# LIVE 5: Strings and Regex-II

- Focus: Basics of strings and regex in Python + interesting problem solving.
- Prereq: Basic knowledge of Strings and Regex in Python + previous code-sessions.
- Reference for basics:
  - https://docs.python.org/3/howto/regex.html (https://docs.python.org/3/howto/regex.html)
  - https://docs.python.org/3/library/re.html (https://docs.python.org/3/library/re.html)
  - https://www.w3schools.com/python/python_strings.asp (https://www.w3schools.com/python/python_strings.asp)
  - https://www.geeksforgeeks.org/python-strings/ (https://www.geeksforgeeks.org/python-strings/)

Slide-show: https://medium.com/@mjspeck/presenting-code-using-jupyter-notebook-slides-a8a3c3b59d67 (https://medium.com/@mjspeck/presenting-code-using-jupyter-notebook-slides-a8a3c3b59d67)

# Problem 2: Extract data from a PDF invoice

- Given a PDF [https://slicedinvoices.com/pdf/wordpress-pdf-invoice-plugin-sample.pdf (https://slicedinvoices.com/pdf/wordpress-pdf-invoice-plugin-sample.pdf)], extract predefined key fields from this PDF
- Assume the format is fixed.

In [125]:

```
# https://realpython.com/pdf-python/#history-of-pypdf-pypdf2-and-pypdf4

!pip3 install pyPDF4
```

```
Requirement already satisfied: pyPDF4 in /usr/local/lib/python3.7/si
te-packages (1.27.0)
```

In [126]:

```
# Google "pyPDF extract text" ---> https://www.soudegesu.com/en/post/python/extract-text-from-pdf-with-pypdf2/
import PyPDF4

FILE_PATH = './invoice.pdf'

with open(FILE_PATH, mode='rb') as f:
    reader = PyPDF4.PdfFileReader(f)
    page = reader.getPage(0)
    print(page.extractText())
```

```
Invoice
Payment is due within 30 days from date of invoice. Late payment is
subject to fees of 5% per month.
Thanks for choosing
DEMO - Sliced Invoices
 |
admin@slicedinvoices.com
Page 1/1
From:
DEMO - Sliced Invoices
Suite 5A-1204
123 Somewhere Street
Your City AZ 12345
admin@slicedinvoices.com
Invoice Number
INV-3337
Order Number
12345
Invoice Date
January 25, 2016
Due Date
January 31, 2016
Total Due
$93.50
To:
Test Business
123 Somewhere St
Melbourne, VIC 3000
test@test.com
Hrs/Qty
Service
Rate/Price
Adjust
Sub Total
1.00
Web Design
This is a sample description...
$85.00
0.00%
$85.00
Sub Total
$85.00
Tax
$8.50
Total
$93.50
ANZ Bank
ACC # 1234 1234
BSB # 4321 432
Paid
```

In [127]:

```
import PyPDF4

FILE_PATH = './invoice.pdf'

with open(FILE_PATH, mode='rb') as f:
    reader = PyPDF4.PdfFileReader(f)
    page = reader.getPage(0)
    txt = page.extractText();
```

In [128]:

```
# extract invoice number

m = re.findall("INV-[0-9]*", txt)
print(m)
```

```
['INV-3337']
```

In [129]:

```
# extract amounts
m = re.findall("$[0-9]*\.[0-9]*", txt)
print(m)
```

```
[]
```

In [130]:

```
# extract amounts
m = re.findall("\$[0-9]*\.[0-9]*", txt)
print(m)
```

```
['$93.50', '$85.00', '$85.00', '$85.00', '$8.50', '$93.50']
```

In [165]:

```
# Extract Total Due:
m = re.findall("Total Due\$[0-9]*\.[0-9]*", txt)
print(m)

# Any suggestions?
```

```
[]
```

In [132]:

```
# Extract Total Due:
m = re.findall("Total Due\n\$[0-9]*\.[0-9]*", txt)
print(m)
```

```
['Total Due\n$93.50']
```

In [133]:

```
print(re.findall("\$[0-9]*\.[0-9]*",m[0]))
```

```
['$93.50']
```

In [141]:

```
# Extract dates in this doc gieven a fixed format using line number
res = re.split("\n", txt);
print(res)

print("\n\n Invoice date:" +res[18])
```

```
['Invoice', 'Payment is due within 30 days from date of invoice. Lat
e payment is subject to fees of 5% per month.', 'Thanks for choosing
', 'DEMO - Sliced Invoices', ' | ', 'admin@slicedinvoices.com', 'Pag
e 1/1', 'From:', 'DEMO - Sliced Invoices', 'Suite 5A-1204', '123 Som
ewhere Street', 'Your City AZ 12345', 'admin@slicedinvoices.com', 'I
nvoice Number', 'INV-3337', 'Order Number', '12345', 'Invoice Date',
'January 25, 2016', 'Due Date', 'January 31, 2016', 'Total Due', '$9
3.50', 'To:', 'Test Business', '123 Somewhere St', 'Melbourne, VIC 3
000', 'test@test.com', 'Hrs/Qty', 'Service', 'Rate/Price', 'Adjust',
'Sub Total', '1.00', 'Web Design', 'This is a sample descriptio
n...', '$85.00', '0.00%', '$85.00', 'Sub Total', '$85.00', 'Tax',
'$8.50', 'Total', '$93.50', 'ANZ Bank', 'ACC # 1234 1234', 'BSB # 43
21 432', 'Paid', '']


 Invocie date:January 25, 2016
```

**NOTE: Web-scarping**

- We can use "re" for extracting data from web-scarping.
- But, it is better to sue ebautiful-soup like libraries as they use the structure of HTML
- https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (https://www.crummy.com/software/BeautifulSoup/bs4/doc/)
- We have done an earlier live session on web-scraping: https://youtu.be/EYzTeb_VXoI (https://youtu.be/EYzTeb_VXoI)

## Ques: How do we handle cases where we want to extract data from multiple invoice formats?

**Assignment: Extract email-addresses from the PDF**

# Problem 3: Check if a number is a valid integer or float

- TRUE: 12, 12.5, 1e10, 1e+6, 1e-10, -2.3, -2.4e-4
- FALSE: abc, -2.4e4.5, 1b2.4
- TRICKY: Handle all the cases carefully.
- Easy problem to code. But, hard not to miss cases.
- Popular interview question to understand handling boundary cases.

In [173]:

```
# Source: https://www.geeksforgeeks.org/check-given-string-valid-number-integer-floating-point/
# Sometimes, reading other's code is a good way to learn.
# Explanatory comments are very important in your code.
# We added more comments and clearly listed cases handled for better interpretability.


# This code elimiates each of the FALSE cases and finally limits to only TRUE cases.
# is a valid number
def valid_number(str):
    i = 0
    j = len(str) - 1

    # Handling whitespaces: "   123    "
    while i<len(str) and str[i] == ' ': # remove  whitespaces int he beginning
        i += 1

    while j >= 0 and str[j] == ' ': # remove whitespaces at the end
        j -= 1

    if i > j: # if only whitespaces in the given string
        return False

    #str[i...j] is a whitespace removed (from beginning and end) string

    # if string is of length 1 and the only
    # character is not a digit
    if (i == j and not(str[i] >= '0' and
                       str[i] <= '9')):
        return False


    # If the 1st char is not '+', '-', '.' or digit
    if (str[i] != '.' and str[i] != '+' and
        str[i] != '-' and not(str[i] >= '0' and
        str[i] <= '9')):
        return False

    # To check if a '.' or 'e' is found in given
    # string.We use this flag to make sure that
    # either of them appear only once.
    flagDotOrE = False

    for i in range(j + 1):

        # If any of the char does not belong to
        # {digit, +, -,., e}
        if (str[i] != 'e' and str[i] != '.' and
            str[i] != '+' and str[i] != '-' and not
           (str[i] >= '0' and str[i] <= '9')):
            return False # "a123" good to write cases eliminated

        if str[i] == '.':

            # check if the char e has already
            # occured before '.' If yes, return 0
            if flagDotOrE:
```

```
                return False #"1e2.3", "1.2.3"

            if i + 1 > len(str):
                return False # "123."

            if (not(str[i + 1] >= '0' and
                    str[i + 1] <= '9')):
                return False # "123a"

            flagDotOrE = True

        elif str[i] == 'e':

            # set flagDotOrE = 1 when e is encountered.
            flagDotOrE = True

            # if there is no digit before e
            if (not(str[i - 1] >= '0' and
                    str[i - 1] <= '9')):
                return False # "e123"

            # if e is the last character
            if i + 1 > len(str):
                return False # "123e"

            # if e is not followed by
            # '+', '-' or a digit
            if (str[i + 1] != '+' and str[i + 1] != '-' and
               (str[i + 1] >= '0' and str[i] <= '9')):
                return False # "1e." "1ea"

    # If the string skips all the
    # above cases, it must be a numeric string
    return True
```

In [174]:

```
print(valid_number("1e5"))
```

```
True
```

In [175]:

```
print(valid_number("1e1.5"))
```

```
False
```

In [176]:

```
print(valid_number("1e+15"))
```

```
True
```

In [177]:

```
print(valid_number("-1.2e-15"))
```

```
True
```

# Problem 4: Regex matching problem

- "?" matches a single character
- "*" matches zero or more charcters
- Given a pattern(p) and a string(s), does p match s?
- examples:
  - TRUE: ("*", "ab"), ("?a" ,"ba"), ("?a" ,"aa"), ("a*" ,"a")
  - FALSE: ("*a", "ab"), ("?a" ,"baa"), ("?a" ,"a"), ("a*" ,"ba")
- Very popular interview question at product-based companies for SDEs.
- Small variations of this are often used in interviews
- Any suggestions?

In [152]:

```
# Handle all cases of recursion thoroughly.

def isMatch(p,s):

    print(p,s) # print statemnt for debugging

    # boundary cases of recursion
    if p == s:
        return True

    if p == "*":
        return True

    if p == "" or s == "":
        return False

    # recursion case-1
    if p[0] == s[0] or p[0] == '?':
        return isMatch(p[1:], s[1:])

    # recursion-case-2
    if p[0] == '*':
        return ( isMatch( p[1:], s) or isMatch( p, s[1:]))

    # last case: if p[0] is a character
    if  p[0] != s[0]:
        return False;
```

In [149]:

```
print(isMatch("*","ab"))
```

```
* ab
True
```

In [150]:

```
print(isMatch("?a" ,"baa"))
```

```
?a baa
a aa
 a
False
```

In [151]:

```
print(isMatch("a*" ,"ba"))
```

```
a* ba
False
```

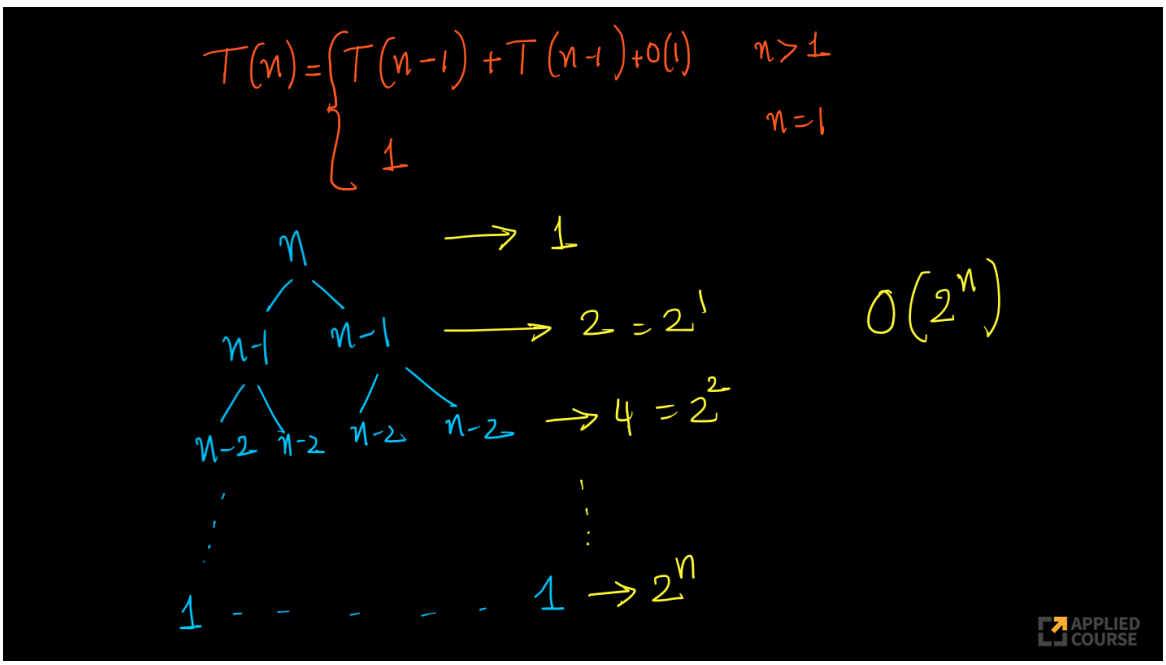
In [153]:

```
# Worst case Time Complexity: T(n) = T(n-1) + T(n-1) = 2* T(n-1)

# => T(n) = 0(2^n) [as shown below]


from IPython.display import Image
Image(url= "https://i.imgur.com/dse47H3.png")
```

Out[153]:

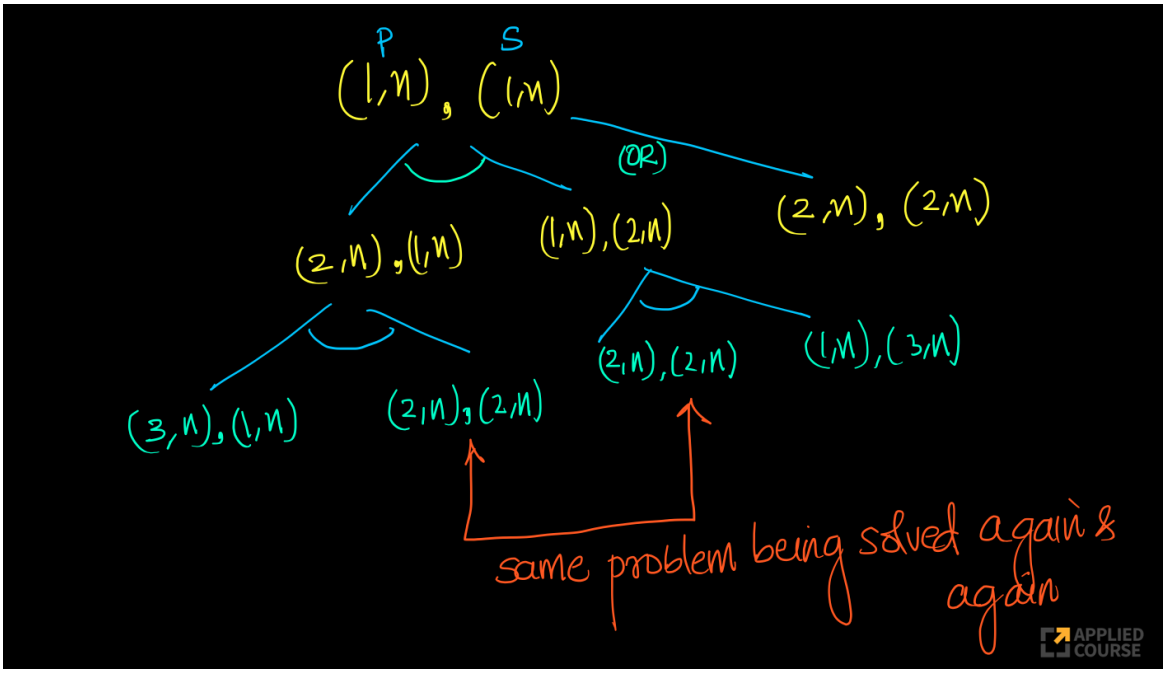


In [155]:

```
# Can we do better?

Image(url= "https://i.imgur.com/Rx6tN8a.png")
```

Out[155]:



In [164]:

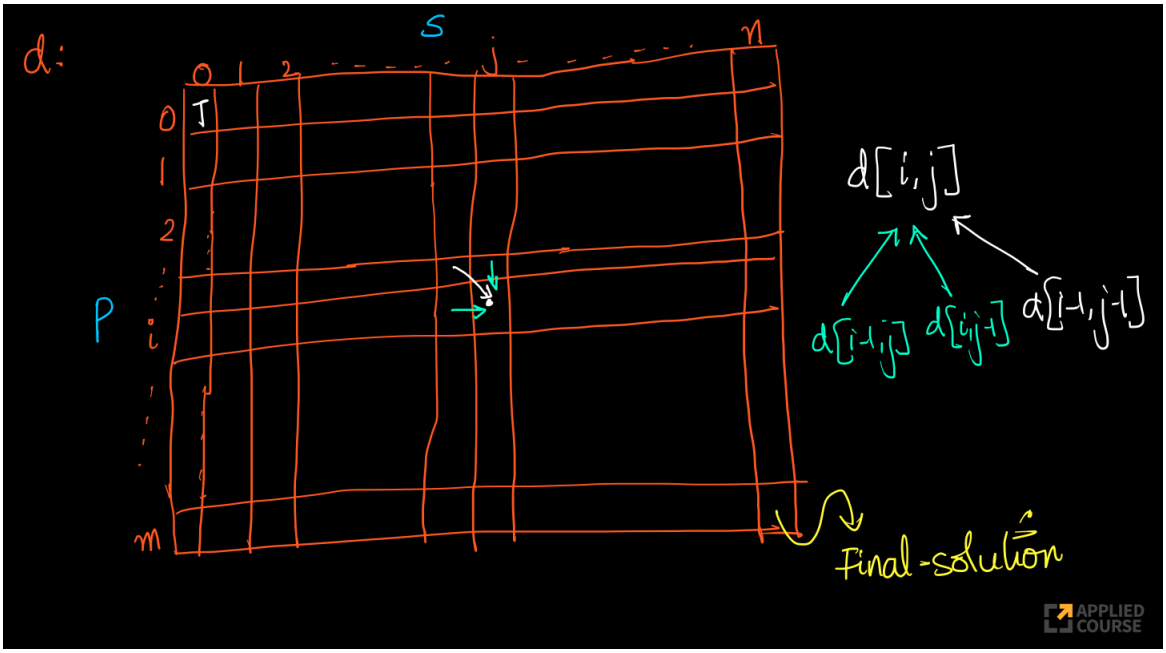
```
# Overlapping sub-problems:
# Why not store solutions to already solved problems in a 2D array of [0...m-1]
[0...n-1]
# Dynamic programming: Recursion + overlapping subproblems

# We discussed this in our course when we learn back-propogation in DL.

# DONOT need to use recursion also now. Can solve iteratively.

Image(url= "https://i.imgur.com/k6ZTMcm.png")
```

Out[164]:



In [172]:

```
# Source: https://leetcode.com/problems/wildcard-matching/solution/
# GIVEN THE ABOVE INUTITON, READING THIS CODE SHOULD BE STRAIGHT FORWARD
# Time Complx: O(m*n).

# Exercise: Go through this code line by line while keeping the logic in mind!

def isMatchDP(p,s):
    s_len = len(s)
    p_len = len(p)

    # base cases
    if p == s or p == '*':
        return True
    if p == '' or s == '':
        return False

    # init all matrix except [0][0] element as False
    d = [ [False] * (s_len + 1) for _ in range(p_len + 1)]
    d[0][0] = True

    # DP compute
    for p_idx in range(1, p_len + 1):

        # the current character in the pattern is '*'
        if p[p_idx - 1] == '*':

            s_idx = 1

            # d[p_idx - 1][s_idx - 1] is a string-pattern match
            # on the previous step, i.e. one character before.
            # Find the first idx in string with the previous math.

            # p=abcd*  s=abcdefg
            while not d[p_idx - 1][s_idx - 1] and s_idx < s_len + 1:
                s_idx += 1

            # If (string) matches (pattern),
            # when (string) matches (pattern)* as well
            d[p_idx][s_idx - 1] = d[p_idx - 1][s_idx - 1]

            # If (string) matches (pattern),
            # when (string)(whatever_characters) matches (pattern)* as well
            while s_idx < s_len + 1:
                d[p_idx][s_idx] = True
                s_idx += 1

        # the current character in the pattern is '?'
        elif p[p_idx - 1] == '?':
            for s_idx in range(1, s_len + 1):
                d[p_idx][s_idx] = d[p_idx - 1][s_idx - 1]

        # the current character in the pattern is not '*' or '?'
        else:
            for s_idx in range(1, s_len + 1):
                # Match is possible if there is a previous match
                # and current characters are the same
                d[p_idx][s_idx] = \
                d[p_idx - 1][s_idx - 1] and p[p_idx - 1] == s[s_idx - 1]
```

```
        return d[p_len][s_len]
```

In [163]:

```
print(isMatchDP("a*" ,"ba"))
```

```
False
```

## Next session: Python's inbuilt data-structures: List, Dict, Set, Tuple