AY: 2024-2025		**Full Name:**	....
M1-S2: Dept. of Electrical Engineering		**ID:**	....
Midterm | NLP		**Class:**	RAIA1 ....
Apr. 2025		**Room:**	....
Teacher: A. Mhamdi		**Time Limit:**	1h

✂---

This document contains **6** pages numbered from **1/6** to **6/6**. As soon as it is handed over to you, make sure that it is complete. The **2** tasks are independent and can be treated in the order that suits you.

Do not write anything in this table.

Task	Points	Score
1	7	
2	13	
Total	20	

The following rules apply:

❶ **A handwritten double-sided A4** sheet is permitted.

❷ **The use of any electronic material**, except basic calculator, is prohibited.

❸ **Mysterious or unsupported answers** will not receive full credit.

❹ **If the provided space** is not sufficient, feel free to attach an additional sheet.

❺ **Task N°2:** Each correct answer will grant a mark with no negative scoring.

Task N°1

⌛ 30mn | (7 points)

For each text sample below, provide the regular expression pattern that matches the specific information described in each case.

(a) (1 point) Extract dates in YYYY-MM-DD format

Text: "Project deadlines: 2023-10-25, Q4 ends on 2024-12-31. Event on 15/11/2023 is postponed."

Extract: 2023-10-25, 2024-12-31

Regex Pattern: r"\d{4}-\d{2}-\d{2}"

(b) (1 point) Extract social media handles

Text: "Follow us @python_team on Twitter and @official_python3 on Instagram! #regex101"

Extract: @python_team, @official_python3

Do not write anything here

✂ -

Regex Pattern: r“@\w+”

(c) (1 point) Extract hashtags

Text: “Trending: #AI, #MachineLearning, and #coding101! Avoid #!!invalid_tags.”

Extract: #AI, #MachineLearning, #coding101

Regex Pattern: r“#\w+”

(d) (1 point) Extract currency amounts

Text: “Prices: $99.99, €50,00, ¥1000. Discount: 25%”

Extract: $99.99, €50,00, ¥1000

Regex Pattern: r“[\$€¥]\d+[\.,]?\d+”

(e) (1 point) Extract urls from text

Text: “Visit `https://www.python.org` or `http://docs.python.org/tutorial`. Avoid fake.site!”

Extract: `https://www.python.org`, `http://docs.python.org/tutorial`

Regex Pattern: r“https?://\S+[^\s\.]”

(f) (1 point) Extract usernames in logs

Text: “User ‘alice_2023’ logged in. Invalid user ‘123admin’ failed.”

Extract: alice_2023

Regex Pattern: r“'([a-zA-Z]\w+)'”

(g) (1 point) Extract license plate numbers

Text: “Plates: ABC-1234, XYZ-789 (invalid), DEF-5678.”

Extract: ABC-1234, DEF-5678

Do not write anything here

✂- -

Regex Pattern: r"[A-Z]{3}-\d{4}"

Task N°2 ⌛ 30mn | (13 points)

(a) (1 point) What does the code print?

```python
import re
text = "Python is fun"
result = re.match(r"is", text)
print(result is not None)
```

○ True [✓ False] ○ Error ○ None

(b) (1 point) What is the output?

```python
import re
text = "2025-04-14"
pattern = r"(\d{4})-(\d{2})-(\d{2})"
match = re.search(pattern, text)
print(match.group(2))
```

○ 2025 [✓ 04] ○ 14 ○ Error

(c) (1 point) What does the code print?

```python
import re
text = "a1b2c3"
result = re.findall(r"\d", text)
print(result)
```

[✓ ['1', '2', '3']] ○ [1, 2, 3] ○ ['a1', 'b2', 'c3'] ○ ['123']

(d) (1 point) What is the result?

```python
import re
text = "Hello World"
new_text = re.sub(r"\s", "-", text)
print(new_text)
```

○ HelloWorld ○ Hello World ○ Hello- World [✓ Hello-World]

Do not write anything here

✂- -

(e) (1 point) What does the code print?

```
import re
text = "<div>content</div>"
greedy = re.search(r"<.*>", text).group()
non_greedy = re.search(r"<.*?>", text).group()
print(greedy, non_greedy)
```

[✓ `<div>content</div> <div>`]

○ `<div> <div>`

○ `<div> </div>`

○ `<div>content</div> <div>content</div>`

(f) (1 point) Which regex matches "cat" or "cot" but not "cut"?

○ `c[a-z]t` ○ `c[^u]t` [✓ `c[ao]t`] ○ `c.o`

(g) (1 point) What does this regex pattern r"\d+(?=%)" match?

[✓ Numbers that are followed by a %]

○ Numbers preceded by a %

○ The % symbol itself

○ Numbers with at least two digits

(h) (1 point) What does `re.IGNORECASE` do?

○ Makes the regex case-sensitive

[✓ Makes matching case-insensitive]

○ Ignores whitespace in the pattern

○ Enables multi-line mode

(i) (1 point) What does r"^Python$" match?

○ Any string containing "Python"

○ "Python" at the start of a line

[✓ The line containing only the string "Python" (no other characters)]

○ "Python" at the end of a line

(j) ($1/2$ point) What is tokenization in NLP?

○ Removing punctuation from text

○ Translating text into another language

INSTITUTE OF TECHNOLOGICAL STUDIES OF BIZERTE

AY: 2024-2025		**Full Name:**	....
M1-S2: Dept. of Electrical Engineering		**ID:**	....
Midterm | NLP		**Class:**	RAIA1 ....
Apr. 2025		**Room:**	....
Teacher: A. Mhamdi		**Time Limit:**	1h

✂- -

- ○ Breaking text into sentences
- [✓ Breaking text into smaller units like words or phrases]

(k) ($1/2$ point) Which library is most suitable for tokenizing multilingual text?

○ spaCy ○ NLTK [✓ Polyglot] ○ PyTorch

(l) ($1/2$ point) How does spaCy handle punctuation during tokenization?

- ○ Punctuation marks are ignored.
- ○ Punctuation marks are merged with the nearest word.
- [✓ Punctuation marks are treated as separate tokens.]
- ○ Punctuation marks are removed entirely from the text.

(m) ($1/2$ point) What is the result?

```python
t = (1, [2, 3], 4)
t[1].append(5)
print(t)
```

- [✓ (1, [2, 3, 5], 4)]
- ○ (1, [2, 3], 4)
- ○ Error (tuples are immutable)
- ○ (1, [2, 3], 5, 4)

(n) ($1/2$ point) What is the output?

```python
nums = [1, 2, 3, 4]
squares = [x**2 for x in nums if x % 2 == 0]
print(squares)
```

○ [1, 4, 9, 16] [✓ [4, 16]] ○ [2, 4] ○ [4]

(o) ($1/2$ point) What does the code print?

```python
class Dog:
    SOUND = ”Woof”
    def __init__(self, name):
        self.name = name

d = Dog(”Buddy”)
Dog.SOUND = ”Bark”
print(d.SOUND)
```

Do not write anything here

○ None ○ Error ○ Woof [✓ Bark]

(p) ($^1/_2$ point) What is the output?

```python
class Animal:
    def speak(self):
        print("Animal sound")

class Cat(Animal):
    def speak(self):
        print("Meow")

c = Cat()
c.speak()
```

○ Animal sound [✓ Meow] ○ Error (no super()) ○ No output

(q) ($^1/_2$ point) What does the code print?

```python
class A:
    def greet(self):
        print("Hello from A")

class B(A):
    def greet(self):
        super().greet()
        print("Hello from B")

b = B()
b.greet()
```

○ Only "Hello from B"

[✓ "Hello from A" followed by "Hello from B"]

○ Error in "super()"

○ "Hello from A"