In [ ]:
```
# Regular expressions

import re

txt = 'Python is a programming language.'

m = re.search('programming', txt)
print(m)  # <re.Match object; span=(12, 23), match='programming'>

m = re.search('are', txt)
print(m) # None
```

In [ ]:
```
m = re.search('^is', txt)
print(m) # None

m = re.search('^Python', txt)
if (m):
    print('yes')
else:
    print('no')
```

In [ ]:
```
m = re.search('programming$', txt)
if (m):
    print('yes')
else:
    print('no')

m = re.search('language.$', txt)
if (m):
    print('yes')
else:
    print('no')
```

In [ ]:
```
x = 'phone number 091212123344 and another 02122334455 number.'

m = re.search('number \d+', x)
print(m)
# <re.Match object; span=(6, 25), match='number 091212123344'>

print(m.group(0)) # number 091212123344
#print(m.group(1))    # Error


m = re.search('number (\d+)', x)
print(m.group(0))    # number 091212123344
print(m.group(1))    #  091212123344


m = re.search('(\w+) (\d+)', x)
print(m.group(0))    # number 091212123344
print(m.group(1))    #  number
print(m.group(2))    #  091212123344


print(re.findall('\d+', x))      # ['091212123344', '02122334455']
print(re.findall('\w+ \d+', x))  # ['number 091212123344', 'another 021223344
print(re.findall('\d+ \w+', x))  # ['091212123344 and', '02122334455 number']
print(re.findall('[0-9]+', x))   # ['091212123344', '02122334455']
print(re.findall('[0-2]+', x))   # ['0', '121212', '02122']
```

In [ ]:
```
name = 'Farshid Shirafkan'
print(re.findall('z', name))        # []
print(re.findall('f', name))        # ['f']

k = re.findall('[a-f]', name)
print(k)                            # ['a', 'd', 'a', 'f', 'a']

print(re.findall('\s+', name))     # [' ']
print(re.findall('\S+', name))     # ['Farshid', 'Shirafkan']
print(re.findall('r[^ ]*', name))  # ['rshid', 'rafkan']
print(re.findall('r[^i]*', name))  # ['rsh', 'rafkan']
```

In [ ]:
```
e = 'From ali@gmail.com to sara@yahoo.com'
words = e.split()
print(words)      # ['From', 'ali@gmail.com', 'to', 'sara@yahoo.com']
print(words[1])   # ali@gmail.com
print(words[3])   # sara@yahoo.com
print(re.findall('\S+@\S+' , e))   # ['ali@gmail.com', 'sara@yahoo.com']
print(re.split('\s', e))      # ['From', 'ali@gmail.com', 'to', 'sara@yahoo.co
print(re.split('\s', e, 1))  # ['From', 'ali@gmail.com to sara@yahoo.com']
```

In [ ]:
```
txt = 'Python is a programming language.'

print(re.sub('\s', '_', txt)) # Python_is_a_programming_language.
print(re.sub('\S', 'a', txt)) # aaaaaa aa a aaaaaaaaaaa aaaaaaaaa
print(re.sub('\s', '_', txt, 2)) # Python_is_a programming language.
```

In [ ]:
```
phone = '0912-197-12345'
print(re.sub('\d', '#', phone))  # ####-###-#####
print(re.sub('\D', '#', phone))  # 0912#197#12345
```

In [ ]:
```
p = '  farsh id   '
r = re.sub('^\s+', '', p)
print(r)  #farsh id

r2 = re.sub('\s+$', '', p)
print(r2)  #   farsh id
```

In [ ]:
```
s ='ABCDEFCGH'
r = re.subn('CD', 'X', s)
print(r)                           # ('ABXEFCGH', 1)

s ='ABCDEFCGH'
u = re.subn('C', 'X', s)
print(u)                          # ('ABXDEFXGH', 2)
```

In [ ]:
```
s ='ABCDEFCGH'
f = re.search('CDE', s)
print(f)        # <re.Match object; span=(2, 5), match='CD'>
a = f.start()   # 2
b = f.end()     # 5

k = s[ :a] + s[b: ]
print(k)                # ABFCGH
```

In [ ]:
```
text = "He was carefully disguised but captured quickly by police."
t = re.findall(r"\w+ly", text)
print(t)                           # ['carefully', 'quickly']

fi = re.finditer(r"\w+ly", text)
for m in fi:
     print(m.start(), m.end(), m.group(0))

'''
7 16 carefully
40 47 quickly
'''
```

In [ ]:

```
from typing import NamedTuple

class Token(NamedTuple):
    type: str
    value: str
    line: int
    column: int

def tokenize(code):
    keywords = {'IF', 'THEN', 'ENDIF'}
    token_specification = [
        ('NUMBER',   r'\d+(\.\d*)?'),  # Integer or decimal number
        ('ASSIGN',   r':='),           # Assignment operator
        ('END',      r';'),            # Statement terminator
        ('ID',       r'[A-Za-z]+'),    # Identifiers
        ('OP',       r'[+\-*/]'),      # Arithmetic operators
        ('NEWLINE',  r'\n'),           # Line endings
        ('SKIP',     r'[ \t]+'),       # Skip over spaces and tabs
        ('MISMATCH', r'.'),            # Any other character
    ]
    tok_regex = '|'.join('(?P<%s>%s)' % pair for pair in token_specification)
    line_num = 1
    line_start = 0
    for mo in re.finditer(tok_regex, code):
        kind = mo.lastgroup
        value = mo.group()
        column = mo.start() - line_start
        if kind == 'NUMBER':
            value = float(value) if '.' in value else int(value)
        elif kind == 'ID' and value in keywords:
            kind = value
        elif kind == 'NEWLINE':
            line_start = mo.end()
            line_num += 1
            continue
        elif kind == 'SKIP':
            continue
        elif kind == 'MISMATCH':
            raise RuntimeError(f'{value!r} unexpected on line {line_num}')
        yield Token(kind, value, line_num, column)

statements = '''
    IF quantity THEN
        total := total + price * quantity;
        tax := price * 0.05;
    ENDIF;
'''

for token in tokenize(statements):
    print(token)
```

دانشگاه شهید مدنی آذربایجان
برنامه نویسی مقدماتی با پایتون

امین گلزاری اسکوئی

۱۴۰۰-۱۴۰۱

Codes and Projects (click here) (https://github.com/Amin-Golzari-Oskouei/Python-Programming-Course-Basic-2021) slides and videos (click here) (https://drive.google.com/drive/folders/1ZsQjBJJ4UAAp9zrGxm3c4qrhnvGBUYHw)