

Tutorial 4

Further Data Types

Exercise Answers

1. The Python project hosts a [Library Reference](https://docs.python.org/3/library/stdtypes.html#string-methods) 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
```

The above function checks to see if the first character of the string is lowercase.

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

The above function always returns **True** if a non-empty string is passed in, because the character 'c' is always lowercase. If the parameter is an empty string, the result is **None**

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

The above function overwrites the value of flag for each character, so only returns **True** if the last character in the string is lowercase.

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

The above function checks to see if the string contains at least one lowercase character.

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

The above function checks to see if the entire string is lowercase.

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.

```
available = {"pen": 3, "pencil": 5, "eraser": 2, "paper": 10}
total = 0
for key in available:
    total += available[key]
print(total)
```

4. [This exercise should be submitted through the Assessments section of Moodle 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.

```
import random

target = random.randint(1, 20)

guess = 0
while guess != target:
    guess = int(input("Enter a guess between 1 and 20 (inclusive): "))
    if guess > target:
        print("You guessed too high!")
    elif guess < target:
        print("You guessed too low!")
    else:
        print("Congratulations!")
```

Last modified: Thursday, 1 February 2024, 10:45 AM