ch.12

December 3, 2020

1 Ch. 12: Modules

[18]:
```
#Ch. 12: Exercices
```

#Ex. 1 Calendar Module

Open help for the calendar module.

(a) Try the following:
1 import calendar
2 cal = calendar.TextCalendar() # Create an instance
3 cal.pryear(2012) # What happens here?

(b) Observe that the week starts on Monday. An adventurous CompSci student believes that it is better mental chunking to have his week start on Thursday, because then there are only two working days to the weekend, and every week has a break in the middle. Read the documentation for TextCalendar, and see how you can help him print a calendar that suits his needs.

(c) Find a function to print just the month in which your birthday occurs this year.

(d) Try this:

1 d = calendar.LocaleTextCalendar(6, "SPANISH")
2 d.pryear(2012)

Try a few other languages, including one that doesn't work, and see what happens.

(e) Experiment with calendar.isleap. What does it expect as an argument? What does it return as a result? What kind of a function is this?
Make detailed notes about what you learned from these exercises.

[19]:
```
from unit_tester import test
import calendar
cal = calendar.TextCalendar(firstweekday = 0) # Create an instance
cal.prmonth(2020, 12)

#help(calendar)
```

```
d = calendar.LocaleTextCalendar(6, "GERMAN")
d.pryear(2012)

print(calendar.isleap(2024))
```

```
   December 2020
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
```

```
                                  2012

       Januar                    Februar                    März
So Mo Di Mi Do Fr Sa      So Mo Di Mi Do Fr Sa      So Mo Di Mi Do Fr Sa
 1  2  3  4  5  6  7                1  2  3  4                1  2  3
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       4  5  6  7  8  9 10
15 16 17 18 19 20 21      12 13 14 15 16 17 18      11 12 13 14 15 16 17
22 23 24 25 26 27 28      19 20 21 22 23 24 25      18 19 20 21 22 23 24
29 30 31                  26 27 28 29               25 26 27 28 29 30 31

       April                      Mai                       Juni
So Mo Di Mi Do Fr Sa      So Mo Di Mi Do Fr Sa      So Mo Di Mi Do Fr Sa
 1  2  3  4  5  6  7             1  2  3  4  5                1  2
 8  9 10 11 12 13 14       6  7  8  9 10 11 12       3  4  5  6  7  8  9
15 16 17 18 19 20 21      13 14 15 16 17 18 19      10 11 12 13 14 15 16
22 23 24 25 26 27 28      20 21 22 23 24 25 26      17 18 19 20 21 22 23
29 30                     27 28 29 30 31            24 25 26 27 28 29 30

       Juli                     August                   September
So Mo Di Mi Do Fr Sa      So Mo Di Mi Do Fr Sa      So Mo Di Mi Do Fr Sa
 1  2  3  4  5  6  7                1  2  3  4                      1
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       2  3  4  5  6  7  8
15 16 17 18 19 20 21      12 13 14 15 16 17 18       9 10 11 12 13 14 15
22 23 24 25 26 27 28      19 20 21 22 23 24 25      16 17 18 19 20 21 22
29 30 31                  26 27 28 29 30 31         23 24 25 26 27 28 29
                                                    30

       Oktober                  November                  Dezember
So Mo Di Mi Do Fr Sa      So Mo Di Mi Do Fr Sa      So Mo Di Mi Do Fr Sa
    1  2  3  4  5  6                   1  2  3                      1
 7  8  9 10 11 12 13       4  5  6  7  8  9 10       2  3  4  5  6  7  8
14 15 16 17 18 19 20      11 12 13 14 15 16 17       9 10 11 12 13 14 15
21 22 23 24 25 26 27      18 19 20 21 22 23 24      16 17 18 19 20 21 22
28 29 30 31               25 26 27 28 29 30         23 24 25 26 27 28 29
```

```
                                                    30 31
```

```
True
```

```
3. Investigate the copy module. What does deepcopy do? In which exercises from las chapter woul
```

[21]:
```
#Ex. 3 Copy Module



from copy import deepcopy
import copy

string = ["florin", "echipa"]
b = deepcopy(string)
c = copy.copy(string)
b[1] = ['gege']
c[1] = ['new']

print(string, b)
print(string, c)
```

```
['florin', 'echipa'] ['florin', ['gege']]
['florin', 'echipa'] ['florin', ['new']]
```

4. Create a module named mymodule1.py. Add attributes myage set to your current age, and year set to the current year. Create another module named mymodule2.py. Add attributes myage set to 0, and year set to the year you were born. Now create a file named namespace_test.py. Import both of the modules above and write the following statement:

   1 print((mymodule2.myage - mymodule1.myage) == 2 (mymodule2.year - mymodule1.year))

   When you will run namespace_test.py you will see either True or False as output depending on whether or not you've already had your birthday this year.

   What this example illustrates is that out different modules can both have attributes named myage and year. Because they're in different namespaces, they don't clash with one another. When we write namespace_test.py, we fully qualify exactly which variable year or myage we are referring to.

[22]:
```
#Ex. 4: New Modules with same names

import mymodule1
import mymodule2

print( (mymodule2.myage - mymodule1.myage) == (mymodule2.year - mymodule1.year)␣
↪)
```

```
True
```

```
5. Add the following statement to mymodule1.py, mymodule2.py, and
```

namespace_test.py from the previous exercise:

```
1 print("My name is", __name__)
```

Run namespace_test.py. What happens? Why? Now add the following to the bottom of mymodule1.py:

```
1 if __name__ == "__main__": <br>
2 print("This won't run if I'm imported.") <br>
```

Run mymodule1.py and namespace_test.py again. In which case do you see the new print statement?

[23]:
```
#Ex. 5

print("My name is", __name__)
```

```
My name is __main__
```

7. Give the Python interpreter's response to each of the following from a continuous interpret session:

```
1 s = "If we took the bones out, it wouldn't be crunchy, would it?" <br>
2 s.split() <br>
3 type(s.split()) <br>
4 s.split("o") <br>
5 s.split("i") <br>
6 "0".join(s.split("o")) <br>
```

Be sure you understand why you get each result. Then apply what you have learned to fill in the body of the function below using the split and join methods of str objects:

```
1 def myreplace(old, new, s): <br>
2 """ Replace all occurrences of old with new in s. """ <br>
3 ... <br>
4 <br>
5 <br>
6 test(myreplace(",", ";", "this, that, and some other thing") == <br>
7 "this; that; and some other thing") <br>
8 test(myreplace(" ", "**", <br>
9 "Words will now be separated by stars.") == <br>
10 "Words**will**now**be**separated**by**stars.") <br>
Your solution should pass the tests. <br>
```

[25]:
```
#Ex. 7


s = "If we took the bones out, it wouldn't be crunchy, would it?"
```

```
s.split()
type(s.split())
s.split("o")
s.split("i")
"0".join(s.split("o"))

#Tests

def myreplace(old, new, s):
    """ Replace all occurrences of old with new in s. """
    return new.join(s.split(old))
```

[26]:
```
test(myreplace(",", ";", "this, that, and some other thing") ==
"this; that; and some other thing")
test(myreplace(" ", "**",
"Words will now be separated by stars.") ==
"Words**will**now**be**separated**by**stars.")
```

```
Test at line 1 ok.
Test at line 3 ok.
```

[ ]:
```

```