LIVE 4: Strings and Regex

- Focus: Basics of strings and regex in Python + Simple 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/library/re.html
    - https://www.w3schools.com/python/python_strings.asp
    - https://www.geeksforgeeks.org/python-strings/

Quick recap of Regex in Python

- Go through multiple examples to understand regex better
- Key life-skill: learn from resources on the internet.
- https://docs.python.org/3/howto/regex.html
- https://www.w3schools.com/python/python_regex.asp
- https://www.tutorialspoint.com/python/python_reg_expressions.htm

Problem-1: Mask personal information in email and phone numbers

- Email: xxxxxxxxx@aaaa.zzzz
    - Masked:x#####x@aaaa.zzzz [FIVE # between first and last char of the name]

- Phone: digits 0-9 or any of the characters from { '-', '(', ')', ' '}
    - Example: 1(234)567-890 --> ###-###-7890"

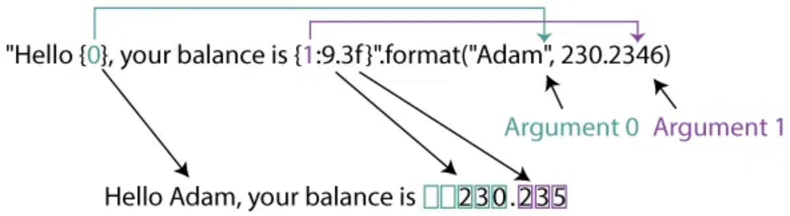
```
# email-masking
s = "abcd@efgh.com";

#output: a#####d@efgh.com
# Any suggestions?
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
```

```
#
#

#simple string formatting : https://www.programiz.com/python-programming/methods/string/format

from IPython.display import Image
Image(url= "https://cdn.programiz.com/sites/tutorial2program/files/python-format-positional-argument.jpg")
```




```
# boundary case: check if email or not
s = "abcd@efgh.com";

def maskEmail(s):
    if '@' in s:
        name, domain = s.split('@')
        return ("{0}#####{1}@{2}".format(name[0], name[-1], domain));

print(maskEmail(s))
```

```
a#####d@efgh.com
```

```
# BOUDNARY CASE: a@bcdef.com
print(maskEmail("a@bcdef.com"))
```

```
a#####a@bcdef.com
```

```
# BOUDNARY CASE: abcd.com
print(maskEmail("abcd.com"))
```

```
None
```

```
# BOUDNARY CASE: abcd@cdef
print(maskEmail("abcd@cdef"))
```

```
a#####d@cdef
```

```
# Check if email is valid is another function.
# Any suggestions?
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
```

```
#
#
#
#
#
#
#
import re
def isValidEmail(s):
    #refer:https://www.w3schools.com/python/python_regex.asp for regex syntax
    #https://docs.python.org/2/library/re.html

    res = re.search('^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$', s, re.IGNORECASE)   #https://www.geeksforgeeks.org/chec
    print(res)

    if(res):
        return True;
    else:
        return False;

print(isValidEmail("abcd@cdef"))
```

```
None
False
```

```
print(isValidEmail("abcd@cdef.c"))
```

```
None
False
```

```
print(isValidEmail("a@cdef.com"))
```

```
<re.Match object; span=(0, 10), match='a@cdef.com'>
True
```

```
print(isValidEmail("abcd@iisc.ac.in"))
```

```
<re.Match object; span=(0, 15), match='abcd@iisc.ac.in'>
True
```

```
regex = '^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$' # highly non-readbale code

#https://docs.python.org/2/library/re.html
regex_verbose = re.compile(r"""                  # VERY readable and easy to understand. Software maintanability.
            ^\w+([\.-]?\w+)*              # start, \w+,
            @                             # single @ sign
            \w+([\.-]?\w+)*               # Domain name
            (\.\w{2,3})+$                 # .com, .ac.in,
             """,re.VERBOSE | re.IGNORECASE)


res = regex_verbose.match("abcd@iisc.ac.in");

print(res.string)
print(res)
```

```
abcd@iisc.ac.in
<re.Match object; span=(0, 15), match='abcd@iisc.ac.in'>
```

```
# PHONE NUMBER MASKING
#Example: 1(234)567-890 --> ###-###-7890"


ph = "1(234)567-890"


digits = re.sub("\D", "", ph)  # \D=>non-decimal, re.substitute, https://docs.python.org/3/library/re.html

print(digits)

masked =  "###-###-{}".format(digits[-4:])
print(masked)
```

```
1234567890
###-###-7890
```

```
def maskPhoneNum(ph):
    digits = re.sub("\D", "", ph)  # \D=>non-decimal, re.substitute, https://docs.python.org/3/library/re.html
    if len(digits) != 10: # BOUNDARY CASE
        return None;
    else:
        masked =  "###-###-{}".format(digits[-4:])
        return masked

print(maskPhoneNum("1(234)567-890"))
```

```
###-###-7890
```

print(maskPhoneNum("1(234)567-89"))

Exercise: 12 digit phone numbers with 2 digits of ISD code strtaing with +

- e.g: +86-(99)12345678 ----> (+86)-###-###-5678

Problem 2: Extract data from a PDF invoice

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

NOTE: Download and save the above PDF as invoice.pdf in the same folder as your iPython notebook for the following code to work

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

!pip3 install pyPDF4
```

```
Collecting pyPDF4
  Downloading PyPDF4-1.27.0.tar.gz (63 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.9/63.9 kB 2.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: pyPDF4
  Building wheel for pyPDF4 (setup.py) ... done
  Created wheel for pyPDF4: filename=PyPDF4-1.27.0-py3-none-any.whl size=61227 sha256=2880c6cb500e17570545115d779442ae08
  Stored in directory: /root/.cache/pip/wheels/83/cc/14/cb307e5c99235c4497c7895cdb60b4f7ba2a738b6a5fc0d423
Successfully built pyPDF4
Installing collected packages: pyPDF4
Successfully installed pyPDF4-1.27.0
```

```
from google.colab import files
uploaded = files.upload()  # This will open a file upload dialog
```

```
Choose files invoice.pdf
  • invoice.pdf(application/pdf) - 43627 bytes, last modified: 21/12/2024 - 100% done
Saving invoice.pdf to invoice (2).pdf
```

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

FILE_PATH = './invoice (2).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
```

```
import PyPDF4

FILE_PATH = './invoice (2).pdf'

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


# extract email address

import re
email_pattern = r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}'

email = re.findall(email_pattern, txt)
print(email)
```

```
['admin@slicedinvoices.com', 'admin@slicedinvoices.com', 'test@test.com']
```

```
# extract invoice number
import re
m = re.findall("INV-[0-9]*", txt)
print(m)
```

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

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

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

Start coding or generate with AI.

```
# extract amounts
```

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

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

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

```
[]
```

```
# Any suggestions?

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

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

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

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

Ques: How do we handle cases where we want to extract data from multiple

- List item
- List item

invoice formats?

Assignment: Extract email-addresses from the PDF

We will continue from here tomorrow. Pelase go through regex- referecnes in detail for tomorrow's session. We will solve a few product based company interview questions.

```
Start coding or generate with AI.
```