IT Scripting and Automation

More Python Basics

Lecturer: Art Ó Coileáin

Python Exercise

- Game: “guess the number”
- The output should look like this:

I am thinking of a number between 1 and 20.

Take a guess.

10

Your guess is too low.

Take a guess.

17

Your guess is too high.

Take a guess

16

Good Job ! You guessed my number in 3 guesses !

Python Exercise

- Hints:
  - Use import *randomand random.randint(1,20)* to generate the random number
  - Use a for loop to allow the player to guess 6 times.

Python Basics

- The *List datatype*: a list is a value that contains multiple values in an **ordered** sequence. The term list value refers to the list itself not the values inside the list value. A list looks like this: [‘cat’,’bat’,’rat’]

Example:

In [1]: [1,2,3]

Out [1]: [1, 2, 3]

In [2]: a = [‘cat’,’bat’,’rat’,[1,2,3]]

Out [2]: [‘cat’, ’bat’, ’rat’,[1,2,3]]

In [3]: a[0]

Out [3]: ‘cat’

In [4]: a[3][2]

Out [4]: 3

Lists

- The *len()* function will return the number of values that are in a list value passed to it, just like it can count the number of characters in a string value.

Example:

```
>>> spam = ['cat', 'dog', 'moose']
>>> len(spam)
3
```

Examples of List Concatenation and list replication:

```
>>> [1, 2, 3] + ['A', 'B', 'C']
[1, 2, 3, 'A', 'B', 'C']
>>> ['X', 'Y', 'Z'] * 3
['X', 'Y', 'Z', 'X', 'Y', 'Z', 'X', 'Y', 'Z']
>>> spam = [1, 2, 3]
>>> spam = spam + ['A', 'B', 'C']
>>> spam
[1, 2, 3, 'A', 'B', 'C']
```

Lists

- Removing Values from List with del statement

Example:

```
>>> spam = ['cat', 'bat', 'rat', 'elephant']
>>> del spam[2]
>>> spam
['cat', 'bat', 'elephant']

>>> del spam[2]
>>> spam
['cat', 'bat']
```

Example (allMyCats.py)

Example of a script using list concatenation:

```
#!/usr/bin/env python
catNames= []
while True:
        print('Enter the name of cat ' + str(len(catNames) + 1) +' (Or enter nothing to stop.):')
        name = raw_input()
        if name == '':
                break
        catNames= catNames+ [name] # list concatenation
print('The cat names are:')
for name in catNames:
        print(' ' + name)
```

Lists and inserts

Example of list interating through a list :

```
>>> supplies = ['pens', 'staplers', 'flame-throwers', 'binders']
>>> for i in range(len(supplies)):
        print('Index ' + str(i) + ' in supplies is: ' + supplies[i])
```

insert() method

- The *insert()* method can insert a value at any index in the list.

Example:

```
>>> spam = ['cat', 'dog', 'bat']
>>> spam.insert(1, 'chicken')

>>> spam
['cat', 'chicken', 'dog', 'bat']
```

Lists and remove

- The ***remove()*** method is passed the value to be removed from the list it is called on.

Example:

```
>>> spam = ['cat', 'bat', 'rat', 'elephant']
>>> spam.remove('bat')
>>> spam
['cat', 'rat', 'elephant']
```

NB:

- If the value appears multiple times in the list, **only the first instance** of the value will be removed.

```
>>> spam = ['cat', 'bat', 'rat', 'cat', 'hat', 'cat']
>>> spam.remove('cat')
>>> spam
['bat', 'rat', 'cat', 'hat', 'cat']
```

Lists and sort()

- Lists of number values or lists of strings can be sorted with the ***sort()*** method.

Examples:

```
>>> spam = [2, 5, 3.14, 1, -7]
>>> spam.sort()
>>> spam
[-7, 1, 2, 3.14, 5]

>>> spam = ['ants', 'cats', 'dogs', 'badgers', 'elephants']
>>> spam.sort()
>>> spam
['ants', 'badgers', 'cats', 'dogs', 'elephants']

>>> spam.sort(reverse=True)
>>> spam
['elephants', 'dogs', 'cats', 'badgers', 'ants']
```

Tuple Data Type

- The ***tuple*** data type is almost identical to the list data type, except in two ways:
  - Tuples are typed with parentheses, (and), (instead of square brackets used in lists i.e. [and].
  - Tuples are immutable.

Example:

```
>>> eggs = ('hello', 42, 0.5)
>>> eggs[0]
'hello'
>>> eggs[1:3]
(42, 0.5)
>>> len(eggs)
3
```

Tuple Data Type

- Converting Types with the list() and tuple() Functions.

Example:

```
>>> tuple(['cat', 'dog', 5])
('cat', 'dog', 5)

>>> list(('cat', 'dog', 5))
['cat', 'dog', 5]

>>> list('hello')
['h', 'e', 'l', 'l', 'o']
```

The Dictionary Data Type

- A *dictionary* is a collection of many values.
- Unlike indexes for lists, indexes for dictionaries can use many different data types, not just integers.
- Indexes for dictionaries are called keys, and a key with its associated value is called a key-value pair.
- In code, a dictionary is typed with braces, {}.

Example:

```
>>> myCat= {'size': 'fat', 'color': 'grey', 'disposition': 'loud'}
>>> myCat['size']
'fat‘

>>> 'My cat has ' + myCat['color'] + ' fur.'
'My cat has greyfur.'
```

Dictionaries vs. Lists

- Unlike lists, items in dictionaries are unordered.
- There is no “first” item in a dictionary (*unlike the first item in a list named spam would be spam[0]*).

Example:

```
>>> spam = ['cats', 'dogs', 'moose']
>>> bacon = ['dogs', 'moose', 'cats']
>>> spam == bacon
False
>>> eggs = {'name': 'Zophie', 'species': 'cat', 'age': '8'}
>>> ham = {'species': 'cat', 'age': '8', 'name': 'Zophie'}
>>> eggs == ham
True
```

Python Exercise (birthdays.py)

Example of a python script using a dictionary for names and birthdays:

```
#!/usr/bin/env python
birthdays = {'Alice': 'Apr 1', 'Bob': 'Dec 12', 'Carol': 'Mar 4'}
while True:
        print('Enter a name: (blank to quit)')
        name = raw_input()
        if name == '':
                break
        if name in birthdays:
                print(birthdays[name] + ' is the birthday of ' + name)
        else:
                print('I do not have birthday information for ' + name)
                print('What is their birthday?')
                bday= raw_input()
                birthdays[name] = bday
                print('Birthday database updated.')
```

The keys(), values(), and items() Methods

- There are three dictionary methods that will return list-like values of the dictionary's keys, values, or both keys and values:
  - keys(),
  - values(),
  - items().

Example:

```
>>> spam = {'colour': 'red', 'age': 42}
>>> for v in spam.values():
        print(v)
```

Output:

```
red
42
```

Continues…

The keys(), values(), and items() Methods

```
>>> for k in spam.keys():
        print(k)
```

Output:
```
colour
age
```

```
>>> for i in spam.items():
        print(i)
```

Output:
```
('colour', 'red')
('age', 42)
```

- Using the *keys()*, *values()*, and *items()* methods, a for loop can iterate over the keys, values, or key-value pairs in a dictionary, respectively.
- Notice that the values in the *dict_items* value returned by the *items()* method are ***tuples*** of the **key** and **value**.

Continues…

The keys(), values(), and items() Methods

- If you want a true list from one of these methods, pass its list-like return value to the *list()* function.

```
>>> spam = {'colour': 'red', 'age': 42}
>>> spam.keys()
```

Output (*python3*):

```
dict_keys(['colour', 'age'])
```

```
>>> list(spam.keys())
```

Output:

```
['colour', 'age']
```

- The *list(spam.keys())* line takes the ***dict_keys*** value returned from keys() and passes it to *list()*, which then returns a list value of ['color', 'age'].

The keys(), values(), and items() Methods

- You can also use the multiple assignment trick in a for loop to assign the key and value to separate variables:

Example:

```
>>> spam = {'colour': 'red', 'age': 42}
>>> for k, v in spam.items():
        print('Key: ' + k + ' Value: ' + str(v))
```

Output:

```
Key: colour Value: red
Key: age Value: 42
```

The get() Method

- It's tedious to check whether a key exists in a dictionary before accessing that key's value. Fortunately, dictionaries have a *get()* method that takes two arguments: the key of the value to retrieve and a fall back value to return if that key does not exist.

Example:

```
>>> spam = {'name': 'Zophie', 'age': 7}
>>> 'name' in spam.keys()
True
>>> 'Zophie' in spam.values()
True
>>> 'color' in spam.keys()
False
>>> 'color' not in spam.keys()
True
```


Next page……

The get() Method

```
>>> picnicItems= {'apples': 5, 'cups': 2}
>>> 'I am bringing ' + str(picnicItems.get('cups', 0)) + ' cups.'
'I am bringing 2 cups.'
>>> 'I am bringing ' + str(picnicItems.get('eggs', 0)) + ' eggs.'
'I am bringing 0 eggs.'
```

- Because there is no 'eggs' key in the *picnicItems* dictionary, the default value 0 is returned by the *get()* method.

The setdefault() Method

- You'll often have to set a value in a dictionary for a certain key only if that key does not already have a value.

Example:

```
spam = {'name': 'Pooka', 'age': 5}
if 'color' not in spam:
spam['color'] = 'black‘
```

- The *setdefault()* method offers a way to do this in one line of code.
- The first argument passed to the method is the key to check for, and the second argument is the value to set at that key if the key does not exist. If the key does exist, the *setdefault()* method returns the key's value.

Next page……

The setdefault() Method

Example:

```
>>> spam = {'name': 'Pooka', 'age': 5}
>>> spam.setdefault('color', 'black')
'black‘

>>> spam
{'color': 'black', 'age': 5, 'name': 'Pooka'}

>>> spam.setdefault('color', 'white')
'black'

>>> spam
{'color': 'black', 'age': 5, 'name': 'Pooka'}
```

The isXString Methods

- There are several methods that have names beginning with the work **is**. These methods return a Boolean value that describes the nature of the string. The most common *isX* methods are:
- **isalpha()** returns True if the string consists only of letters and is not blank.
- **isalnum()** returns True if the string consists only of letters and numbers and is not blank.
- **isdecimal()** returns True if the string consists only of numeric characters and is not blank.
- **isspace()** returns True if the string consists only of spaces, tabs, and newlines and is not blank.
- **istitle()** returns True if the string consists only of words that begin with an uppercase letter followed by only lowercase letters.
- **isupper()** and **islower()** methods will return a Boolean True value if the string has at least one letter and all the letters are uppercase or lowercase, respectively.

The isupper()/islower() Methods

Examples:

```
>>> spam = 'Hello world!'

>>> spam.islower()
False

>>> spam.isupper()
False

>>> 'HELLO'.isupper()
True

>>> 'abc12345'.islower()
True

>>> '12345'.islower()
False

>>> '12345'.isupper()
False
```

The isXMethods

Examples:

```
>>> 'hello'.isalpha()
True
>>> 'hello123'.isalpha()
False
>>> 'hello123'.isalnum()
True
>>> 'hello'.isalnum()
True
>>> '123'.isdecimal()
True
>>> ' '.isspace()
True
```

The istitle() Method

Examples:

```
>>> 'This Is Title Case'.istitle()
True

>>> 'This Is Title Case 123'.istitle()
True

>>> 'This Is not Title Case'.istitle()
False

>>> 'This Is NOT Title Case Either'.istitle()
False
```