

Exercise

Data Structure

Quang-Vinh Dinh
Ph.D. in Computer Science

Outline

- **Common Errors and Bad Code**
- **Code Reading**
- **Dictionary**
- **List**

Common Errors

Error #1

```
3
4 def a_function(n):
5     return a_function(n)
6
7 a_function(5)
```

```
-----
RecursionError                                Traceback (most recent call last)
<ipython-input-10-bda7ef50bf68> in <module>
      5     return a_function(n)
      6
----> 7 a_function(5)

<ipython-input-10-bda7ef50bf68> in a_function(n)
      3
      4 def a_function(n):
----> 5     return a_function(n)
      6
      7 a_function(5)

... last 1 frames repeated, from the frame below ...

<ipython-input-10-bda7ef50bf68> in a_function(n)
      3
      4 def a_function(n):
----> 5     return a_function(n)
      6
      7 a_function(5)

RecursionError: maximum recursion depth exceeded
```

Common Errors

Error #2

```
1 # aivietnam - example
2
3 # create a list of numbers
4 numbers = [1, 2, 3, 4, 5]
5
6 for num in numbers:
7     # print the num
8     print(num)
9
10    # check if the num is equal to 3
11    if (num == 3):
12        # add 3 to the list
13        numbers.append(3)
```

↓

numbers = [1, 2, 3, 4, 5]

Output

1

↓

numbers = [1, 2, 3, 4, 5]

Output

1, 2

↓

numbers = [1, 2, 3, 4, 5]

Output

1, 2, 3

↓

numbers = [1, 2, 3, 4, 5, 3]

Common Errors

Error #2

```
1 # aivietnam - example
2
3 # create a list of numbers
4 numbers = [1, 2, 3, 4, 5]
5
6 for num in numbers:
7     # print the num
8     print(num)
9
10    # check if the num is equal to 3
11    if (num == 3):
12        # add 3 to the list
13        numbers.append(3)
```

↓

numbers = [1, 2, 3, 4, 5, 3]

numbers = [1, 2, 3, 4, 5, 3, 3]


Output

1, 2, 3, 4, 5, 3

Common Errors


Error #2

```
1 # aivietnam - example
2
3 # create a list of numbers
4 numbers = [1, 2, 3, 4, 5]
5
6 for num in numbers:
7     # print the num
8     print(num)
9
10    # check if the num is equal to 3
11    if (num == 3):
12        # add 3 to the list
13        numbers.append(3)
```


numbers = [1, 2, 3, 4, 5, 3]

Output

1, 2, 3, 4, 5, 3


numbers = [1, 2, 3, 4, 5, 3, 3]

Output

1, 2, 3, 4, 5, 3, 3

numbers = [1, 2, 3, 4, 5, 3, 3, 3]

o o o

Common Errors

Error #3

```
1 # common error
2
3 name = 'John'
4 age  = 26
5
6 print('Hello {name}. Are you {age}?')
```

Hello {name}. Are you {age}?

```
1 # common error
2
3 fruits = {'banana': 2}
4 fruits['apple'] += 10
```

```
-----
KeyError                                Traceback
<ipython-input-1-139246a7a818> in <module>
      2
      3 fruits = {'banana': 2}
----> 4 fruits['apple'] += 10

KeyError: 'apple'
```

Common Errors

Error #4

???

```
1 # common error
2
3 fruits = ['apple', 'banana', 'peach']
4
5 for fruit in fruits:
6     if (fruit == 'peach'):
7         fruits.append('peach')
8
9     print('Do something with this fruit -', fruit)
```

```
1 # common error
2
3 fruits = ['apple', 'banana', 'peach']
4
5 for fruit in fruits:
6     if (fruit == 'banana'):
7         fruits.remove('banana')
8
9     print('Do something with this fruit -', fruit)
```


Bad Codes

```
1 # bad code 1
2
3 fruits= [ 'apple' , "lemon", 'banana']
```

```
1 # bad code 2
2
3 x = 1
4 y = 2
5 z = 3
6
7 f1 = x + y - z
8 f2 = x/y + z
9 f3 = x - y*z
10 f4 = (x/y) * z
```

```
1 # bad code 3
2
3 weights_1 = [1.0, 2.5, 3.7]
4 weights_2 = [1.0, 2.5, 3.7]
5 weights_3 = [1.0 , 2.5 , 3.7]
```

```
1 # bad code 4
2
3 'aivietnam' . upper()
```

```
1 # bad code 5
2
3 print ('aivietnam')
```

```
1 # bad code 6
2
3 fruits = ['apple', 'lemon', 'banana']
4 apple = fruits [0]
5 sub_set = fruits [:2]
```

Bad Codes

```
1 # bad code 7
2
3 weights_1 = [1.0, 2.5, 3.7]
4 weights_2 = [ 1.0, 2.5, 3.7 ]
```

```
1 # bad code 8
2
3 print('AI VIETNAM') # a comment
4 print('AI VIETNAM') # a comment
```

```
1 # bad code 9
2
3 class AdamOptimizer:
4     def exampleMethod1():
5         # ...
6     def exampleMethod2():
7         #...
8 def train():
9     #...
```

```
1 # bad code 10
2
3 import math, os, sys
4
5 # -- or --
6
7 import math
8 import os
9 import sys
```

Bad Codes

```
1 # bad code 11 - style
2
3 fruits = ['apple', 'lemon', 'banana']
4 apple = fruits[0]
5 sub_set = fruit[:2]
6
7 # -- or --
8
9 fruits = ['apple', 'lemon', 'banana']
10 apple = fruits[0]
11 sub_set = fruit[:2]
```

```
1 # bad code 12
2
3 fruits = ['apple', 'lemon', 'banana']
4 number_of_fruits = len(fruits)
5 print(number_of_fruits)
6
7 len = 3
8 print(len(fruits))
```

```
1 # bad code 13 - magic number
2
3 radius = 4
4 area = radius*radius*3.14159
```

```
1 # bad code 14 - one entry, one exit
2
3 def ReLU(number):
4     if (number <= 0):
5         return 0
6     else:
7         return number
8
9 print(ReLU(-8))
```

Bad Codes

```
1  # bad code 15 - dead code
2
3  def ReLU(number):
4      if (number <= 0):
5          return 0
6      else:
7          return number
8
9      return 'Input is not a number!'
10
11 print(ReLU(-8))
```

```
1  # bad code 16 - comment
2
3  def flip(times):
4      number_of_heads = 0
5      number_of_tails = 0
6
7      for _ in range(times):
8          number = random.randint(0, 1)
9          if (number == 1):
10             number_of_heads = number_of_heads + 1
11          else:
12             number_of_tails = number_of_tails + 1
13
14      return number_of_heads, number_of_tails
15
16 number_of_heads, number_of_tails = flip(1000)
17 print(number_of_heads)
18 print(number_of_tails)
```

Bad Codes

```
1  # bad code 17 - global
2
3  number_of_heads = 0
4  number_of_tails = 0
5
6  def flip(times):
7      global number_of_heads
8      global number_of_tails
9
10     for _ in range(times):
11         number = random.randint(0, 1)
12         if (number == 1):
13             number_of_heads = number_of_heads + 1
14         else:
15             number_of_tails = number_of_tails + 1
16
17 flip(1000)
18 print(number_of_heads)
19 print(number_of_tails)
```

```
1  # bad code 18 - make thing complicated
2
3  import random
4  class Dice:
5      def __init__(self, sides=6):
6          self.sides = sides
7      def roll(self):
8          return random.randint(1, self.sides)
9
10 d = Dice()
11 print('You rolled a', d.roll())
```

Bad Codes

```
1 # bad code 19
2
3 numbers = []
4
5 for i in range(1, 100):
6     if (i%5 == 0):
7         numbers.append(i)
8
9 print(numbers)
```

```
1 numbers = [i for i in range(1, 100) if (i%5 == 0)]
2 print(numbers)
```

```
1 # bad code 21
2
3 # open a file
4 a_file = open('hello_world.txt', 'w')
5
6 # write data to file
7 text3 = 'writing line \n'
8 a_file.write(text3)
```

```
1 # bad code 20
2
3 try:
4     num = input('Enter a number: ')
5     num = int(num)
6 except ValueError:
7     pass # do nothing
```

Bad Codes

```
1  # bad code 22 - Unpythonic
2
3  path = 'E:\Data\AICourse-2021\1.BasicPython\file\hello_world.txt'
4  print(path)
5
6  with open(path, 'r') as file:
7      lines = file.readlines()
8      print(lines)
```

```
1  path = 'E:\\Data\\AICourse-2021\\1.BasicPython\\file\\hello_world.txt'
2  print(path)
3
4  with open(path, 'r') as file:
5      lines = file.readlines()
6      print(lines)
```

```
1  path = r'E:\Data\AICourse-2021\1.BasicPython\file\hello_world.txt'
2  print(path)
3
4  with open(path, 'r') as file:
5      lines = file.readlines()
6      print(lines)
```

Bad Codes

```
1 # bad code 23
2
3 name = 'John'
4 age  = 26
5
6 print('Hello ' + name + '. Are you ' + str(age) + '?')
```

```
1 name = 'John'
2 age  = 26
3
4 print(f'Hello {name}. Are you {age}?')
```

```
1 # bad code 24 - Unpythonic
2
3 fruits = {'banana': 2}
4 if 'apple' not in fruits:
5     fruits['apple'] = 0
6
7 fruits['apple'] += 10
8 print(fruits)
```

```
1 fruits = {'banana': 2}
2 fruits.setdefault('apple', 0)
3
4 fruits['apple'] += 10
5 print(fruits)
```

```
1 # bad code 25
2
3 def get_salary_rate(employee_class):
4     if employee_class == 'level-1':
5         result = 5.7
6     elif employee_class == 'level-2':
7         result = 4.2
8     elif employee_class == 'level-3':
9         result = 3.8
10    elif employee_class == 'level-4':
11        result = 3.3
12    else:
13        result = 2.9
14
15    return result
16
17 print(get_salary_rate('level-1'))
18 print(get_salary_rate('level-4'))
```

```
1 salary_rates = {'level-1': 5.7,
2                 'level-2': 4.2,
3                 'level-3': 3.8,
4                 'level-4': 3.3,
5                 'level-5': 2.9}
6
7 print(salary_rates['level-1'])
8 print(salary_rates['level-4'])
```


Bad Codes

```
1 # bad code 26 - Unpythonic
2
3 def a_function(value):
4     # do something
5
6     if 1 < value and value < 10:
7         print('code inside if')
8     else:
9         print('code inside else')
10
11     # do something and return something
12
13 a_function(4)
14 a_function(40)
```

```
1 def a_function(value):
2
3     # do something
4
5     if 1 < value < 10:
6         print('code inside if')
7     else:
8         print('code inside else')
9
10     # do something and return something
11
12 a_function(4)
13 a_function(40)
```

```
1 # bad code 27
2
3 def contain_fruit(a_fruit):
4     if (a_fruit == 'banana'):
5         result = True
6     elif (a_fruit == 'apple'):
7         result = True
8     elif (a_fruit == 'peach'):
9         result = True
10    else:
11        result = False
12
13    return result
14
15 print(contain_fruit('banana'))
16 print(contain_fruit('pineapple'))
```

```
1 def contain_fruit(a_fruit):
2     result = a_fruit in ('banana',
3                           'apple',
4                           'peach')
5
6     return result
7
8 print(contain_fruit('banana'))
9 print(contain_fruit('pineapple'))
```

Bad Codes

```
1  # super bad code 28
2
3  value = True + True + False + True
4  char1 = 'aivietnam'[False]
5  char2 = 'aivietnam'[True]
6  char3 = 'aivietnam'[-True]
7
8  print(value)
9  print(char1)
10 print(char2)
11 print(char3)
```

```
1  # bad code 29
2
3  def add_fruit(a_fruit, fruits=['apple']):
4      fruits.append(a_fruit)
5
6      return fruits
```

```
1  fruits = add_fruit('banana')
2  print(fruits)
3
4  fruits = add_fruit('banana')
5  print(fruits)
```

['apple', 'banana']

['apple', 'banana', 'banana']

Bad Codes

```
1  # bad code 30
2
3  import math
4
5  def quadratic_equation(a, b, c):
6      # compute delta
7      delta = b*b - 4*a*c
8
9      if delta < 0:
10         result = 'The equation has no solution'
11     elif delta == 0:
12         x = (-b+math.sqrt(delta))/2*a
13         result = (x,)
14     else:
15         x1 = (-b+math.sqrt(delta))/(2*a)
16         x2 = (-b-math.sqrt(delta))/(2*a)
17         result = (x1,x2)
18
19     return result
20
21 print(quadratic_equation(3, 2, 1))
22 print(quadratic_equation(1, 2, 1))
```

Good Code

```
1  # swap
2
3  x, y = 3, 4
4  print(x, y)
5
6  x, y = y, x
7  print(x, y)
```

3 4
4 3

```
1  # condition
2
3  n = 8
4  result = 1 < n < 10
5  print(result)
```

True

```
1  # reverse a string
2
3  name = "ai vietname"
4  print(name)
5  print(name[::-1])
```

ai vietname
emanteiv ia

```
1  # join
2
3  a = ["Hi", "AI", "VIETNAM"]
4  print(" ".join(a))
```

Hi AI VIETNAM

Good Code

```
1  # unpacking
2
3  a_list = [1, 2, 3]
4  x, y, z = a_list
5
6  print(x, y, z)
```

1 2 3

```
1  # check if contained
2  m = 1
3
4  if m in [1, 3, 5, 7]:
5      print('Contained!')
```

Contained!

```
1  # enumerate
2
3  a_list = [4, 5, 6]
4  for i, value in enumerate(a_list):
5      print(i, ': ', value)
```

0 : 4
1 : 5
2 : 6

Good Code

```
1  # Unpacking operator
2
3  def print_data(x, y, z):
4      print(x, y, z)
5
6  a_dict = {'x': 1, 'y': 2, 'z': 3}
7  a_list = [3, 4, 5]
8
9  print_data(*a_dict)
10 print_data(**a_dict)
11 print_data(*a_list)
```

```
x y z
1 2 3
3 4 5
```

```
1  # Use a dictionary to store a switch
2
3  ops = {
4      'addition': lambda x, y: x + y,
5      'subtraction': lambda x, y: x - y,
6      'multiplication': lambda x, y: x * y,
7      'division': lambda x, y: x / y
8  }
9
10 print(ops['addition'](4, 6))
11 print(ops['multiplication'](4, 6))
```

```
10
24
```

Good Code

```
1 # get the most frequent element
2
3 test = [1, 2, 3, 4, 2]
4
5 print(max(test))
6 print(max(test, key=test.count))
```

4

2

```
1 # create dict from two tuples
2
3 t1 = (1, 2, 3)
4 t2 = (10, 20, 30)
5
6 a_dict = dict(zip(t1,t2))
7 print(a_dict)
```

{1: 10, 2: 20, 3: 30}

Outline

- **Common Errors and Bad Code**
- **Code Reading**
- **Dictionary**
- **List**

Code Reading Skill

Example 1

```
1  n = 1
2  for i in range(0, 500, 100):
3      n = i
4  print(n)
```

Example 2

```
1  data = "I'm learning Python!"
2  print(data.split()[1])
```

Example 3

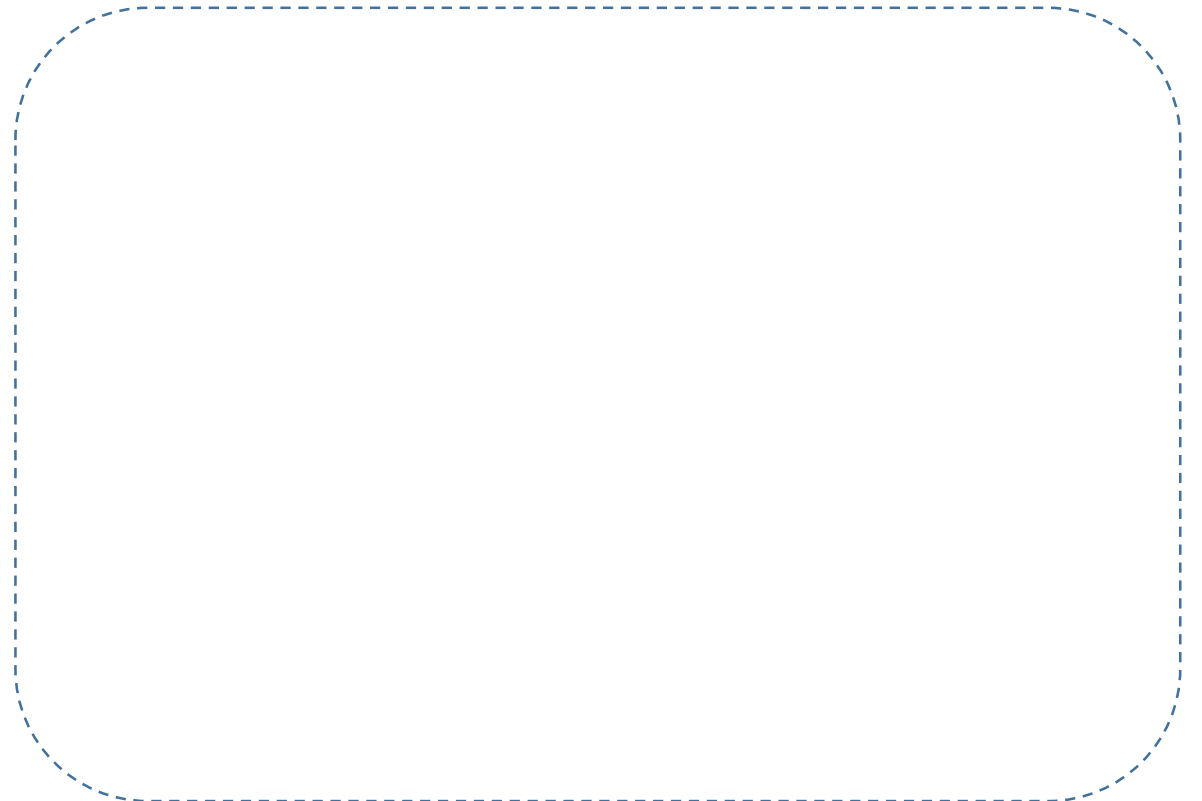
```
1  n = 0
2  for i in range(5):
3      n += i
4      if n > 0 and n % 3 == 0:
5          break
6  print(n)
```

Example 4

```
1 space1 = "Cherry Blossom After Winter"
2 space2 = "Flowers are blooming on the hillsides, which signals the coming of spring"
3 space = space1 + space2
4 print(space[-6:])
```

Example 5

```
1 my_list = [0, 1, 1, 2, 1]
2 odd = 0
3 even = 0
4
5 for number in my_list:
6     if number % 2 == 0:
7         odd += number
8     else:
9         even += number
10 print(f"odd:{odd}, even:{even}")
```



Code Reading Skill

```
1 weather = '@the drizzle in spring makes the air more humid. @@'  
2 me = "i love iT!"  
3  
4 txt = weather.strip('@').capitalize()+ me.title()  
5 print(txt)
```

Example 5

```
1 my_string = "Peach blossoms bloom in spring"  
2  
3 my_bag_of_word = []  
4 for element in my_string:  
5     if element not in my_bag_of_word:  
6         my_bag_of_word.append(element)  
7 print(my_bag_of_word)
```

Example 6

Code Reading Skill

Example 7

```
1 selected_numbers = [x for x in range(1,5) if x % 2==0]
2 print(selected_numbers)
```

Example 8

```
1 X = [[1, 1],
2       [2, 2]]
3
4 result = [[0,0],
5            [0,0]]
6
7 for i in range(len(X)):
8     for j in range(len(X[0])):
9         result[j][i] = X[i][j]
```

Code Reading Skill

Example 9

```
1  def my_function(data, max_value, min_value):
2      result = []
3      for i in data:
4          if i < min_value:
5              result.append(min_value)
6          elif i > max_value:
7              result.append(max_value)
8          else:
9              result.append(i)
10     return result
11
12 data = [10, 2, 5, 0, 1]
13 max_value = 2
14 min_value = 1
15 print(my_function(data, max_value, min_value))
```

Code Reading Skill

Example 10

```
1 def my_function(signal):
2     var = True
3     while var:
4         var = False
5         for i in range(len(signal) - 1):
6             if signal[i] > signal[i + 1]:
7                 signal[i], signal[i + 1] = signal[i + 1], signal[i]
8                 var = True
9
10 my_signal = [1, 2, 0]
11 my_function(my_signal)
12 print(my_signal)
```

Outline

- **Common Errors and Bad Code**
- **Code Reading**
- **Dictionary**
- **List**

1. Thực hiện theo các yêu cầu sau .

- (a) Viết function trả về một dictionary **đếm số lượng chữ xuất hiện trong một từ**, với key là chữ cái và value là số lần xuất hiện

Input: một từ

Output: dictionary đếm số lần các chữ xuất hiện

Note: Giả sử các từ nhập vào đều có các chữ cái thuộc [a-z] hoặc [A-Z]

```
1 # Examples 1(a)
2 string = 'Happiness'
3 count_chars(string)
4 >> {'H': 1, 'a': 1, 'e': 1, 'i': 1, 'n': 1, 'p': 2, 's': 2}
5
6 string = 'smiles'
7 count_chars(string)
8 >> {'e': 1, 'i': 1, 'l': 1, 'm': 1, 's': 2}
```


Viết function trả về một dictionary, đếm số lượng chữ xuất hiện trong một từ

```
1 # Examples 1(a)
2 string = 'Happiness'
3 count_chars(string)
4 >> {'H': 1, 'a': 1, 'e': 1, 'i': 1, 'n': 1, 'p': 2, 's': 2}
5
6 string = 'smiles'
7 count_chars(string)
8 >> {'e': 1, 'i': 1, 'l': 1, 'm': 1, 's': 2}
```

Dùng vòng lặp duyệt
từng chữ trong từ

Khởi tạo dictionary rỗng

Nếu chữ xuất hiện lần đầu lấy
chữ đó làm key với value = 1

Nếu chữ đã xuất hiện tìm trong
dictionary và tăng value lên 1

Hints

```
1 # iterate characters in a word
2 word = 'Sun'
3 for character in word:
4     print(character)
```

S
u
n

```
1 # create an empty dictionary
2
3 character_statistic = {}
4 print(character_statistic)
5 print(type(character_statistic))
```

{}
<class 'dict'>

```
1 # check if a key is in a dictionary
2
3 is_contained_1 = 'a' in character_statistic
4 print(is_contained_1)
5
6 is_contained_2 = 'b' in character_statistic
7 print(is_contained_2)
```

True
False

```
1 # insert a pair (key, value)
2
3 character_statistic['a'] = 1
4 print(character_statistic)
5
6 character_statistic['a'] = 5
7 print(character_statistic)
```

{'a': 1}
{'a': 5}

Giving the First Try

Examples

Problem???

```
1 # give it a try
2 character_statistic = {}
3
4 word = 'Happiness'
5 for character in word:
6     character_statistic[character] = 1
7
8 # print
9 print(character_statistic)
```

```
{'H': 1, 'a': 1, 'p': 1, 'i': 1, 'n': 1, 'e': 1, 's': 1}
```

```
1 # give it a try
2 character_statistic = {}
3
4 word = 'Baby'
5 for character in word:
6     character_statistic[character] = 1
7
8 # print
9 print(character_statistic)
```

```
{'B': 1, 'a': 1, 'b': 1, 'y': 1}
```

Problem-Solving (1)

Lowercase input_string

Problem???

```
1 # lowercase a string
2 word = 'Baby'
3 print(word)
4
5 word_lowercase = word.lower()
6 print(word_lowercase)
```

Baby
baby

```
1 # solve problem 1
2 character_statistic = {}
3
4 word = 'Baby'
5 word = word.lower()
6 for character in word:
7     character_statistic[character] = 1
8
9 # print
10 print(character_statistic)
```

{'b': 1, 'a': 1, 'y': 1}

Problem-Solving (2)

Check if a key already exists in a dictionary

```
1 # check if a key is in a dictionary
2
3 is_contained_1 = 'a' in character_statistic
4 print(is_contained_1)
5
6 is_contained_2 = 'b' in character_statistic
7 print(is_contained_2)
```

True
False

```
1 # solve problem 2
2 character_statistic = {}
3
4 word = 'Baby'
5 word = word.lower()
6 for character in word:
7     if character in character_statistic:
8         character_statistic[character] += 1
9     else:
10         character_statistic[character] = 1
11
12 # print
13 print(character_statistic)
```

{'b': 2, 'a': 1, 'y': 1}

Viết function trả về một dictionary, đếm số lượng chữ xuất hiện trong một từ

Dùng vòng lặp duyệt
từng chữ trong từ

Khởi tạo dictionary rỗng

Nếu chữ xuất hiện lần đầu lấy
chữ đó làm key với value = 1

Nếu chữ đã xuất hiện tìm trong
dictionary và tăng value lên 1

```
FUNCTION count_chars(inp_string)
    result = dict()
    FOR start at first char TO last char
        IF char in result
            result[char] += 1
        ELSE
            result[char] = 1
        ENDIF
    ENDFOR

    RETURN result
ENDFUNCTION
```


1. Thực hiện theo các yêu cầu sau .

(b) Viết function **đọc các câu trong một file txt, đếm số lượng các từ xuất hiện** và trả về một dictionary với key là từ và value là số lần từ đó xuất hiện.

- Input: Đường dẫn đến file txt
- Output: dictionary đếm số lần các từ xuất hiện
- Note:
 - Giả sử các từ trong file txt đều có các chữ cái thuộc [a-z] hoặc [A-Z]
 - Không cần các thao tác xử lý string phức tạp nhưng cần xử lý các từ đều là viết thường
 - Các bạn dùng lệnh này để download và tham khảo Code Listing 2
!gdown <https://drive.google.com/uc?id=1IBScGdW2xlNsc9v5zSAya548kNgiOrko>

```
1 # Examples 1(b)
2 !gdown https://drive.google.com/uc?id=1IBScGdW2xlNsc9v5zSAya548kNgiOrko
3 file_path = '/content/P1_data.txt'
4 word_count(file_path)
5 >>{'a': 7,
6    'again': 1,
7    'and': 1,
8    'are': 1,
9    'at': 1,
10   'be': 1,
11   'become': 2,
12   ...}
```

Code Listing 2: Đây là các ví dụ các bạn không cần thiết đặt tên giống ví dụ

Viết function trả về một dictionary, đếm số lượng từ xuất hiện trong một câu

Dùng vòng lặp duyệt
từng **chữ** trong **từ**

Khởi tạo dictionary rỗng

Nếu **chữ** xuất hiện lần đầu, lấy
chữ đó làm key với value = 1

Nếu **chữ** đã xuất hiện, tìm trong
dictionary và tăng value lên 1

From the previous exercise



Dùng vòng lặp duyệt
từng **từ** trong **câu**

Khởi tạo dictionary rỗng

Nếu **từ** xuất hiện lần đầu, lấy
từ đó làm key với value = 1

Nếu **từ** đã xuất hiện, tìm trong
dictionary và tăng value lên 1

For this exercise

Viết function trả về một dictionary, đếm số lượng từ xuất hiện trong một câu

Dùng vòng lặp duyệt từng từ trong câu

Khởi tạo dictionary rỗng

Nếu từ xuất hiện lần đầu, lấy từ đó làm key với value = 1

Nếu từ đã xuất hiện, tìm trong dictionary và tăng value lên 1

```
1 sentence = 'AI is a topic that is very interesting'
2 print(sentence)
3
4 sentence = sentence.lower()
5 print(sentence)
6
7 sentence = sentence.split()
8 print(sentence)
```

```
AI is a topic that is very interesting
ai is a topic that is very interesting
['ai', 'is', 'a', 'topic', 'that', 'is', 'very', 'interesting']
```

Viết function trả về một dictionary, đếm số lượng từ xuất hiện trong một câu

Dùng vòng lặp duyệt từng từ trong câu

Khởi tạo dictionary rỗng

Nếu từ xuất hiện lần đầu, lấy từ đó làm key với value = 1

Nếu từ đã xuất hiện, tìm trong dictionary và tăng value lên 1

```
1 # statistic words in a sentence
2 word_statistic = {}
3
4 sentence = 'AI is a topic that is very interesting'
5 sentence = sentence.lower()
6 sentence = sentence.split()
7
8 for word in sentence:
9     if word in word_statistic:
10         word_statistic[word] += 1
11     else:
12         word_statistic[word] = 1
13
14 # print
15 print(word_statistic)
```

{ 'ai': 1, 'is': 2, 'a': 1, 'topic': 1, 'that': 1, 'very': 1, 'interesting': 1 }

Viết function trả về một dictionary, đếm số lượng từ xuất hiện trong các câu

Problem???

```
1  # statistic words in a sentence
2  word_statistic = {}
3
4  sentence = 'AI is a topic that is very interesting. However, it is not easy.'
5  sentence = sentence.lower()
6  sentence = sentence.split()
7
8  for word in sentence:
9      if word in word_statistic:
10         word_statistic[word] += 1
11     else:
12         word_statistic[word] = 1
13
14  # print
15  print(word_statistic)
```

```
{'ai': 1, 'is': 3, 'a': 1, 'topic': 1, 'that': 1, 'very': 1, 'interesting.': 1, 'however.': 1, 'it': 1, 'not': 1, 'easy.': 1}
```

Viết function trả về một dictionary, đếm số lượng từ xuất hiện trong các câu

```
1 sentence = 'AI is a topic that is very interesting. However, it is not easy.'
2 print(sentence)
3
4 sentence = sentence.lower()
5 print(sentence)
6
7 sentence = sentence.replace('.', '')
8 sentence = sentence.replace(',', '')
9 print(sentence)
10
11 sentence = sentence.split()
12 print(sentence)
```

AI is a topic that is very interesting. However, it is not easy.

ai is a topic that is very interesting. however, it is not easy.

ai is a topic that is very interesting however it is not easy

['ai', 'is', 'a', 'topic', 'that', 'is', 'very', 'interesting', 'however', 'it', 'is', 'not', 'easy']

Viết function trả về một dictionary, đếm số lượng từ xuất hiện trong các câu

```
1  # solution
2  word_statistic = {}
3
4  sentence = 'AI is a topic that is very interesting. However, it is not easy.'
5  sentence = sentence.lower()
6  sentence = sentence.replace('.', '').replace(',', '')
7  sentence = sentence.split()
8
9  for word in sentence:
10     if word in word_statistic:
11         word_statistic[word] += 1
12     else:
13         word_statistic[word] = 1
14
15  # print
16  print(word_statistic)
```

```
{'ai': 1, 'is': 3, 'a': 1, 'topic': 1, 'that': 1, 'very': 1, 'interesting': 1, 'however': 1, 'it': 1, 'not': 1, 'easy': 1}
```

Outline

- **Common Errors and Bad Code**
- **Code Reading**
- **Dictionary**
- **List**

Getting Max over Kernel

❖ Introduction

Problem 01: Cho một list các số nguyên *num_list* và một sliding window (các bạn có thể tạm hiểu sliding window như là một list có kích thước nhỏ hơn *num_list*) có kích thước size *k* di chuyển từ trái sang phải. Mỗi lần dịch chuyển 1 vị trí sang phải có thể nhìn thấy được *k* số trong *num_list* và tìm số lớn nhất trong *k* số này sau mỗi lần trượt. *k* phải lớn hơn hoặc bằng 1. Các bạn hãy viết chương trình Python giải quyết vấn đề trên.

Example:

- **Input:**

- . num_list = [3, 4, 5, 1, -44, 5, 10, 12, 33, 1]
 - . k = 3

- **Output:** [5, 5, 5, 5, 10, 12, 33, 33]

Getting Max over Kernel

❖ How to solve?

3	1	-2	-1	5	4
---	---	----	----	---	---

$\max(3, 1) \rightarrow 3$

3	1	-2	-1	5	4
---	---	----	----	---	---

$\max(1, -2) \rightarrow 1$

3	1	-2	-1	5	4
---	---	----	----	---	---

$\max(-2, -1) \rightarrow -1$

3	1	-2	-1	5	4
---	---	----	----	---	---

$\max(-1, 5) \rightarrow 5$

3	1	-2	-1	5	4
---	---	----	----	---	---

$\max(5, 4) \rightarrow 5$

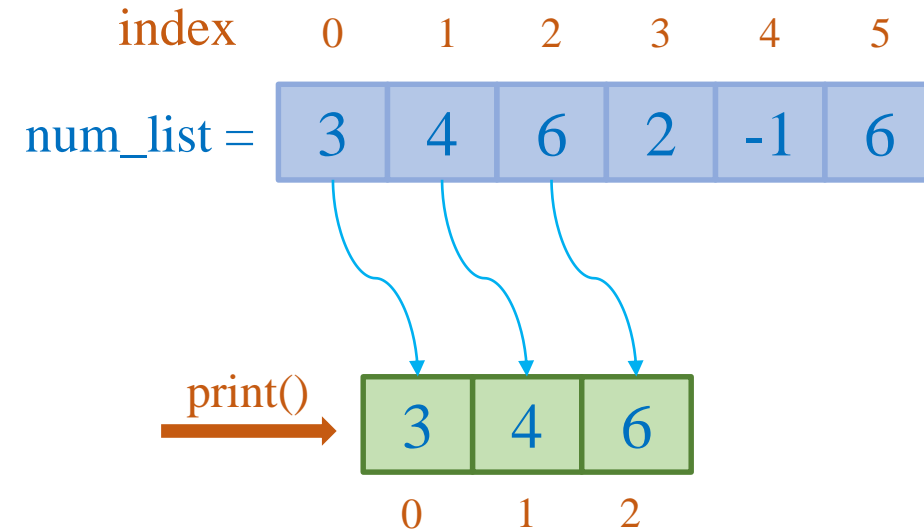
Use *slicing* and *max()* function

Getting Max over Kernel

❖ Get kernels

```
1 # get kernels - version 1
2
3 num_list = [3, 4, 6, 2, -1, 5]
4
5 sub_list = []
6 for element in num_list:
7     sub_list.append(element)
8     if len(sub_list) == k:
9         print(sub_list)
10        del (sub_list[0])
```

```
[3, 4, 6]
[4, 6, 2]
[6, 2, -1]
[2, -1, 5]
```

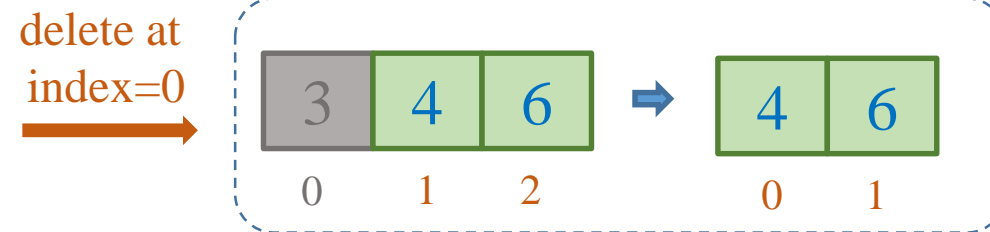
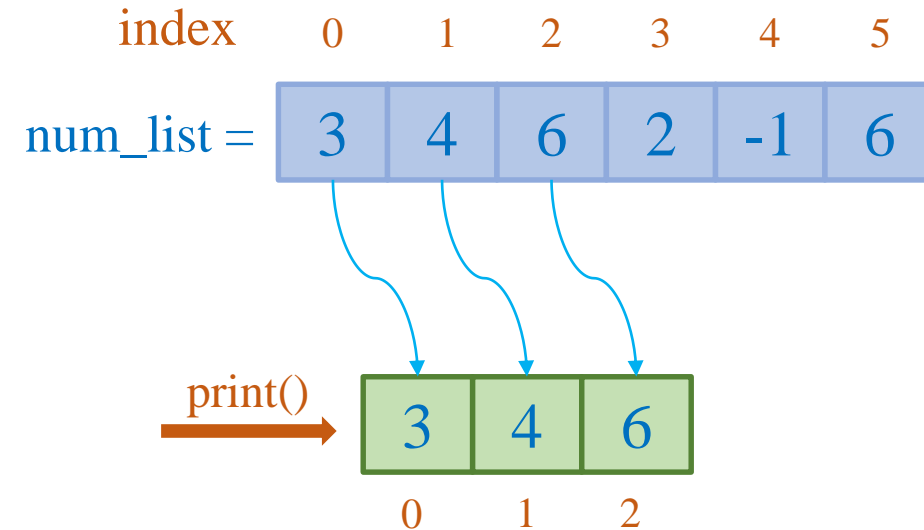


Getting Max over Kernel

❖ Get kernels

```
1 # get kernels - version 1
2
3 num_list = [3, 4, 6, 2, -1, 5]
4
5 sub_list = []
6 for element in num_list:
7     sub_list.append(element)
8     if len(sub_list) == k:
9         print(sub_list)
10    → del (sub_list[0])
```

```
[3, 4, 6]
[4, 6, 2]
[6, 2, -1]
[2, -1, 5]
```

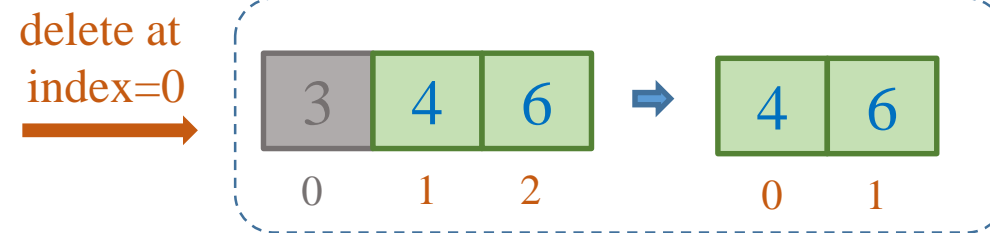
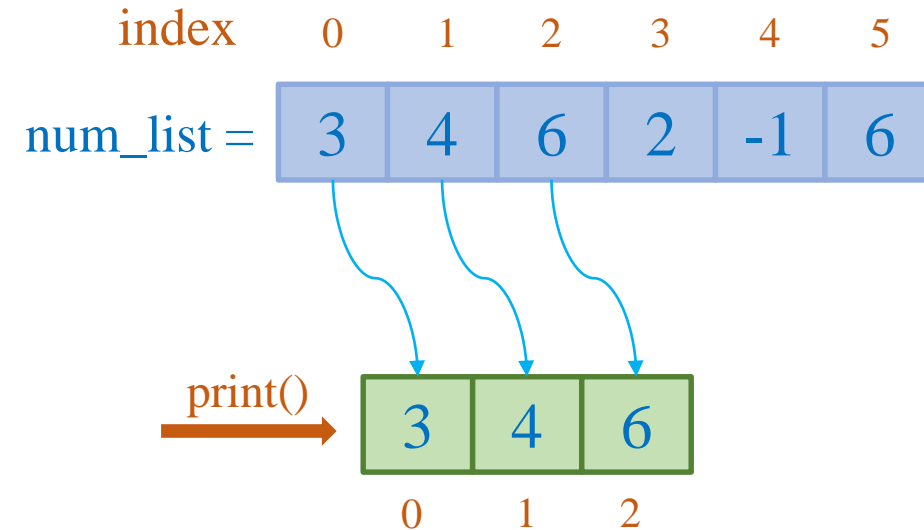


Getting Max over Kernel

❖ Get kernels

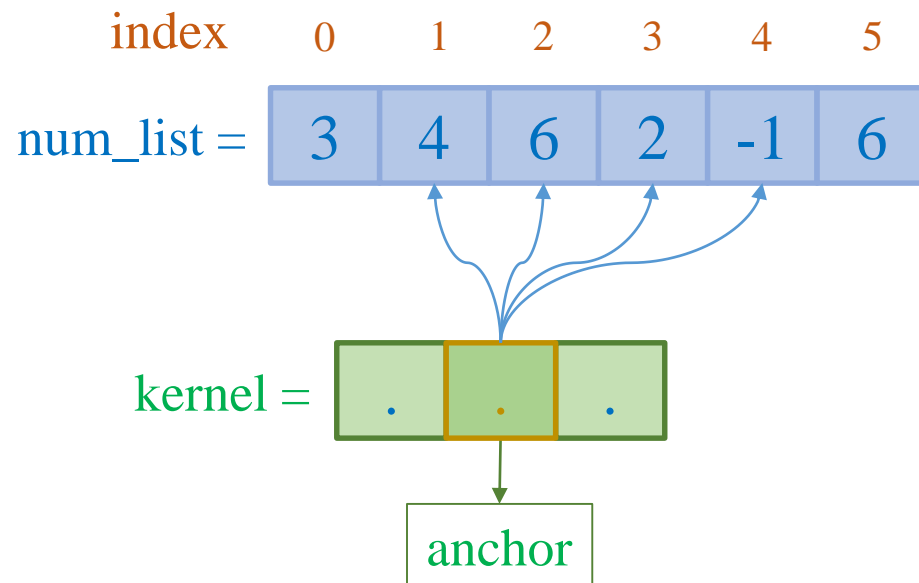
```
1 # get kernels - version 1
2
3 num_list = [3, 4, 6, 2, -1, 5]
4
5 sub_list = []
6 for element in num_list:
7     → sub_list.append(element)
8     if len(sub_list) == k:
9         print(sub_list)
10        del (sub_list[0])
```

```
[3, 4, 6]
[4, 6, 2]
[6, 2, -1]
[2, -1, 5]
```



Getting Max over Kernel

❖ Get kernels: Using indices

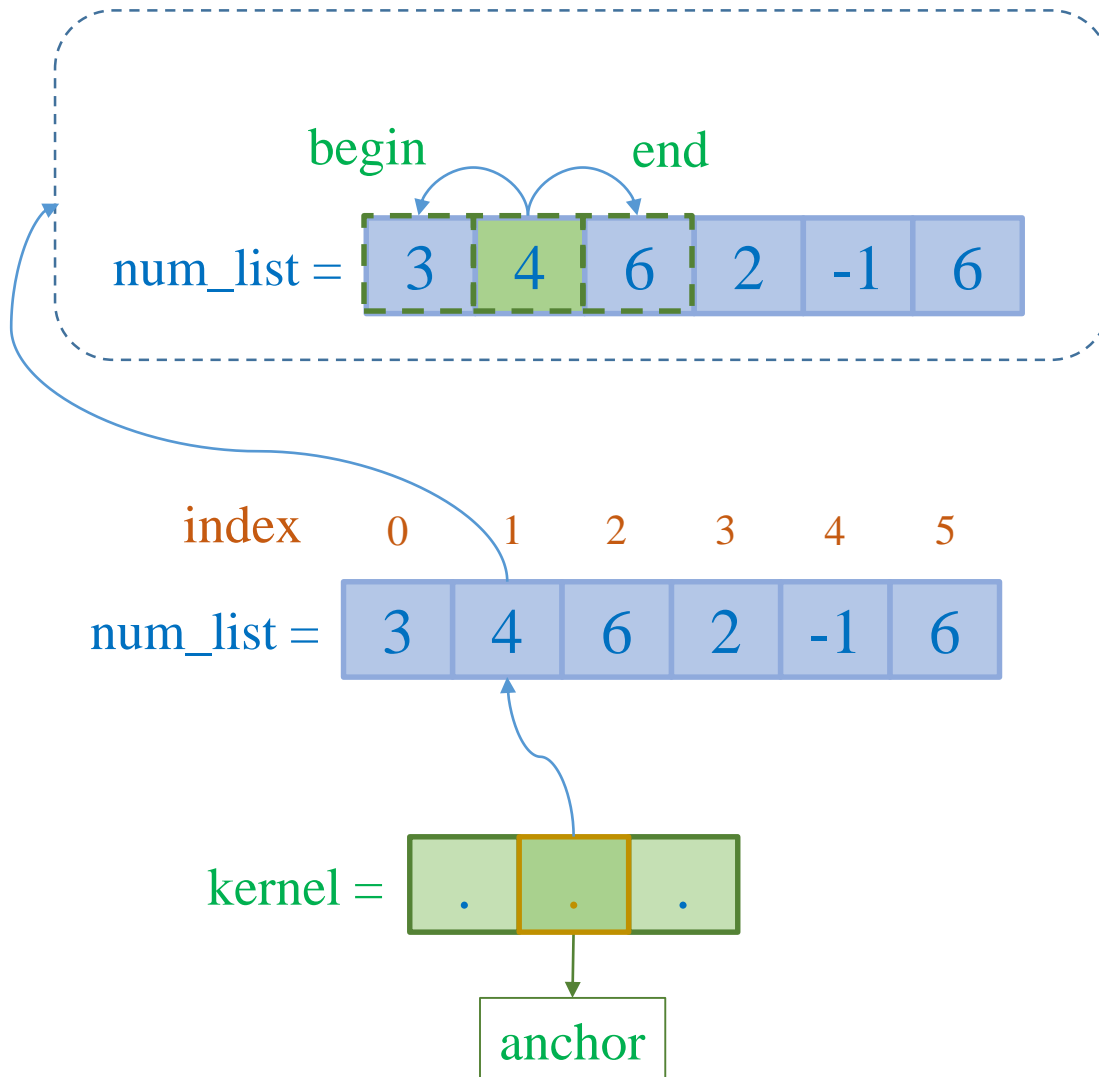


```
1  # get kernels - version 2
2
3  num_list = [3, 4, 6, 2, -1 , 5]
4  k = 3
5  radius = k//2
6  length = len(num_list)
7
8  for i in range(1, length-1):
9      begin = i-radius
10     end = i+radius
11     print(num_list[begin:end+1])
```

```
[3, 4, 6]
[4, 6, 2]
[6, 2, -1]
[2, -1, 5]
```

Getting Max over Kernel

❖ Get kernels: Using indices



```
1 # get kernels - version 2
2
3 num_list = [3, 4, 6, 2, -1, 5]
4 k = 3
5 radius = k//2
6 length = len(num_list)
7
8 for i in range(1, length-1):
9     begin = i-radius
10    end = i+radius
11    print(num_list[begin:end+1])
```

```
[3, 4, 6]
[4, 6, 2]
[6, 2, -1]
[2, -1, 5]
```

Getting Max over Kernel

```
1 # solution 1
2 num_list = [3, 4, 6, 2, -1 , 5]
3 k = 3
4
5 result = []
6 sub_list = []
7 for element in num_list:
8     sub_list.append(element)
9     if len(sub_list) == k:
10         result.append(max(sub_list))
11         del (sub_list[0])
12
13 print(result)
```

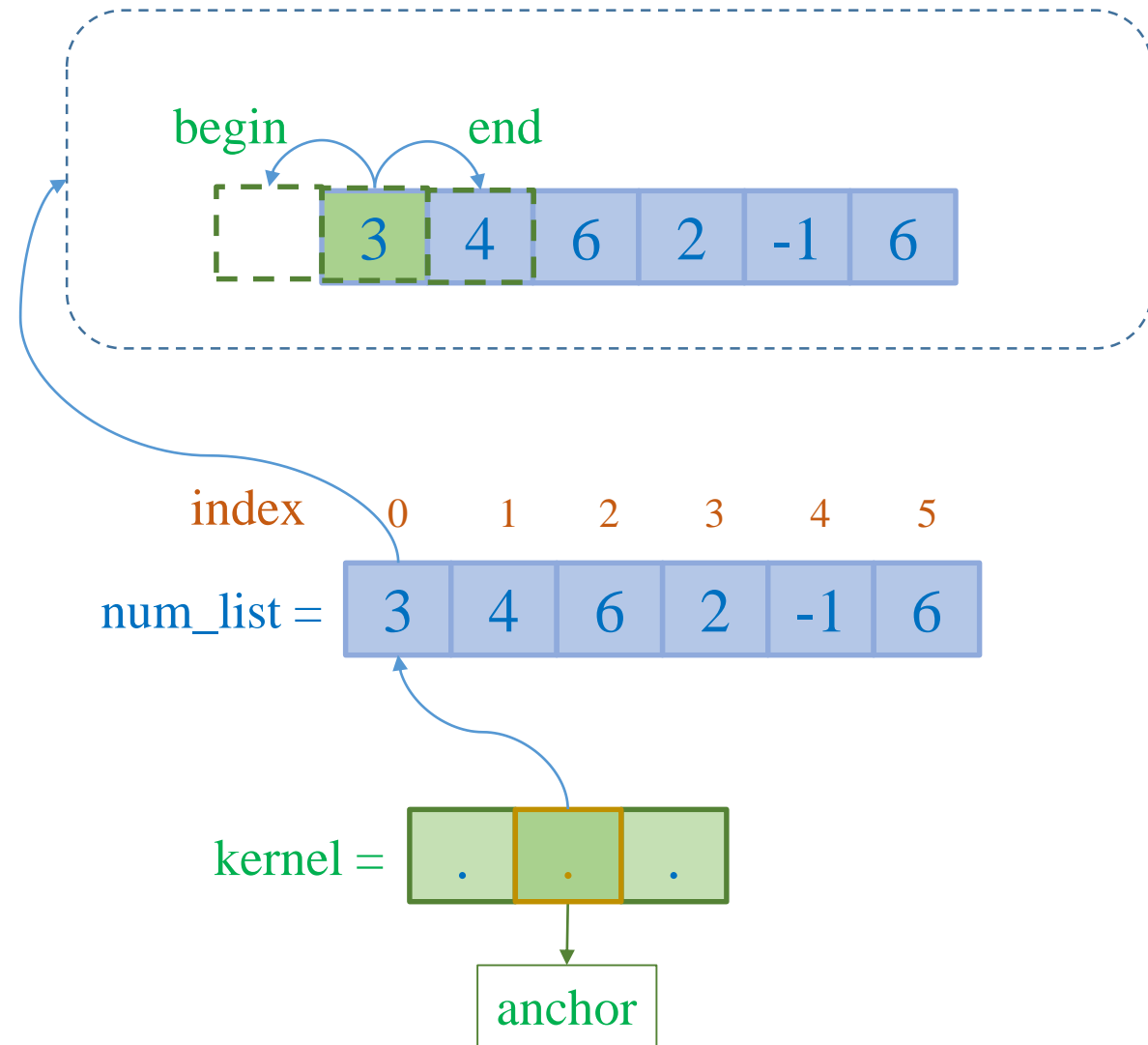
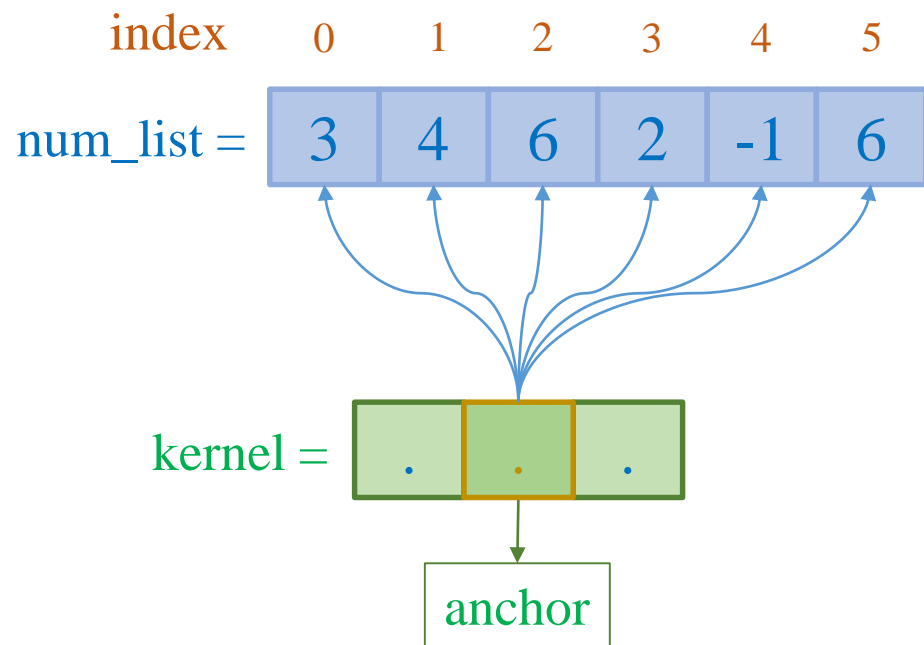
[6, 6, 6, 5]

```
1 # solution 2
2
3 num_list = [3, 4, 6, 2, -1 , 5]
4 k = 3
5 radius = k//2
6 length = len(num_list)
7
8 result = []
9 for i in range(1, length-1):
10     begin = i-radius
11     end = i+radius
12     result.append(max(num_list[begin:end+1]))
13 print(result)
```

[6, 6, 6, 5]

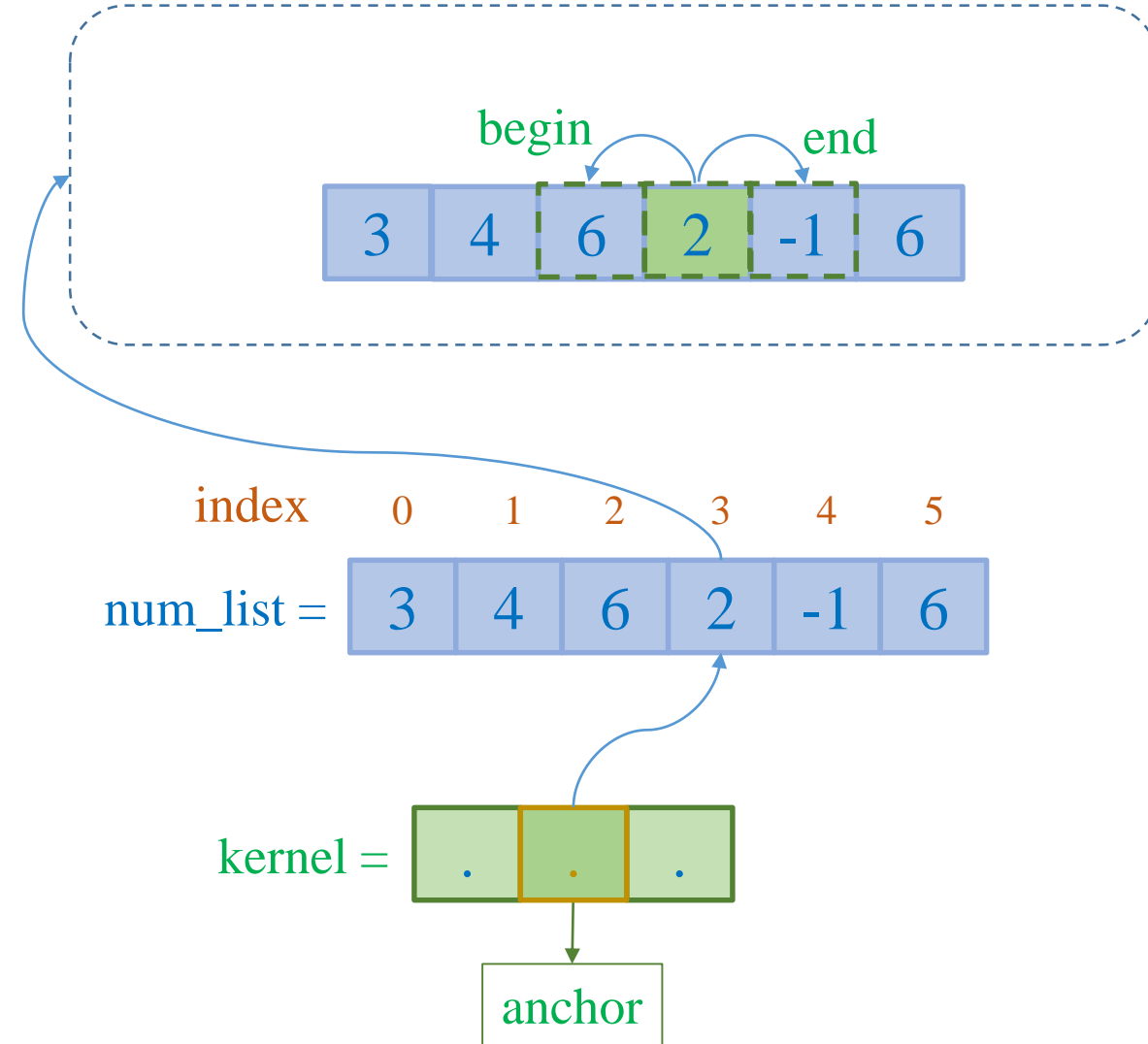
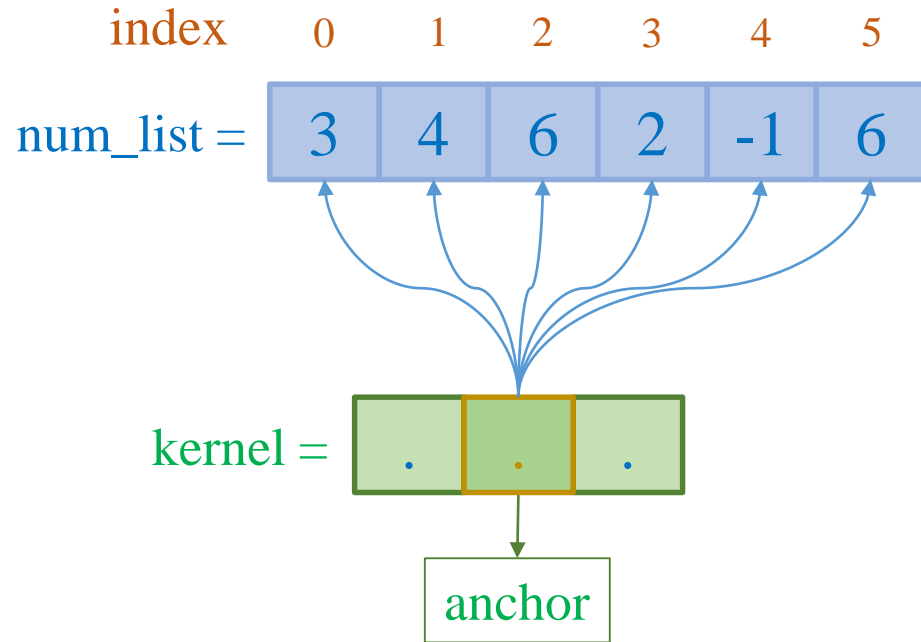
Getting Max over Kernel

❖ Extension 1: boundary awareness



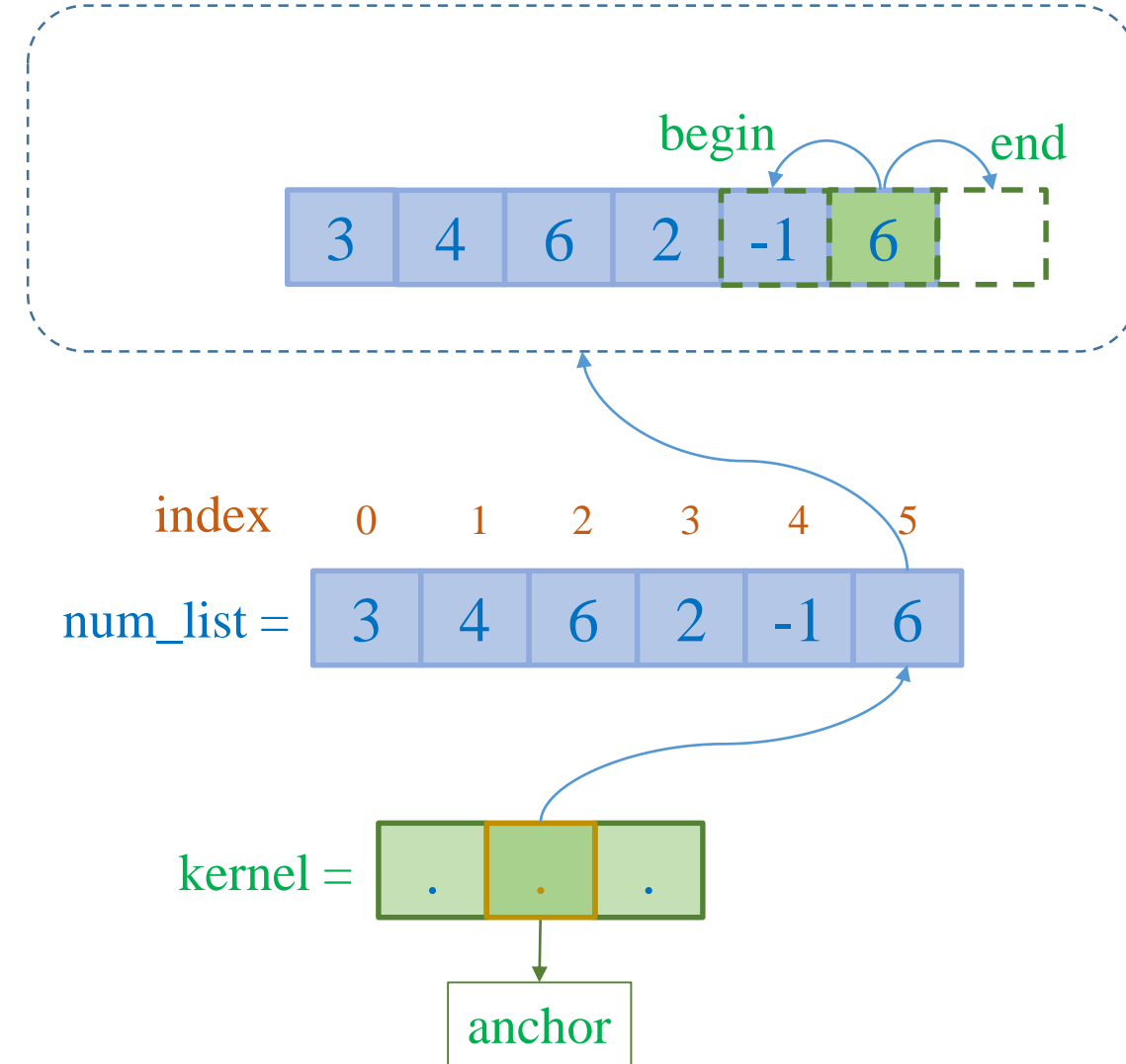
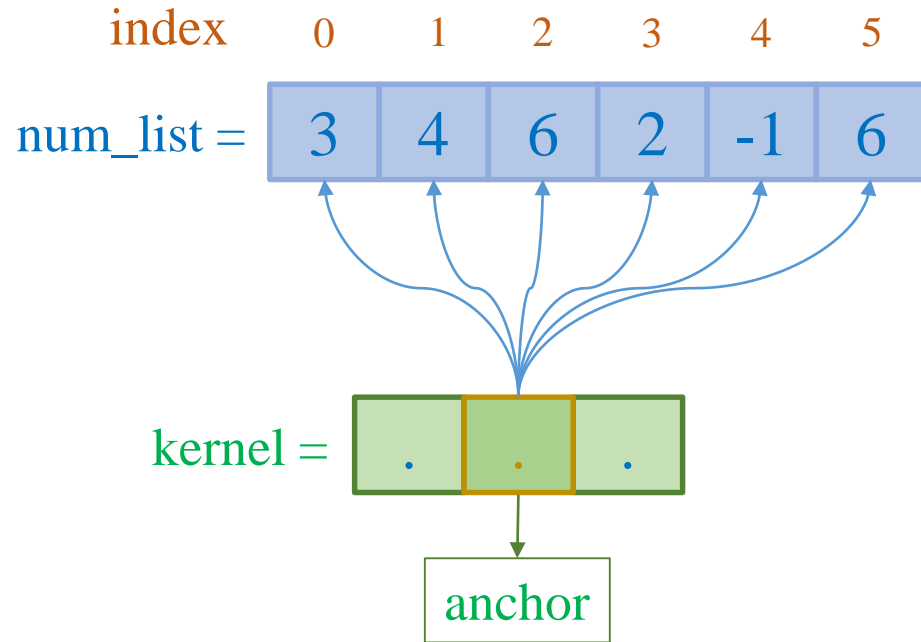
Getting Max over Kernel

❖ Extension 1: boundary awareness



Getting Max over Kernel

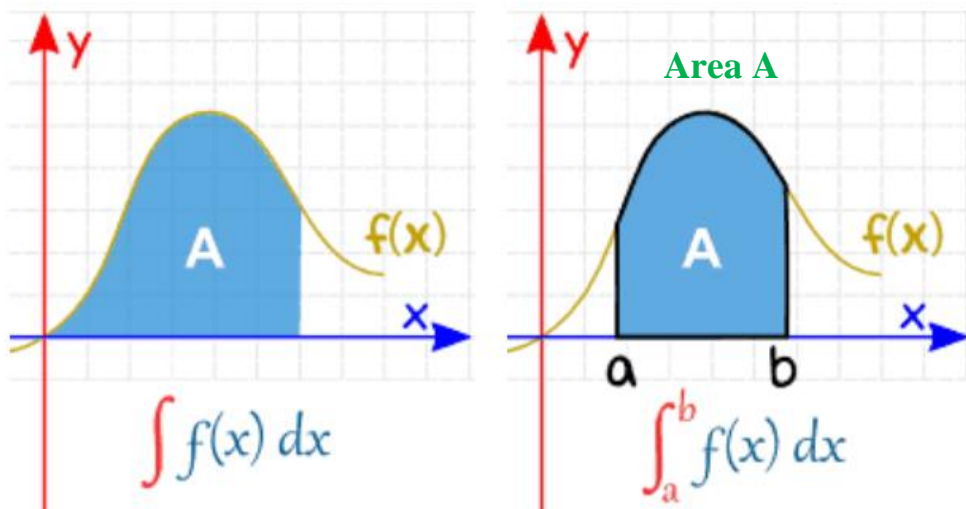
❖ Extension 1: boundary awareness



Getting Max over Kernel

❖ Extension 2: average over kernel (optional)

Formula



<https://www.mathsisfun.com/calculus/integration-introduction.html>

$$F(a) = \int_{-\infty}^a f(x) d(x)$$

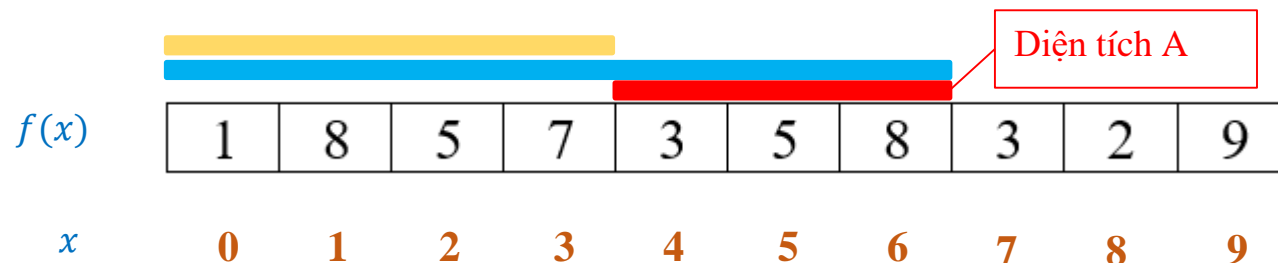
$$F(b) = \int_{-\infty}^b f(x) d(x)$$

Area A

$$A = F(b) - F(a)$$

$$f(x) \geq 0$$

For discrete functions (1D)



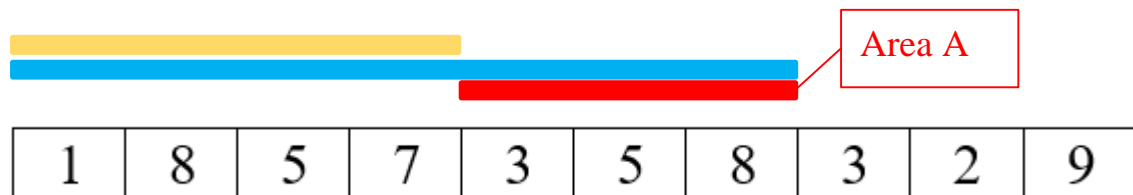
$$F(3) = \sum_{x \leq 3} f(x) = f(0) + f(1) + f(2) + f(3) \\ = 1 + 8 + 5 + 7 = 21$$

$$F(6) = \sum_{x \leq 6} f(x) = 1 + 8 + 5 + 7 + 3 + 5 + 8 = 37$$

$$A = F(6) - F(3) = \sum_{4 \leq x \leq 6} f(x) = 3 + 5 + 8 = 16$$

Getting Max over Kernel

For discrete functions (1D)



x 0 1 2 3 4 5 6 7 8 9

$$F(3) = \sum_{x \leq 3} f(x) = f(0) + f(1) + f(2) + f(3)$$

$$= 1 + 8 + 5 + 7 = 21$$

$$F(6) = \sum_{x \leq 6} f(x) = 1 + 8 + 5 + 7 + 3 + 5 + 8 = 37$$

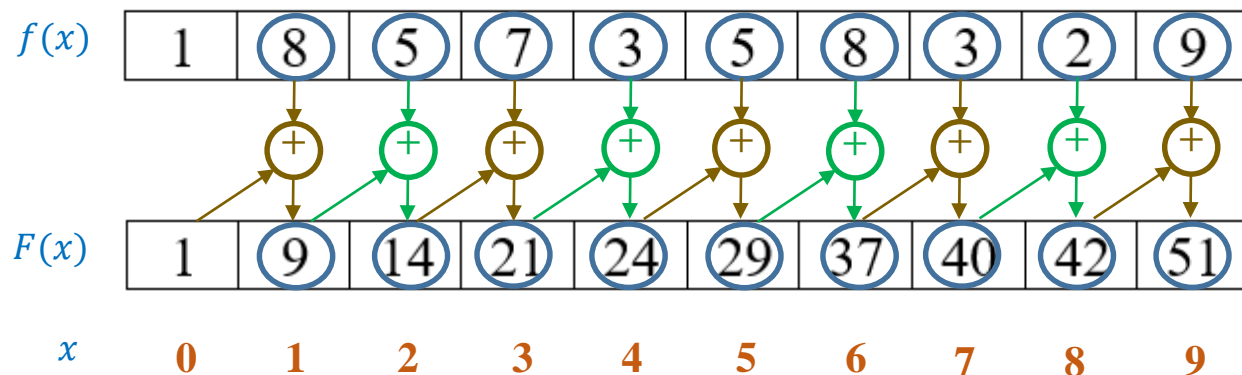
$$A = F(6) - F(3) = \sum_{4 \leq x \leq 6} f(x) = 3 + 5 + 8 = 16$$

Properties

$$F(x) = f(x) + F(x-1)$$

$$F(7) = f(7) + F(6) = 3 + 37 = 40$$

Construct integral array using $F(x) = f(x) + F(x-1)$



Compute sums with the complexity $\sim O(1)$

$$\sum_{a \leq x \leq b} f(x) = F(b) - F(a-1)$$

$$\sum_{4 \leq x \leq 6} f(x) = F(6) - F(3) = 37 - 21 = 16$$

