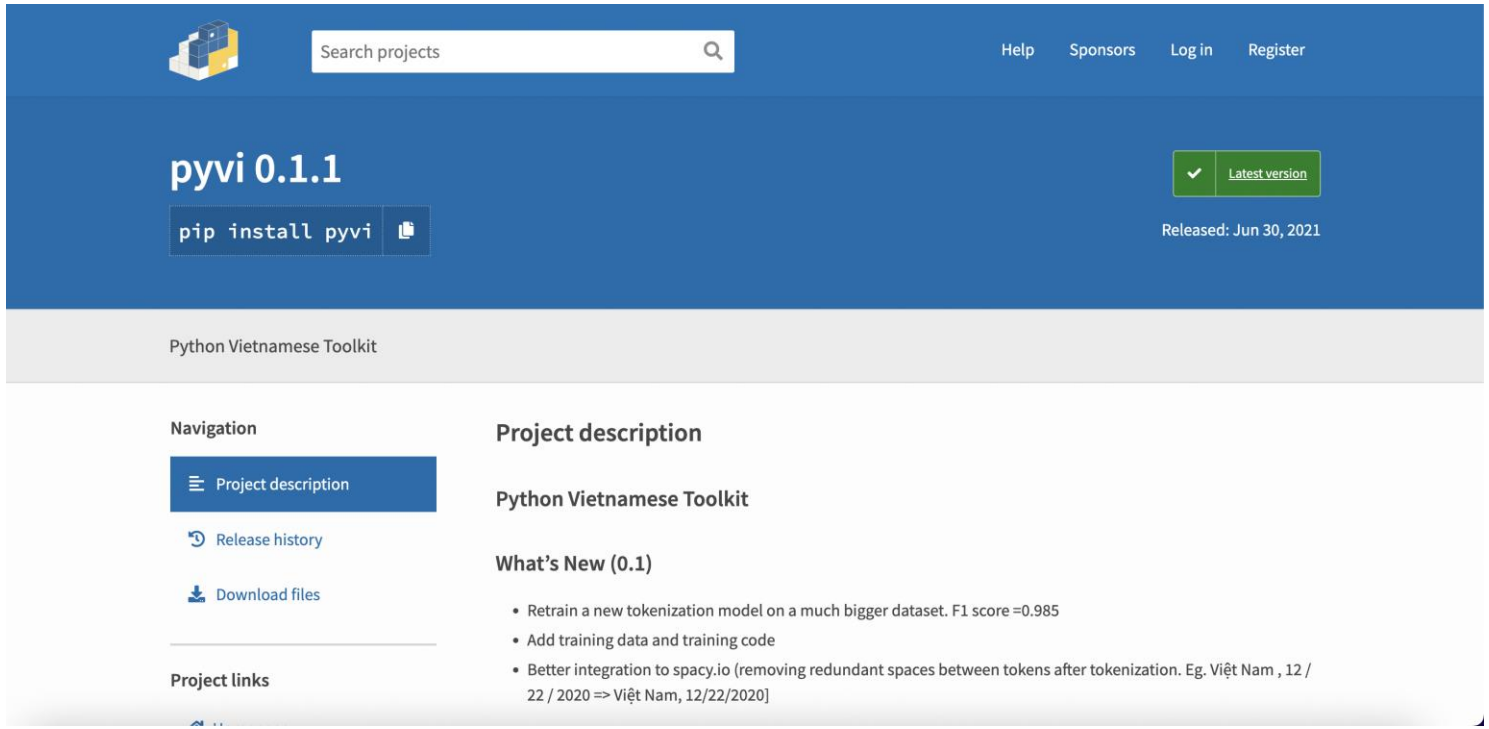
```
from googletrans import Translator
# define a translate object
translate = Translator()
# Translate some text
result = translate.translate('Chúng tôi là nhóm vinasupport')
print(result)
print(result.text)
```

```
Translated(src=vi, dest=en, text=We are Vinasupport, pronunciation=None, extra_data="{'confiden...")
We are Vinasupport
```

source: https://pypi.org/project/googletrans/

Text Libraries

❖ pyvi Library



Python Vietnamese Toolkit

🔨 **Installation**:

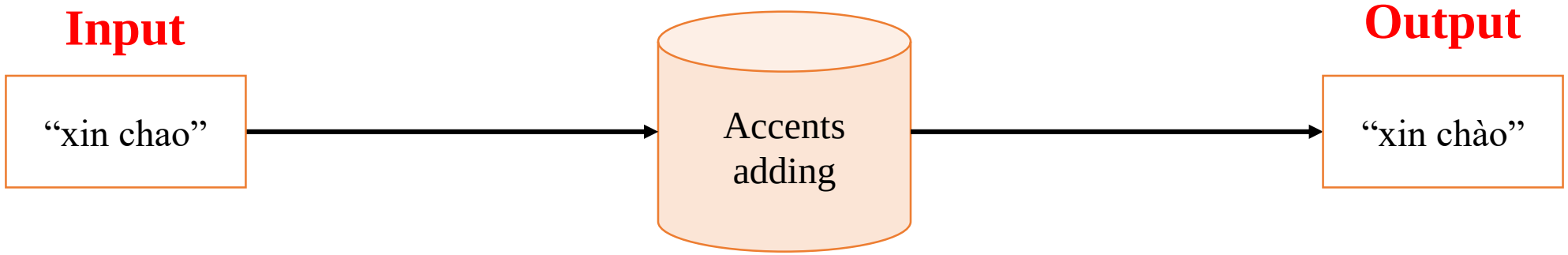
```
1 !pip install pyvi
```

source: https://pypi.org/project/pyvi/

Text Libraries

❖ pyvi Library

Accents adding



🥹 **Syntax:** `pyvi.ViUtils.add_accents(`***`text`***`)`

Input:

- *text*: un-accented text (unicode)

Output:

- Input text with accents

```
1 #4. Thêm dấu cho tiếng việt (Accents adding)
2 from pyvi import ViUtils
3 ViUtils.add_accents(u'truong dai hoc bach khoa ha noi')
```

```
'Trường Đại học Bách Khoa hà nội'
```

source: https://pypi.org/project/pyvi/

Text Libraries

❖ newspaper3k Library

Amazing Python library for extracting & curating articles

🔨 **Installation**:

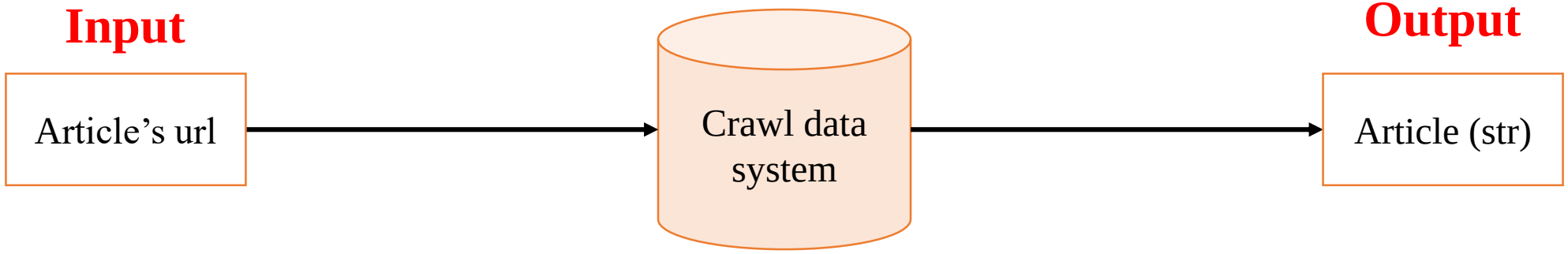
```
1 !pip install newspaper3k
```

source: https://newspaper.readthedocs.io/en/latest/

Text Libraries

❖ newspaper3k Library

Text crawler



😲 **Syntax: `newspaper.Article`**

Input:

- *url*: article's url (str)

Output:

- <class 'newspaper.article.Article'>

```
#6. Crawl văn bản
from newspaper import Article
url = "https://thanhnien.vn/tai-chinh-kinh-doanh/gia-vang-hom-nay-2692021-chuyen-gia-du-bao-se-tiep-tuc-giam-1454975.html"
article = Article(url)
article.download()
article.parse()
article.text.replace("\n", "")
```

```
'Sáng 26.9, giá vàng miếng chốt tuần tại Công ty vàng bạc đá quý Sài Gòn - SJC được mua vào là 56,35 triệu đồng/lượng và bán ra 57 triệu đồng/lượng. So với cuối tuần qua, mỗi lượng vàng miếng tăng 350.000 đồng ở cả hai chiều. Mức chênh lệch giữa giá mua và bán vàng miếng SJC lên 700.000 đồng/lượng thay vì mức 650.000 đồng vào cuối tuần qua.Trên thị trường thế giới, giá vàng chốt phiên cuối tuần ở mức 1.750,7 USD/ounce, giảm gần 4 USD so với cuối tuần trước. Vàng thế giới quy đổi theo tỷ giá của ngân hàng Vietcombank tương đương 48,43 triệu đồng/lượng (chưa bao gồm thuế, phí), giảm 100.000 đồng/lượng so với cuối tuần qua. Bất chấp vàng thế giới đi xuống thì trong tuần SJC lại tăng giá bán ra. Vì vậy, mỗi lượng vàng miếng SJC trong nước hiện cao hơn thế giới lên đến 8,7 triệu đồng.Kim loại quý đã giảm tuần thứ ba liên tiếp và vẫn đang giao dịch dưới ngưỡng 1.800 USD/ounce. Sau khi Cục Dự trữ Liên bang Mỹ (Fed) đưa ra quan điểm vẫn giữ lãi suất gần bằng 0% ở hiện tại và chưa công bố thời đ...'
```

source: https://newspaper.readthedocs.io/en/latest/

Text Libraries

❖ gensim Library

A Python library for topic modelling, document indexing and similarity retrieval with large corpora

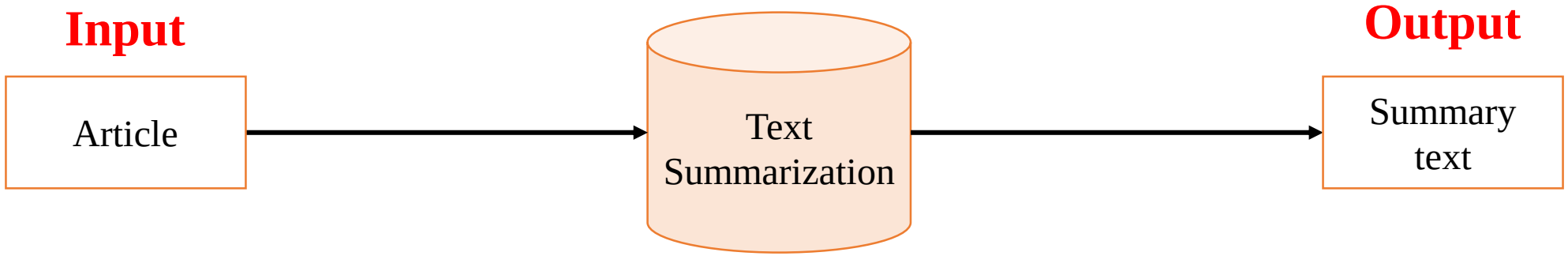
🔨 **Installation**:

```
!pip install gensim==3.6.0
```

source: https://radimrehurek.com/gensim/

Text Libraries

❖ gensim Library



🥹 **Syntax:** `gensim.summarization.summarize(text, ratio=0.2)` *(deprecated in gensim 4.x)*

Input:

- *text*: article text (str)
- ratio: the propotion of the number of sentences of the original text to be chosen for the summary

Output:

- Most representative sentences of the given text

```
#6.Tóm tắt văn bản (Text Summrization)
from gensim.summarization import summarize
short_summary = summarize(article.text.replace("\n", ""), ratio=0.2)
short_summary
```

```
'Vì vậy, mỗi lượng vàng miếng SJC trong nước hiện cao hơn thế giới lên đến 8,7 triệu đồng.Kim loại quý đã giảm tuần thứ ba liên tiếp và vẫn đang giao d
ịch dưới ngưỡng 1.800 USD/ounce.\nViệc ngân hàng trung ương nhiều nước sắp tới cũng dự kiến sẽ tăng lãi suất trở lại cũng là một yếu tố bất lợi cho giá
vàng.Cuộc khảo sát về giá vàng hằng tuần của Kitco News cho thấy nhiều chuyên gia tiếp tục bi quan trong tuần tới.'
```

source: https://radimrehurek.com/gensim/

Speech Libraries

❖ librosa Library

A python package for music and audio analysis

🔨 **Installation**:

```
1 !pip install librosa
```

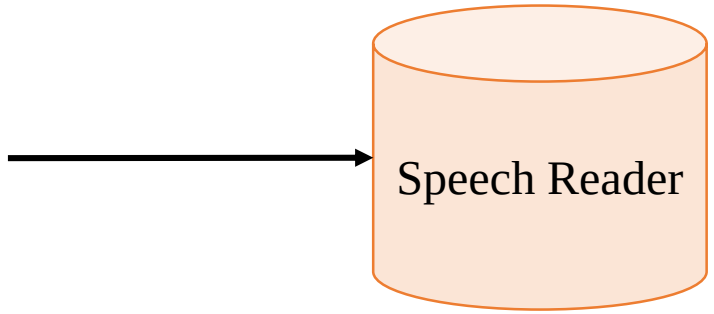
source: https://librosa.org/doc/latest/index.html

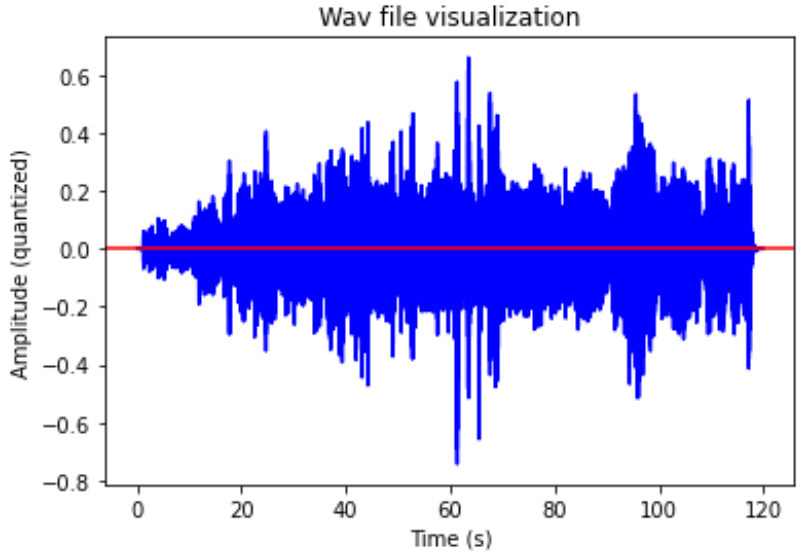
Text Libraries

❖ librosa Library: Usages

Sound reader

Input





😲 **Syntax:** `librosa.load(filename, sr=22050)`

Input:

- *filename:* filepath of .wav soundfile

Output:

- data: sound datapoints (in time-domain)
- sr: sampling rate of the sound

```
# 1. read .wav
import librosa
filename = librosa.example('nutcracker')
# Load the audio as a waveform `y`
# Store the sampling rate as `sr`
audio, sr = librosa.load(filename)
```

source: https://librosa.org/doc/main/generated/librosa.load.html

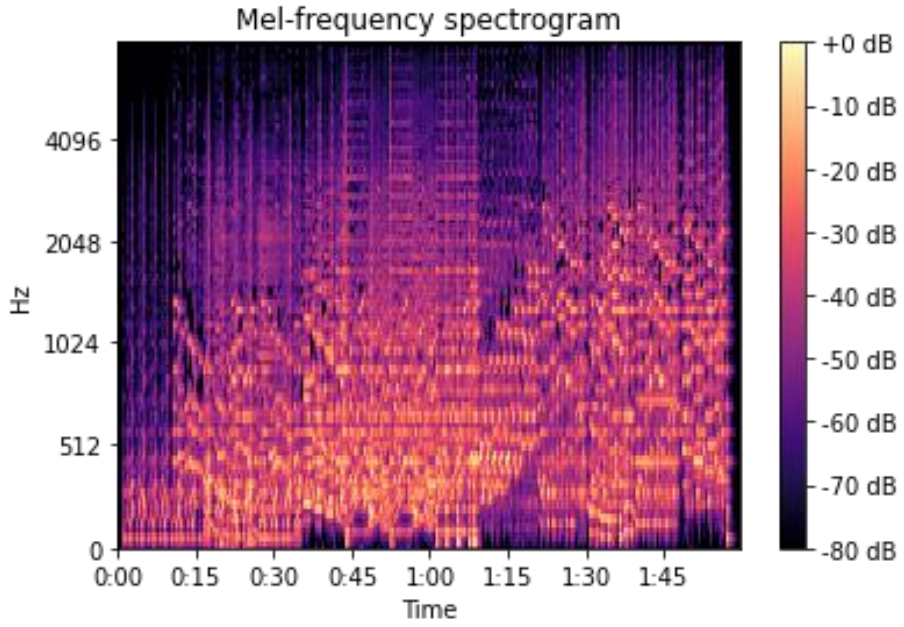
Text Libraries

❖ librosa Library: Usages

Mel spectrogram converter

Input

Audio data points, sampling rate → Mel spectrogram converter



Syntax: `librosa.feature.melspectrogram(y, sr)`

Input:

- *y:* audio datapoints
- *sr:* sampling rate of audio

Output:

- S: vector of mel spectrogram

```
# 2. mel spectrogram
import librosa
filename = librosa.example('nutcracker')
# Load the audio as a waveform `y`
# Store the sampling rate as `sr`
audio, sr = librosa.load(filename)
S = librosa.feature.melspectrogram(y=audio, sr=sr)
```

source: https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html

Speech Libraries

❖ scipy Library

A free and open-source Python library used scientific computing and technical computing

🔨 **Installation**:

```
1 !pip install scipy
```

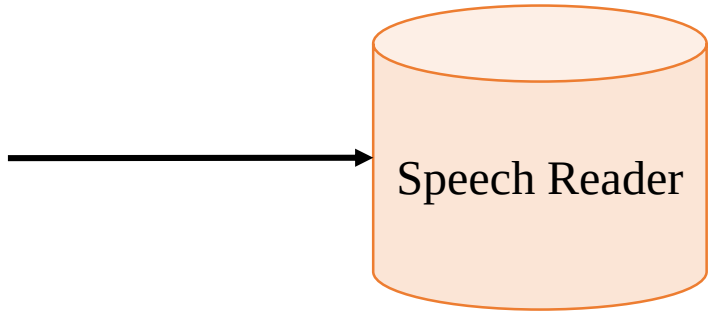
source: https://librosa.org/doc/latest/index.html

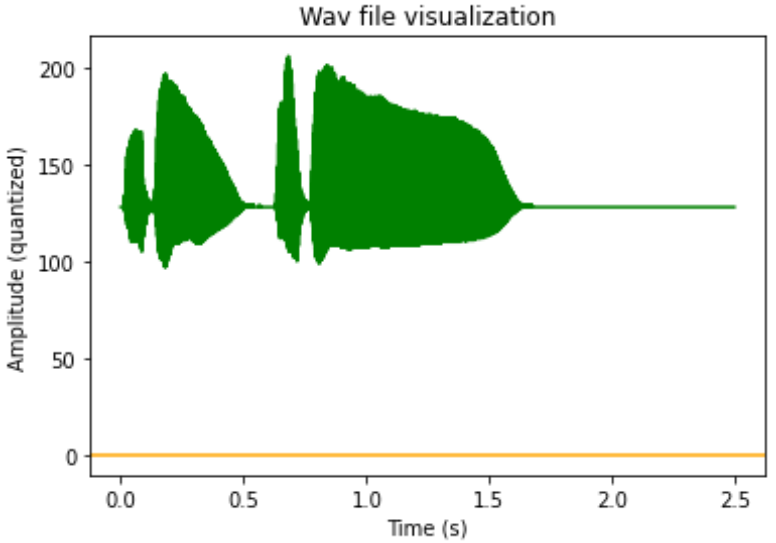
Text Libraries

❖ scipy Library

Sound reader

Input





😳 **Syntax: `scipy.io.wavfile(filename)`**

Input:

- *filename:* filepath of .wav soundfile

Output:

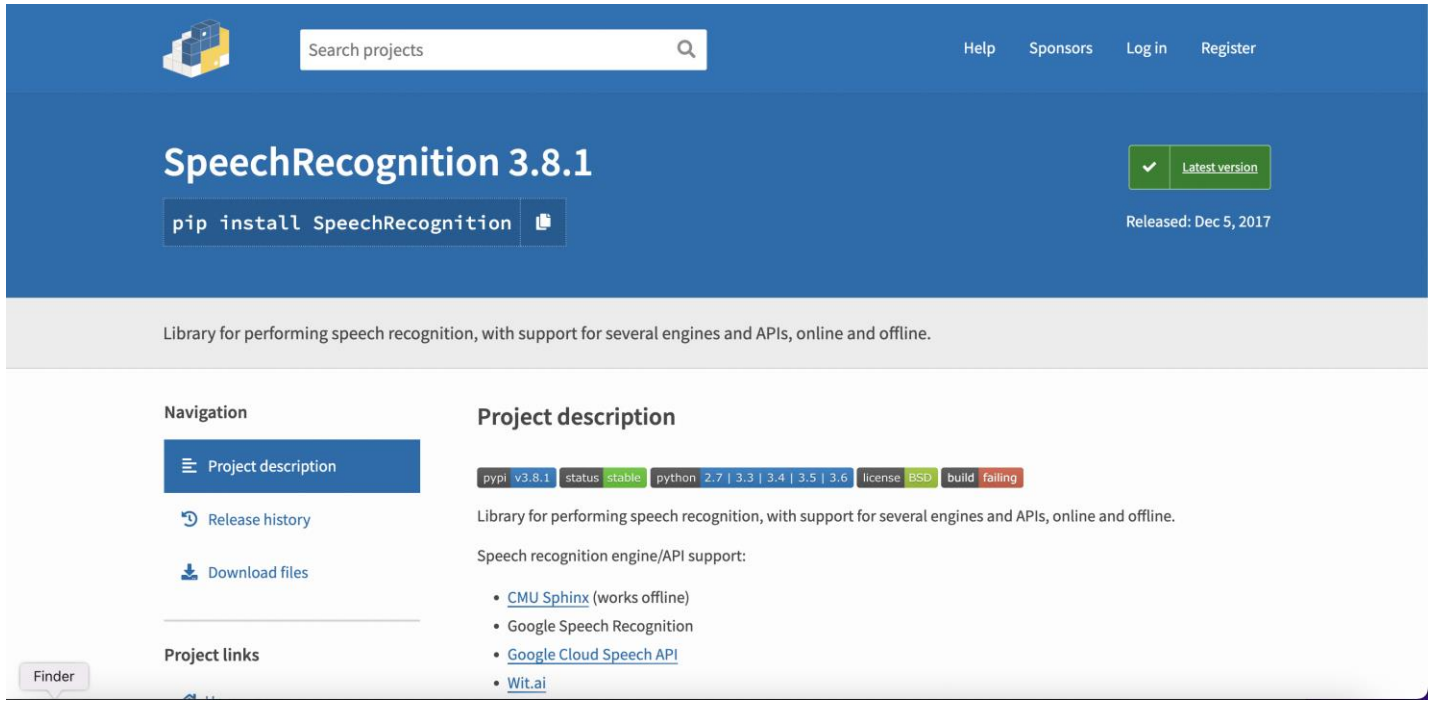
- sr: sampling rate of the sound
- data: sound datapoints (in time-domain)

```
# 3. read wav
from scipy.io import wavfile

wav_filepath = './51.wav'
sampling_rate, data = wavfile.read(wav_filepath)
```

source: https://librosa.org/doc/main/generated/librosa.load.html

Speech Libraries

❖ SpeechRecognition Library



Library for performing speech recognition, with support for several engines and APIs, online and offline

🔨 **Installation**:

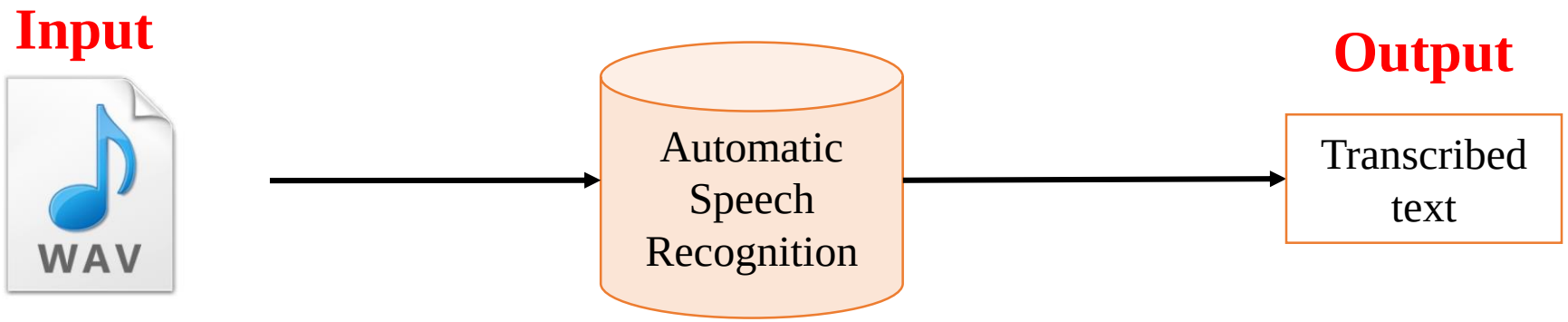
```
1 !pip install SpeechRecognition
```

source: https://pypi.org/project/SpeechRecognition/

Text Libraries

❖ SpeechRecognition Library

Automatic Speech Recognition



🥹 **Syntax:** `speech_recognition.Recognizer.recognize_google(audio, language)`

Input:

- *audio:* audio data (AudioData class)
- *language:* input audio language

Output:

- text: transcribed text (str)

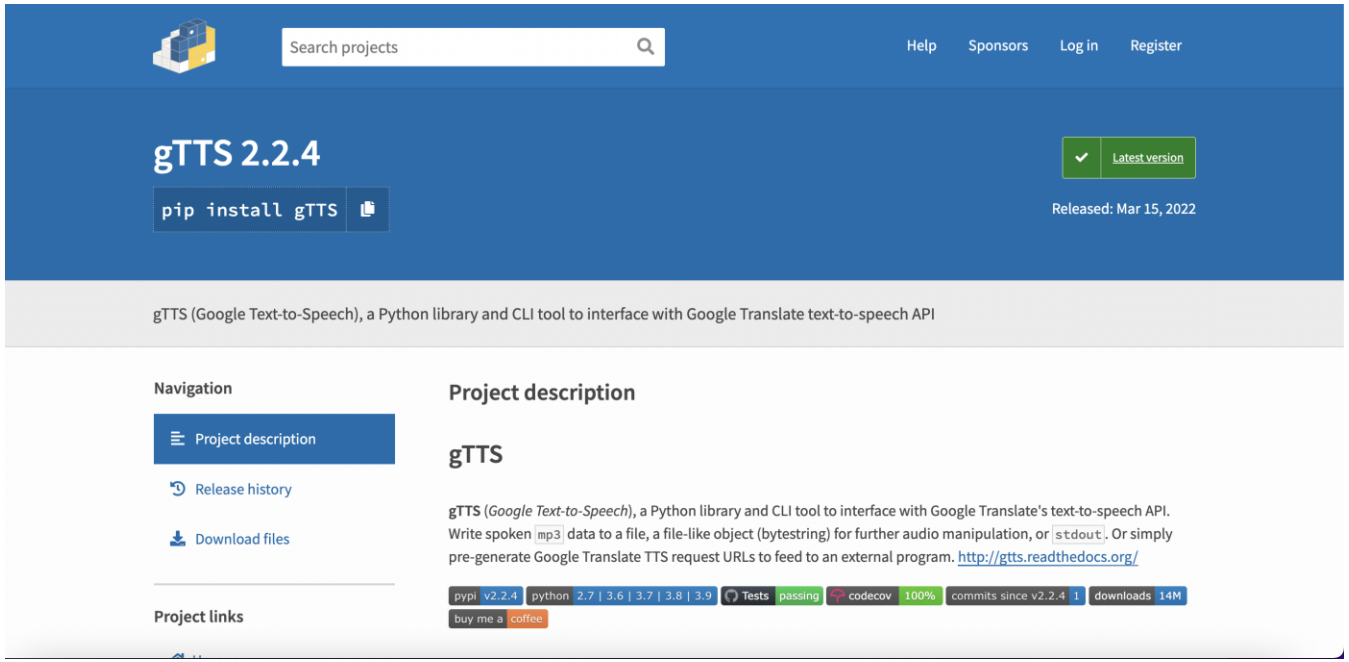
```
# 3. automatic speech recognition
import speech_recognition as sr

r = sr.Recognizer()
audio_filename = './vivos/train/waves/VIVOSSPK01/VIVOSSPK01_R002.wav'
my_audio = sr.AudioFile(audio_filename)
with my_audio as source:
  audio = r.record(source)

your_speech = r.recognize_google(audio, language="vi-VN")
print("Audio transcription: ", your_speech)
```

```
Audio transcription:  chị bằng cách luôn nỗ lực thì cuối cùng bạn mới được đền đáp
```

source: https://pypi.org/project/SpeechRecognition/

Speech Libraries

❖ gTTS Library



A Python library and CLI tool to interface with Google Translate's text-to-speech API.

🔨 **Installation**:

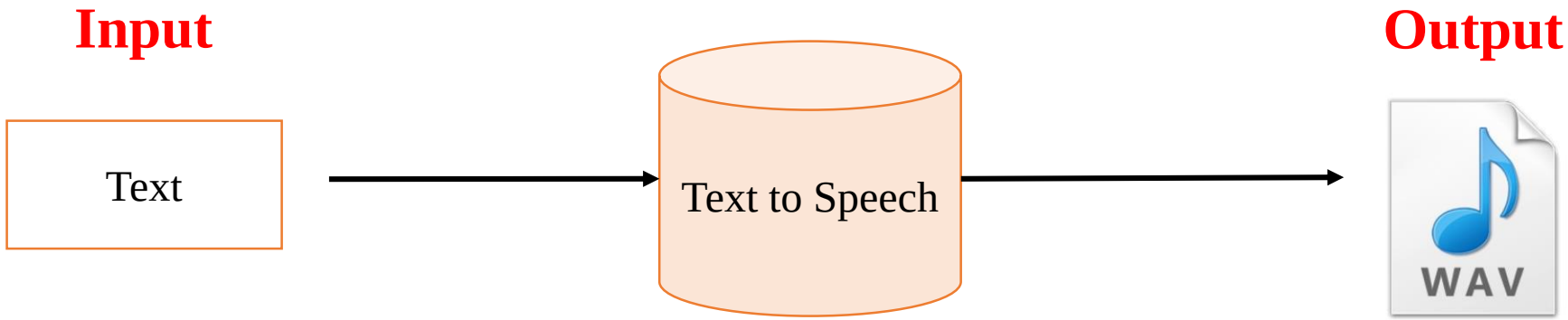
```
1 !pip install gtts
```

source: https://pypi.org/project/gTTS/

Text Libraries

❖ gTTS Library

Text to Speech



🥹 **Syntax: `gTTS(content, lang)`**

Input:

- *content:* a string
- *lang:* spoken language

Output: <class 'gtts.tts.gTTS'>

```
1 from gtts import gTTS
2 lang='vi'
3 output_filename = 'record.mp3'
4 content = "xin chào mọi người"
5 output = gTTS(content, lang=lang, slow=False) # text to speech
6
7 output.save(output_filename) # save google audio to a file
```

source: https://pypi.org/project/gTTS/