Programming with Python

Lesson 5: Dictionaries and Tuples!

November 29th, 2016

Last week's goals

Last week's goals

- We learnt about lists

Last week's goals

- We learnt about lists
- We learnt about how we can iterate on lists with a for loop

Last week's goals

- We learnt about lists
- We learnt about how we can iterate on lists with a for loop
- We learnt about some list operations

Last week's goals

- We learnt about lists
- We learnt about how we can iterate on lists with a for loop
- We learnt about some list operations
- We begun writing our own text based game!

A flashback to lists

Lists - A collection of anything you want

A flashback to lists

Lists - A collection of anything you want
Can be resized

A flashback to lists

Lists - A collection of anything you want
Can be resized
Elements can be changed when we want

Tuples

Tuples - A collection of anything you want

Tuples

Tuples - A collection of anything you want
Cannot be resized

Tuples

Tuples - A collection of anything you want

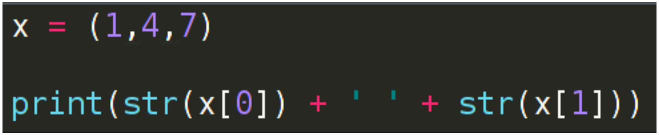
Cannot be resized

Once made, elements cannot be changed - You will not be able to do for example x[0] = 4.

Tuples

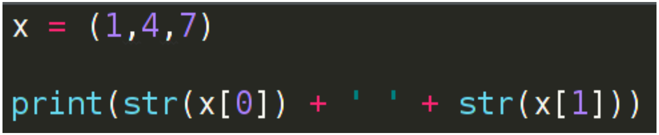
Tuples - A collection of anything you want
Cannot be resized
Once made, elements cannot be changed - You will not be able to do for example $x[0] = 4$. This means tuples are *immutable*.

Tuple syntax

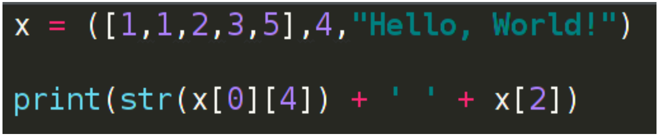


```
x = (1,4,7)

print(str(x[0]) + ' ' + str(x[1]))
```

Tuple syntax



```
x = (1,4,7)

print(str(x[0]) + ' ' + str(x[1]))
```



```
x = ([1,1,2,3,5],4,"Hello, World!")

print(str(x[0][4]) + ' ' + x[2])
```

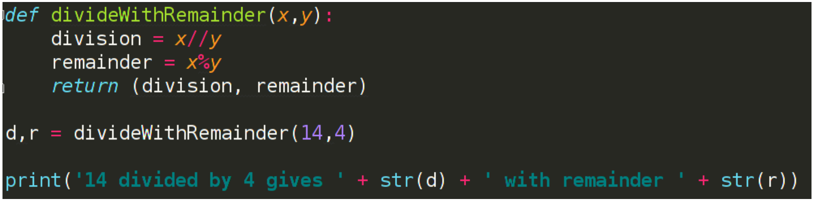
What’s the point?

As tuples are immutable, they are much faster.

What's the point?

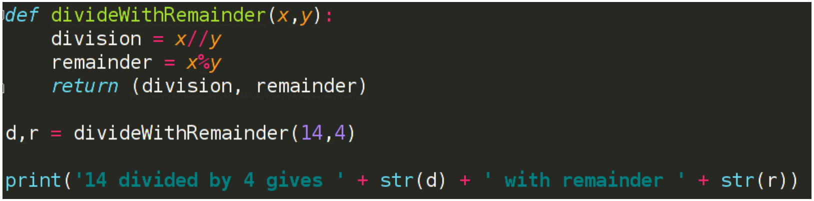
As tuples are immutable, they are much faster.
We can also use tuples if we want to return multiple values from a function.

Example involving functions



```
def divideWithRemainder(x,y):
    division = x//y
    remainder = x%y
    return (division, remainder)

d,r = divideWithRemainder(14,4)

print('14 divided by 4 gives ' + str(d) + ' with remainder ' + str(r))
```

Example involving functions



```python
def divideWithRemainder(x,y):
    division = x//y
    remainder = x%y
    return (division, remainder)

d,r = divideWithRemainder(14,4)

print('14 divided by 4 gives ' + str(d) + ' with remainder ' + str(r))
```

Note the neat syntax on the left of the equals sign. You can kind of 'map' variables to the locations in a tuple.

What is a dictionary

Dictionaries are big 'lists' of key-value pairs.

What is a dictionary

Dictionaries are big 'lists' of key-value pairs.
Keys can be ints, floats, strings, or tuples. Values can be absolutely anything.

What is a dictionary

Dictionaries are big 'lists' of key-value pairs.
Keys can be ints, floats, strings, or tuples. Values can be absolutely anything.
They have plenty of uses, including:

- Storing place name to coordinates for a GPS tracking app

What is a dictionary

Dictionaries are big 'lists' of key-value pairs.
Keys can be ints, floats, strings, or tuples. Values can be absolutely anything.
They have plenty of uses, including:

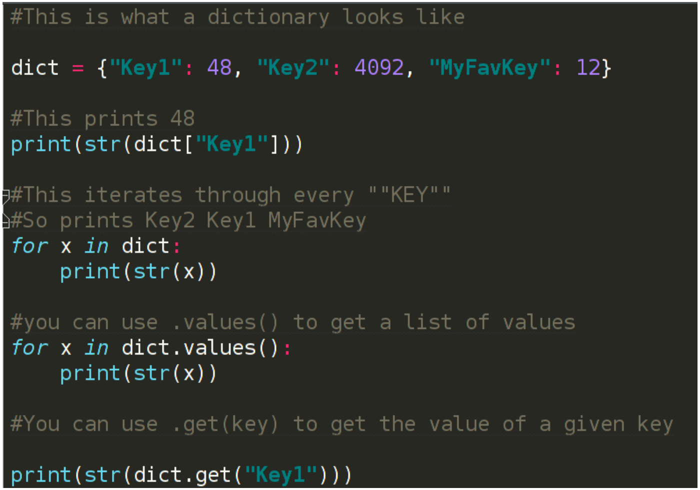
- Storing place name to coordinates for a GPS tracking app
- Storing weapon names to weapon stats for a game

What is a dictionary

Dictionaries are big 'lists' of key-value pairs.
Keys can be ints, floats, strings, or tuples. Values can be absolutely anything.
They have plenty of uses, including:

- Storing place name to coordinates for a GPS tracking app
- Storing weapon names to weapon stats for a game
- Translating something from english to german

An example of a dictionary



```
#This is what a dictionary looks like

dict = {"Key1": 48, "Key2": 4092, "MyFavKey": 12}

#This prints 48
print(str(dict["Key1"]))

#This iterates through every ""KEY""
#So prints Key2 Key1 MyFavKey
for x in dict:
    print(str(x))

#you can use .values() to get a list of values
for x in dict.values():
    print(str(x))

#You can use .get(key) to get the value of a given key

print(str(dict.get("Key1")))
```

Objects & classes

Objects are ways to store more advanced structures of data

Objects & classes

Objects are ways to store more advanced structures of data
An object is a physical version of a class - like if an object were a car, it's class would be it's blueprints

Objects & classes

Objects are ways to store more advanced structures of data
An object is a physical version of a class - like if an object were a car, it's class would be it's blueprints
There's a tonne of stuff on objects and classes online, though they are not for the faint of heart!

Time for more games!

presentations!

That's all for tonight

To summarise:

That's all for tonight

To summarise:

- We learnt about tuples and their uses in returning multiple values

That's all for tonight

To summarise:

- We learnt about tuples and their uses in returning multiple values
- We learnt about dictionaries

That's all for tonight

To summarise:

- We learnt about tuples and their uses in returning multiple values
- We learnt about dictionaries
- We discussed objects

That's all for tonight

To summarise:

- We learnt about tuples and their uses in returning multiple values
- We learnt about dictionaries
- We discussed objects
- We finished writing our own text based game!

For next week

Source code plus lecture slides will be available online soon after the lesson.
If you are new to HackSocNotts, please join us on *http://hacksocnotts.slack.com*.
If you have any questions, feel free to ask now or over slack.