***class* calendar.TextCalendar(*firstweekday=0*)**
This class can be used to generate plain text calendars.

formatmonth(theyear, themonth, w=0, l=0)
Return a month's calendar in a multi-line string.

formatyear(*theyear*, *w=2*, *l=1*, *c=6*, *m=3*)
Return a *m*-column calendar for an entire year as a multi-line string.

Example:

```
import calendar
def main():
    cal=calendar.TextCalendar()
    monthcal=cal.formatmonth(2022,1)
    print(monthcal)
    with open("monthcal.txt","w") as f:
        f.write(monthcal)
    f.close()

    yearcal=cal.formatyear(2022)
    print(yearcal)
    with open("yearcal.txt","w") as f:
        f.write(yearcal)
    f.close()
main()
```

Output:

```
   January 2022
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31


                                2022

      January                   February                  March
```

```
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa
Su
                1  2          1  2  3  4  5  6          1  2  3  4  5  6
 3  4  5  6  7  8  9       7  8  9 10 11 12 13       7  8  9 10 11 12 13
10 11 12 13 14 15 16      14 15 16 17 18 19 20      14 15 16 17 18 19 20
17 18 19 20 21 22 23      21 22 23 24 25 26 27      21 22 23 24 25 26 27
24 25 26 27 28 29 30      28                        28 29 30 31
31

       April                     May                      June
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa
Su
             1  2  3                     1           1  2  3  4  5
 4  5  6  7  8  9 10       2  3  4  5  6  7  8       6  7  8  9 10 11 12
11 12 13 14 15 16 17       9 10 11 12 13 14 15      13 14 15 16 17 18 19
18 19 20 21 22 23 24      16 17 18 19 20 21 22      20 21 22 23 24 25 26
25 26 27 28 29 30         23 24 25 26 27 28 29      27 28 29 30
                          30 31

       July                    August                   September
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa
Su
             1  2  3       1  2  3  4  5  6  7                1  2  3  4
 4  5  6  7  8  9 10       8  9 10 11 12 13 14       5  6  7  8  9 10 11
11 12 13 14 15 16 17      15 16 17 18 19 20 21      12 13 14 15 16 17 18
18 19 20 21 22 23 24      22 23 24 25 26 27 28      19 20 21 22 23 24 25
25 26 27 28 29 30 31      29 30 31                  26 27 28 29 30

      October                  November                 December
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa
Su
                1  2          1  2  3  4  5  6                1  2  3  4
 3  4  5  6  7  8  9       7  8  9 10 11 12 13       5  6  7  8  9 10 11
10 11 12 13 14 15 16      14 15 16 17 18 19 20      12 13 14 15 16 17 18
17 18 19 20 21 22 23      21 22 23 24 25 26 27      19 20 21 22 23 24 25
24 25 26 27 28 29 30      28 29 30                  26 27 28 29 30 31
31

>>>
```

calendar.HTMLCalendar(*firstweekday=0*)
This class can be used to generate HTML calendars.

formatmonth(theyear, themonth, withyear=True)
Return a month's calendar as an HTML table

formatyear(*theyear*, *width=3*)
Return a year's calendar as an HTML table. *width* (defaulting to 3) specifies the number of months per row.

Example:

```
import calendar
def main():
    hcal=calendar.HTMLCalendar() # Creating HTMLCalendar object
    month=hcal.formatmonth(2020,5)
    with open("month.html","w") as f:
        f.write(month)
    year=hcal.formatyear(2019)
    with open("year.html","w") as f:
        f.write(year)

main()
```

Output:
Output is stored in two different files
month.html and year.html

Regular Expressions or re module

This module comes with python software
Regular expression is sequence of characters that form search pattern
Regular expression is used to find patterns within string
Regular expression is used to extract required information from input string
Regular expression is used to validate strings

re modules provide the following functions

1. match
2. search
3. findall

4. split
5. compile

re.match(pattern, string, flags=0)
If zero or more characters at the beginning of string match the regular expression pattern, return a corresponding match object. Return None if the string does not match the pattern;

re.compile(*pattern*, *flags=0*)
Compile a regular expression pattern into a regular expression object, which can be used for matching using its match(), search() and other methods

Example:

```
import re
def main():
    pattern=re.compile("py")
    m=pattern.match("python")
    print(m)
    print(m.span())
    m=pattern.match("guido van rossum develop python")
    print(m)

main()
```

Output:

```
<re.Match object; span=(0, 2), match='py'>
(0, 2)
None
>>>
```

Flags : Flag characters define set rules and regulation for finding or searching pattern

re.I
re.IGNORECASE
Perform case-insensitive matching; expressions like [A-Z] will also match lowercase letters

Example:

```
import re
def main():
    pattern=re.compile('py',re.IGNORECASE)
    m1=pattern.match("python")
    print(m1)
    m2=pattern.match("PYTHON")
    print(m2)
    pattern1=re.compile('py')
    m1=pattern1.match('python')
    m2=pattern1.match('PYTHON')
    print(m1)
    print(m2)
main()
```

Output:

```
<re.Match object; span=(0, 2), match='py'>
<re.Match object; span=(0, 2), match='PY'>
<re.Match object; span=(0, 2), match='py'>
None
>>>
```

Creating patter or regular expression without using compile function

Example:

```
import re
def main():
    m1=re.match(r'py','python')
    print(m1)
    m2=re.match(r'py','father of python is guido van rossum')
    print(m2)
    m3=re.match(r'py','PYTHON',re.I)
    print(m3)

main()
```

Output:

```
<re.Match object; span=(0, 2), match='py'>
None
<re.Match object; span=(0, 2), match='PY'>
```

>>>

re.search(pattern, string, flags=0)
Scan through string looking for the first location where the regular expression pattern produces a match, and return a corresponding match object. Return None if no position in the string matches the pattern.

Example:

```
import re
def main():
    pattern=re.compile("py")
    s=pattern.search("this is python")
    m=pattern.match("this is python")
    print(s)
    print(m)
    s=pattern.search("this is python python")
    print(s)

main()
```

Output:

```
<re.Match object; span=(8, 10), match='py'>
None
<re.Match object; span=(8, 10), match='py'>
>>>
```