Done

Tutorial 4

Further Data Types

Introduction

The purpose of this practical is to continue using Python to develop programs, making use of more data types. Don't forget that you are required to submit the indicated exercise through myLearn for assessment.

Data types

Computation really requires both the instructions that specify how the computation should be performed and the data over which the computation should be performed. This week we examined some more complicated data types, including a further look at strings, and the use of dictionaries and tuples. This week's exercises will use each of these data types.

Exercises

Use material from this week's lectures to perform the following exercises:

1. The Python project hosts a Library Reference that describes the standard library included with Python. This library details the built-in modules that are included in Python. Browse through the library to see the type of information it includes. Pay particular attention to the string methods specified at https://docs.python.org/3/library/stdtypes.html#string-methods
2. The following functions are all intended to check whether a string contains any lowercase letters, but at least some of them are wrong. For each function, describe what the function actually does:

```
def any_lowercase1(s):
  for c in s:
    if c.islower()
      return True
    else:
      return False
```

```
def any_lowercase2(s):
  for c in s:
    if 'c'.islower():
      return 'True'
    else:
      return 'False'
```

```
def any_lowercase3(s):
  for c in s:
    flag = c.islower()
  return flag
```

```
def any_lowercase4(s):
  flag = False
  for c in s:
    flag = flag or c.islower()
  return flag
```

```
def any_lowercase5(s):
  for c in s:
    if not c.islower():
      return False
  return True
```

3. Create a dictionary named `available`, initialised with the following values:

```
"pen": 3
"pencil": 5
"eraser": 2
"paper": 10
```

The available dictionary specifies the number of each item that is available. Write a program that loops through all the keys in the `available` dictionary to help determine the total number of items available (e.g. in the given case, there are 3 pens, 5 pencils, 2 erasers, and 10 papers, so your program should output 20). Your program should display a count of the total number of items specified in the `available` dictionary even if you modify the keys/values in the dictionary.

4. **[This exercise should be submitted through the Assessments section of myLearn for Tutorial Exercise 4]** Create a dictionary named `weight`, initialised with the following values:

```
"pencil": 10
"pen": 20
"paper": 4
"eraser": 80
```

Create another dictionary named `available`, initialised with the following values:

```
"pen": 3
"pencil": 5
"eraser": 2
"paper": 10
```

If the `weight` dictionary gives the weight of an individual item of each type, and the `available` dictionary specifies the number of each item that is available, write code that determines the total weight of all available items (i.e. what is the overall weight of all the pens, pencils, paper, and erasers?)

5. Typically if a program is given a particular set of input, we always want it to output the same result. These programs are said to be deterministic. However, sometimes we want the computer to be a bit more unpredictable. It is actually quite difficult to make a program truly nondeterministic, but there are ways we can make it at least seem unpredictable. One way is to use pseudorandom numbers. These are numbers that are not truly random, but are close enough for many purposes. Python includes a pseudorandom number generator in its *random* module. To use it, add `import random` to the beginning of your program, then call `random.random()`:

```
import random

for i in range(10):
    random_value = random.random()
    print(random_value)
```

Note: Be careful not to name your file `random.py` - if you do, then when you try to `import random`, it will import your file rather than the built-in Python library.

You can also select random integers using `random.randint()` or random elements using `random.choice()`. Examine the documentation for the random module and write a program that selects a random integer between 1 and 20, and continually asks the user to guess a value until they get it correct, offering hints as to whether any incorrect guess was too high or too low.

Last modified: Thursday, 1 February 2024, 10:52 AM