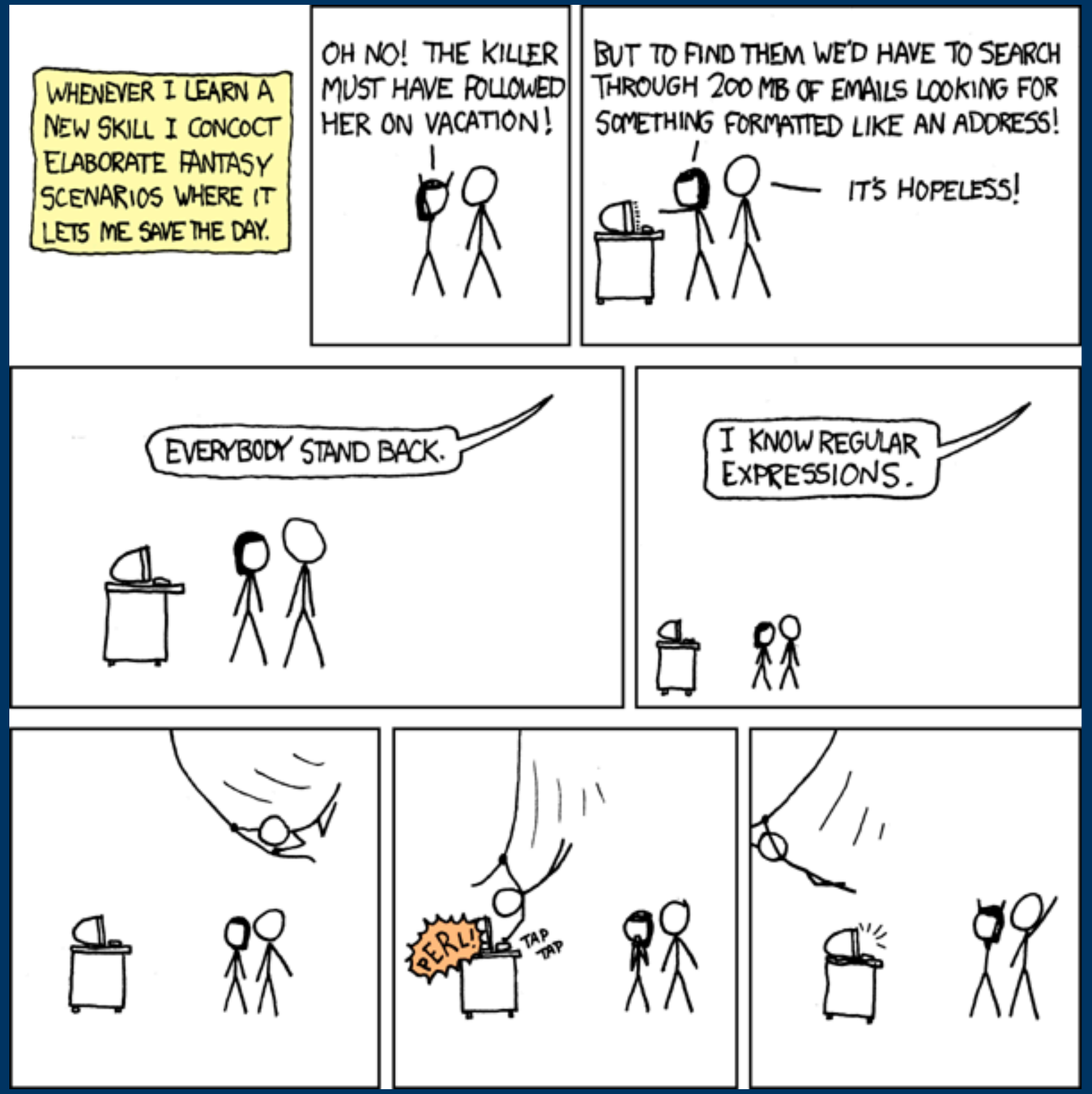
Recitation 7

Regex

Anamika Lochab



Regex

Regular Expression

- Special string for describing a pattern of characters, a form of pattern matching.

Regular expression	Description
[abc]	One of those three characters
[a-z]	A lowercase
[a-z0-9]	A lowercase or a number
.	Any one character
\.	An actual period
*	0 to many
?	0 or 1
+	1 to many

Why and When?

- WHY?
  - To find all of one particular kind of data
  - To verify that some piece of text follows a very particular format
- WHEN?
  - Used when data are unstructured or string operations are inadequate to process the data

How to define Regex?

- Mark regular expressions as raw strings r"
- Use square brackets "[" and "]" for “any character”

  r"[bce]" matches either “b”, “c”, or “e”

- Use ranges or classes of characters

  r"[A-Z]" matches any uppercase letter

  r"[a-z]" matches any lowercase letter

  r"[0-9]" matches any number

Note: use "-" right after [or before] for an actual "-"

r"[-a-z]" matches "-" followed by any lowercase letter

How to define Regex?

- Mark regular expressions as raw strings r"

- Use square brackets "[" and "]" for “any character”

  r"[bce]" matches either “b”, “c”, or “e”

- Use ranges or classes of characters

  r"[A-Z]" matches any uppercase letter

  r"[a-z]" matches any lowercase letter

  r"[0-9]" matches any number

Note: use "-" right after [or before] for an actual "-"

r"[-a-z]" matches "-" followed by any lowercase letter

How to define Regex?

- Use "*" for 0 to many

  r"[a-z]*" matches text with any number of lowercase letter

- Use "?" for 0 or 1

  r"[a-z]?" matches text with 0 or 1 lowercase letter

- Use "+" for 1 to many

  r"[a-z]+" matches text with at least 1 lowercase letter

- Use "|" for option

  r"[ab|12]" matches either ab or 12

How to define Regex?

- Use "^" for negate
  - r"[^a-z]" matches anything except lowercase letters
  - r"[^0-9]" matches anything except decimal digits
- Use "^" for “start” of string
  - r"^[a-zA-Z]" must start with a letter
- Use "$" for “end” of string
  - r".*[a-zA-Z]$" must end with a letter
- Use "{" and "}" to specify the number of characters
  - r"[a-zA-Z]{2,3}" must contain 2-3 letters
  - r"[a-zA-Z]{3}" must contain 3 letters

Predefined Regex Classes

\d	matches any decimal digit – [0-9]
\D	matches any non-digit character – [^0-9]
\s	matches any whitespace character – [\t\n]
\S	matches any non-whitespace – [^\t\n]
\\	matches a literal backslash
\w	matches any alphanumeric character – [a-zA-Z0-9_]
\W	matches any non-alphanumeric character – [^a-zA-Z0-9_]

Excercise

Define Regex patterns for the following

- Names :

- Phone-number (Format : _ _ _-_ _ _-_ _ _ _) :

- RUID :

?]

Exercise

Define Regex patterns for the following

- Names :

```
r"[A-Z][a-z]+"
```

- Phone-number (Format : _ _ _-_ _ _-_ _ _ _) :

```
r"[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]"
```

- RUID :

??

Regex in Python

- Import re module

```
import re
```

- Define a regular expression

```
regex = re.compile(r"[A-Za-z]+")
```

- Search / find the pattern in a given text

```
regex= re.compile(r"[A-Za-z]+")
a = "Alexandar"
print(re.search(regex,a))
print(regex.search(a))
print(re.search(r"[A-Za-z]+",a))
```

re.match(pattern, string)

Use re.match() when you want to check if the string starts with a certain pattern.

- This function attempts to match the regular expression pattern at the **beginning** of the string.
- If the match is found at the beginning of the string, it returns a match object; otherwise, it returns **None**

```
result = re.match("Python", "Python is great!")
print(result.group())
result = re.match("a", "Hakuna Matata")
print(result)
```

```
Python
None
```

re.search(pattern, string)

Use re.search() when you are interested in finding a pattern anywhere in the string but only need the first occurrence.

- This function searches the **entire** string for a match and returns the first occurrence as a match object.
- If no match is found, it returns **None**

```
result = re.search("great", "Python is great!")
print(result.group())
result = re.search("a", "Hakuna Matata")
print(result,"\n", result.group())
```

```
great
<re.Match object; span=(1, 2), match='a'>
 a
```

re.findall(pattern, string)

Use re.findall() when you want to find all occurrences of a pattern in a string.

- This function returns **all** occurrences of the pattern in the string.
- The result is a list of strings.

```
result = re.findall("a", "Hakuna Matata")
print(result)
```

```
['a', 'a', 'a', 'a', 'a']
```

re.finditer(pattern, string)

Use re.finditer() when you need more information about all occurrences of a pattern (like positions).

- This function is similar to *findall()*, but it returns an **iterator yielding match objects** instead of strings.
- This is useful for capturing groups and more advanced pattern matching.

```
result = re.finditer("a", "Hakuna Matata")
for match in result:
    print(match.start(), match.end(), match.group())
```

```
1 2 a
5 6 a
8 9 a
10 11 a
12 13 a
```

Exercise

How would you match a date in the format "YYYY-MM-DD" using regex?

```
[0-9]{4}-[0-9]{2}-[0-9]{2}
```

Exercise

Given a URL like "http://www.example.com", write a regex to match just the domain name "example".

```
text =  "http://www.example.com"
regex = re.compile(r'http://www\.(\w+)\.com')
re.findall(regex,text)
```

```
['example']
```

Exercise

You are given the string "Price: $500". Write a regex to extract just the numeric value (500).

```
text =  "Price:$500"
regex = re.compile(r'\$(\d+)')
re.findall(regex,text)
```

```
['500']
```

Exercise

Write a regex to validate a password that must contain at least 8 characters, one uppercase letter, one lowercase letter, and one number.

```python
def validate_password(password):
    pattern = r'(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}'
    if re.fullmatch(pattern, password):
        return "Password is valid."
    else:
        return "Password is invalid."


print(validate_password("StrongP@ss1"))
print(validate_password("weakpass"))
print(validate_password("SHORT1"))
```

```
Password is valid.
Password is invalid.
Password is invalid.
```