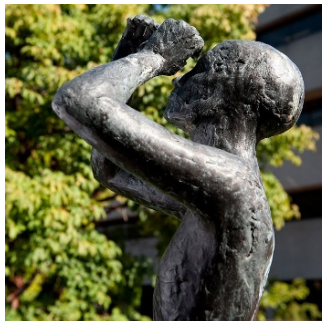


CS1117 – Introduction to Programming

Dr. Jason Quinlan,

School of Computer Science and Information Technology

A TRADITION OF INDEPENDENT THINKING

UCC

University College Cork, Ireland

Coláiste na hOllscoile Corcaigh

Announcements

CS1117 DSA students

I could not find an alternative lab last week

So I have extended the submit for Lab 2 by an extra week for everyone ☺

Announcements

Lab 2

Error in the lab 2 lecture 6 sheet

I ask you to create a folder called *functions* and place *functions*.py in this folder

On some versions of PyCharm, when you import *functions*. Python tries to import its library file *functions.py*

So, we need to call the folder we create *my_functions*

Then everything works fine. This is not an issue in Atom, where I tested the code. Apologies…

Announcements

Concepts covered so far:

In the second session today, I will go over everything we have done so far

More over, I will go over coding of the Lab 1 submission

And then focus on functions, formatting, and if/else/elif, and anything else you want to cover…

So a long live coding session…

Sound good??

Announcements

Lab 1 submission

I have uploaded videos of me completing the lab 1 assignments

Video should include audio, so if can't hear the audio, let me know

I'll try and do this for every lab submission

Announcements

Extra coding class

Going forward I plan on adding an extra class

on either Tuesday or Thursday

It will be in one of the smaller labs, where any CS1117 student can come and ask me questions

I'll go over anything and everything covered so far

Attendance is not mandatory, no sign in will be taken, it's just for you to ask me anything that you are unsure of.

Announcements

Extra coding class

As part of this class, I have uploaded a Google form to Canvas under Modules - Week 4

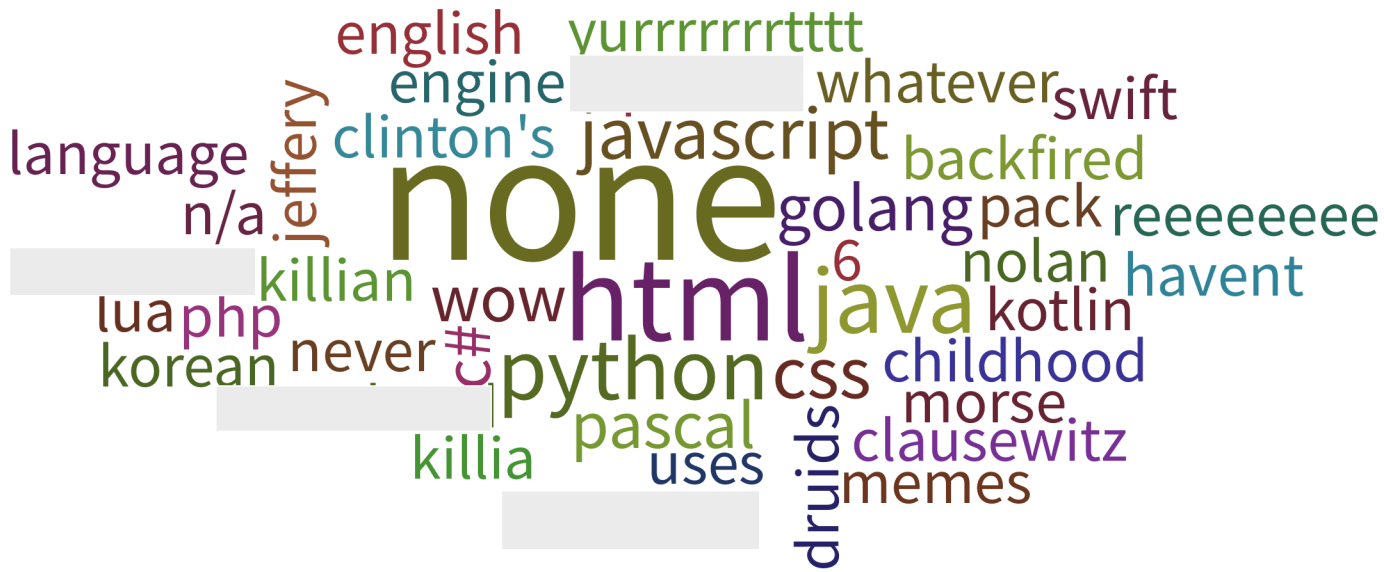
You can select concepts we have covered and give some feedback on what you would like me to cover

The form is anonymous…. but….

Announcements

Remember this from week 1??

What other language have you coded in?

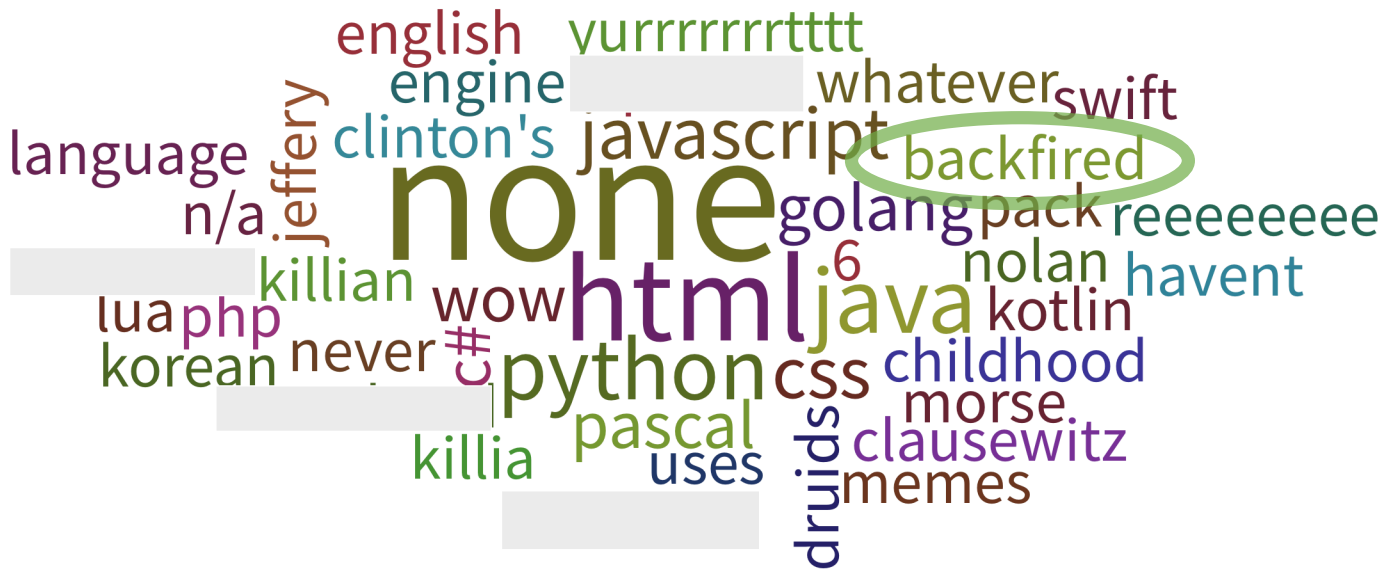


UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Announcements

Remember this from week 1??

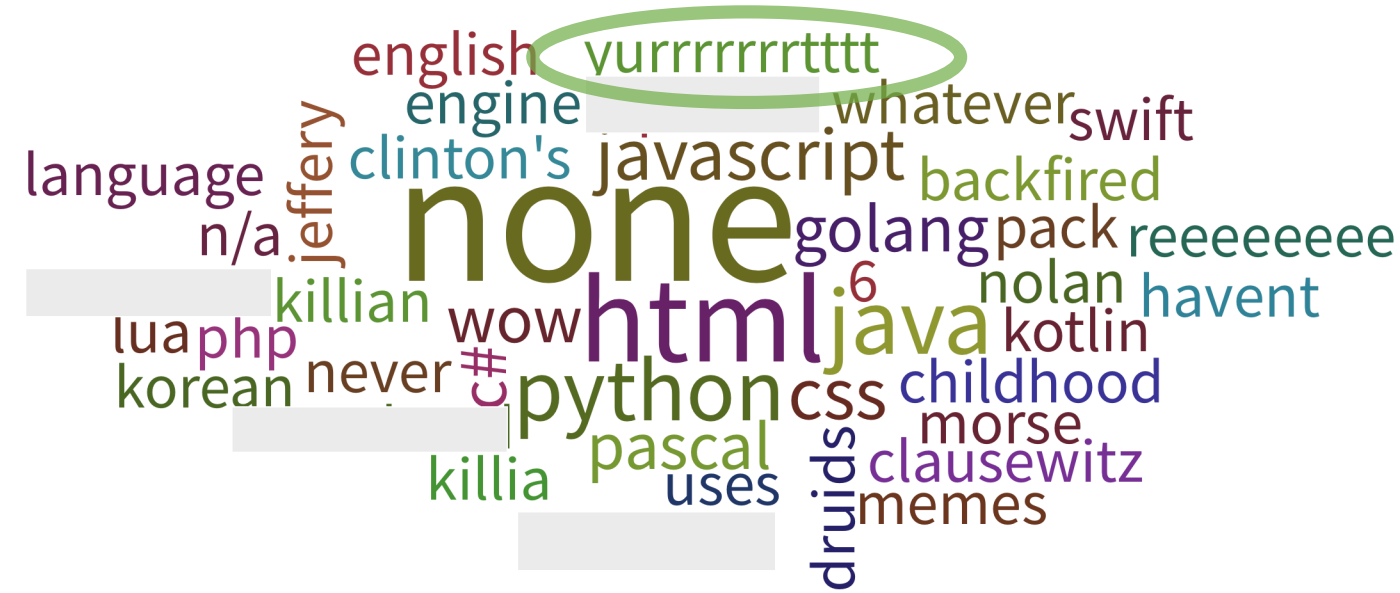
What other language have you coded in?



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Announcements

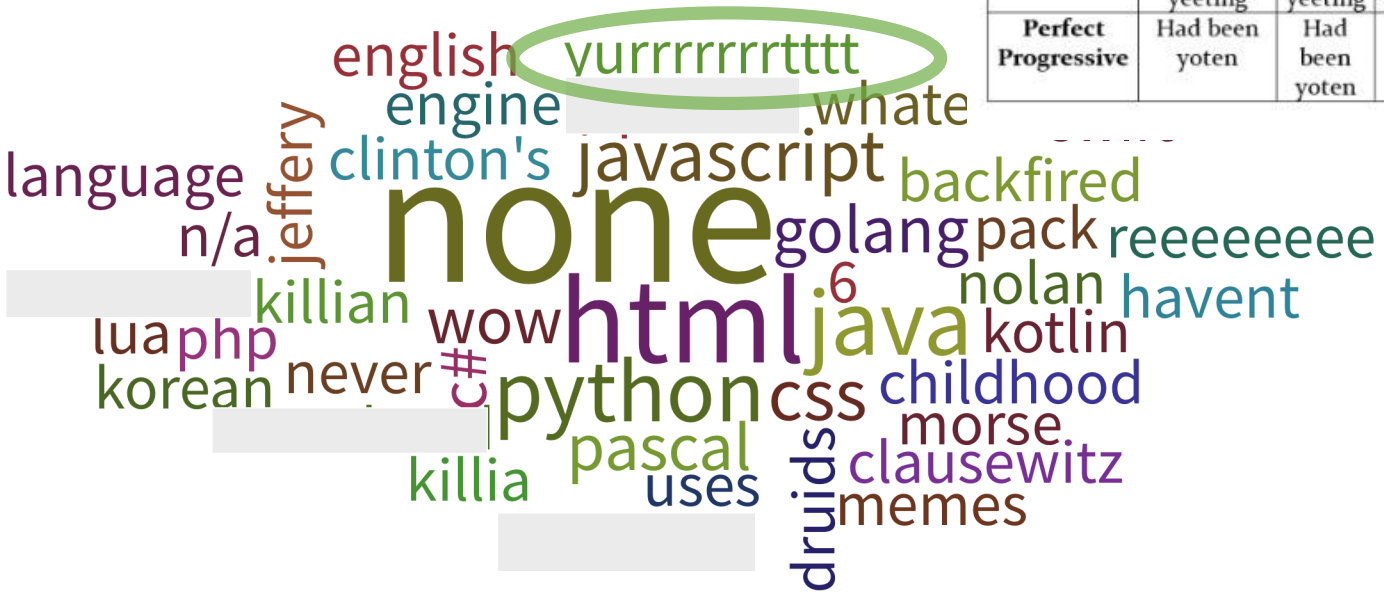
Remember this from week 1??



Announcemer

Remember this from week 1

What other language have you



VERB INFLECTION – TO YEET

	SINGULAR			PLURAL		
	First	Second	Third	First	Second	Third
Indicative	Yeet	Yeet	Yeets	Yeet	Yeet	Yeet
Subjunctive	That I yeet	That you yeet	That he yeets	That we yeet	That you all yeet	That they yeet
Optative	Yeeten	Yeeten	Yeetens	Yeeten	Yeeten	Yeeten
Imperative		Yeet!	Yeet!	Yeet!	Yeet!	Yeet!
Participle	Yeeting	Yeeting	Yeeting	Yeeting	Yeeting	Yeeting
FUTURE	Will yeet	Will yeet	Will yeet	Will yeet	Will yeet	Will yeet
PAST						
Perfect	Had yoten	Had yoten	Had yoten	Had yoten	Had yoten	Had yoten
Simple	Yote	Yote	Yote	Yote	Yote	Yote
Habitual	Used to yeet	Used to yeet	Used to yeet	Used to yeet	Used to yeet	Used to yeet
Imperfect	Was yeeting	Were yeeting	Was yeeting	Were yeeting	Were yeeting	Were yeeting
Perfect Progressive	Had been yoten	Had been yoten	Had been yoten	Had been yoten	Had been yoten	Had been yoten

Loki: yeet

Thor: ??

Loki: yoot

Loki: yotun

Loki: yute

Loki: yeeten

Loki: yate

Loki: yeeth

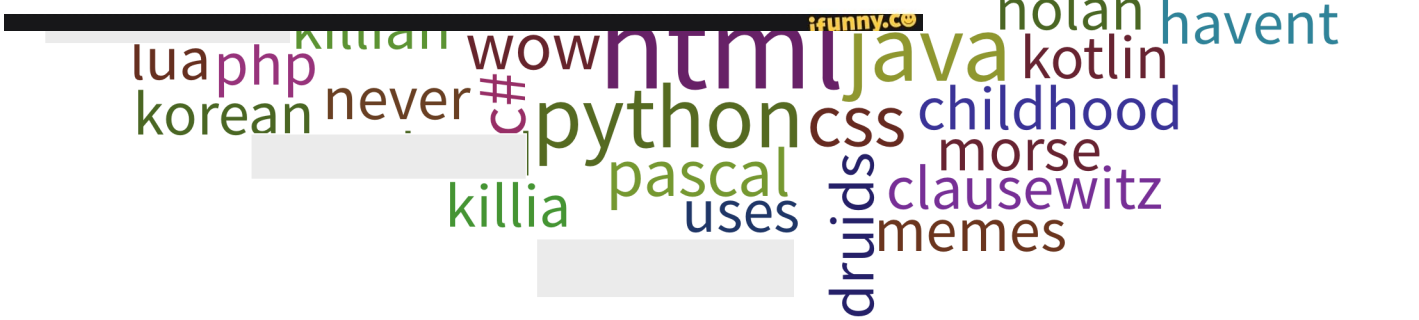
Loki: yeeted

Thor: You stop that right now

Peter Parker, an intellectual: No let him finish

VERB INFLECTION – TO YEET

	SINGULAR			PLURAL		
	First	Second	Third	First	Second	Third
Indicative	Yeet	Yeet	Yeets	Yeet	Yeet	Yeet
Subjunctive	That I yeet	That you yeet	That he yeets	That we yeet	That you all yeet	That they yeet
Optative	Yeeten	Yeeten	Yeetens	Yeeten	Yeeten	Yeeten
Imperative		Yeet!	Yeet!	Yeet!	Yeet!	Yeet!
Participle	Yeeting	Yeeting	Yeeting	Yeeting	Yeeting	Yeeting
FUTURE	Will yeet	Will yeet	Will yeet	Will yeet	Will yeet	Will yeet
PAST						
Perfect	Had yoten	Had yoten	Had yoten	Had yoten	Had yoten	Had yoten
Simple	Yote	Yote	Yote	Yote	Yote	Yote
Habitual	Used to yeet	Used to yeet	Used to yeet	Used to yeet	Used to yeet	Used to yeet
Imperfect	Was yeeting	Were yeeting	Was yeeting	Were yeeting	Were yeeting	Were yeeting
Perfect Progressive	Had been yoten	Had been yoten	Had been yoten	Had been yoten	Had been yoten	Had been yoten



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Loki: yeet

Thor: ??

Loki: yoot

Loki: yotun

Loki: yute

Loki: yeeten

Loki: yate

Loki: yeeth

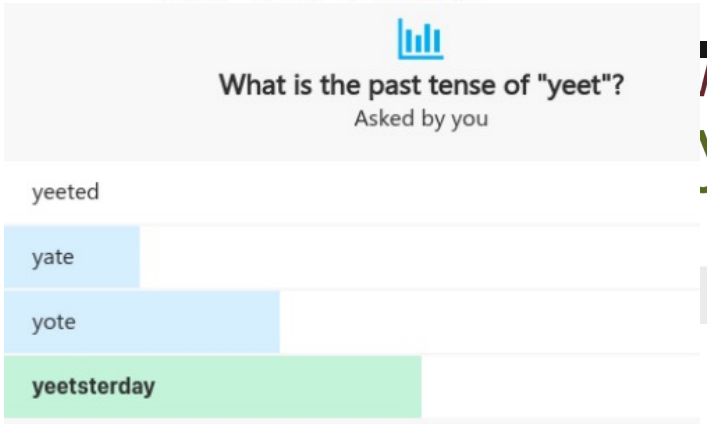
Loki: yeeted

Thor: You stop that right now

Peter Parker, an intellectual: No let him finish

VERB INFLECTION – TO YEET

	SINGULAR			PLURAL		
	First	Second	Third	First	Second	Third
Indicative	Yeet	Yeet	Yeets	Yeet	Yeet	Yeet
Subjunctive	That I yeet	That you yeet	That he yeets	That we yeet	That you all yeet	That they yeet
Optative	Yeeten	Yeeten	Yeetens	Yeeten	Yeeten	Yeeten
Imperative		Yeet!	Yeet!	Yeet!	Yeet!	Yeet!
Participle	Yeeting	Yeeting	Yeeting	Yeeting	Yeeting	Yeeting
FUTURE	Will yeet	Will yeet	Will yeet	Will yeet	Will yeet	Will yeet
PAST						
Perfect	Had yoten	Had yoten	Had yoten	Had yoten	Had yoten	Had yoten
Simple	Yote	Yote	Yote	Yote	Yote	Yote
Habitual	Used to yeet	Used to yeet	Used to yeet	Used to yeet	Used to yeet	Used to yeet
Imperfect	Was yeeting	Were yeeting	Was yeeting	Were yeeting	Were yeeting	Were yeeting
Perfect Progressive	Had been yoten	Had been yoten	Had been yoten	Had been yoten	Had been yoten	Had been yoten



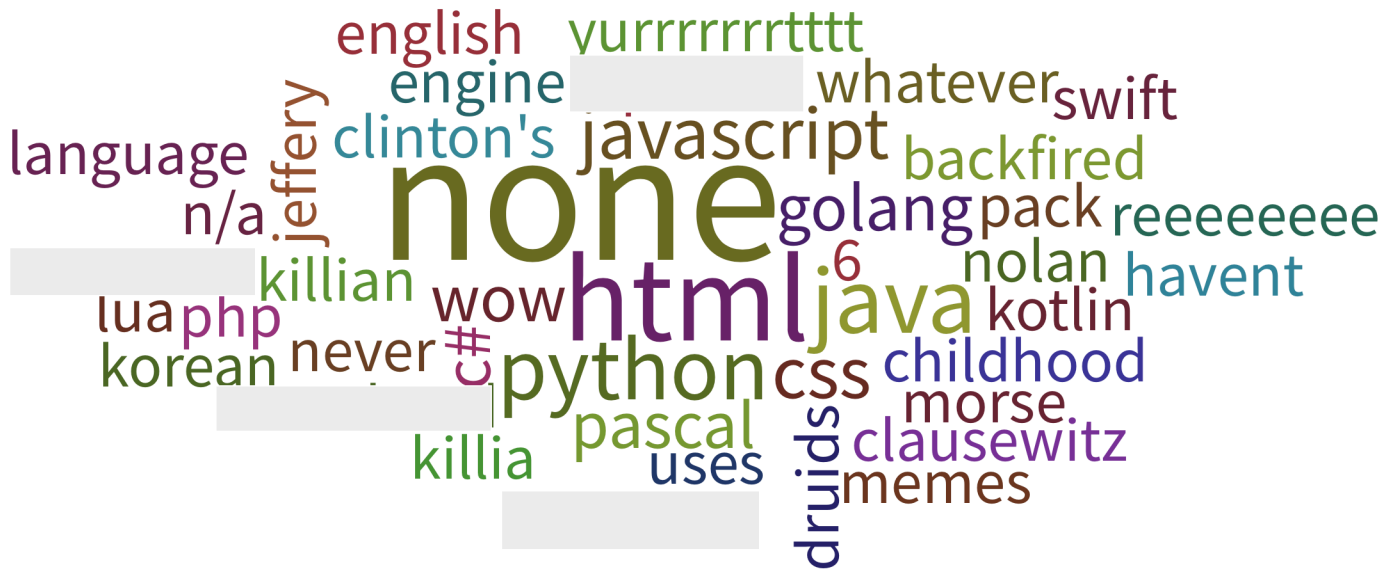
me_irl

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Announcements

Remember this from week 1??

What other language have you coded in?



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Announcements

Extra coding class

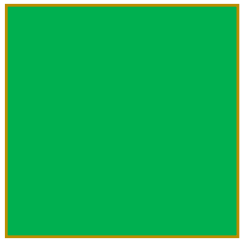
As part of this class, I have uploaded a Google form to Canvas under Modules - Week 4

You can select concepts we have covered and give some feedback on what you would like me to cover

The form is anonymous…. but….

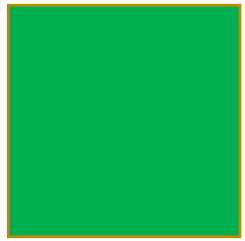
It is anonymous to me, but not to Google, so answer the questions asked and no more ☺

Tuple recap



We can now create tuples – 2 different ways

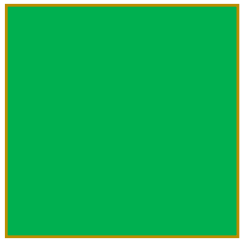
Tuple recap



We can now create tuples – 2 different ways

We can get a value based on index number – tuple[index]

Tuple recap

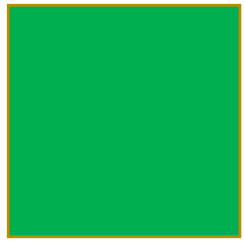


We can now create tuples – 2 different ways

We can get a value based on index number – tuple[index]

We can get the length of a tuple – len(tuple)

Tuple recap



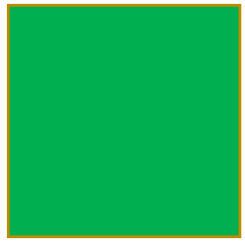
We can now create tuples – 2 different ways

We can get a value based on index number – tuple[index]

We can get the length of a tuple – len(tuple)

We can't add to a tuple – immutable – tuple[4] = value

Tuple recap



We can now create tuples – 2 different ways

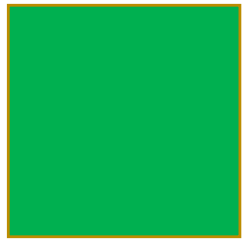
We can get a value based on index number – tuple[index]

We can get the length of a tuple – len(tuple)

We can’t add to a tuple – immutable – tuple[4] = value

We can count how many times a value appears in a tuple – tuple.count(value)

Tuple recap



We can now create tuples – 2 different ways

We can get a value based on index number – tuple[index]

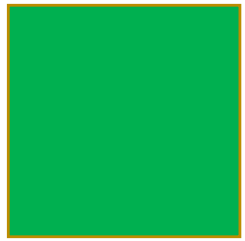
We can get the length of a tuple – len(tuple)

We can't add to a tuple – immutable – tuple[4] = value

We can count how many times a value appears in a tuple – tuple.count(value)

And we can get the index based on value (throws exception) if the value does not exist in the tuple – tuple.index(value)

Tuple recap



We can now create tuples – 2 different ways

We can get a value based on index number – tuple[index]

We can get the length of a tuple – len(tuple)

We can't add to a tuple – immutable – tuple[4] = value

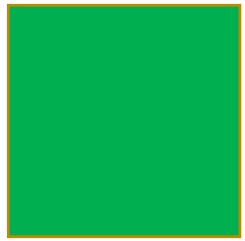
We can count how many times a value appears in a tuple – tuple.count(value)

And we can get the index based on value (throws exception) if the value does not exist in the tuple – tuple.index(value)

Oh and we can use - if value in tuple:

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Tuple recap



But…

Tuples are immutable

Once we create them we can't change them

Which of little benefit to us when we want a dynamic system

So we need a structure like Tuples, but which can change

So let's look at our list of Data types

Data Types

Some of the other Data types available in Python

Type	Example
Numeric: Integer, Float	x = 10 x = 10.0
String	x = “Mike”
Boolean	x = True x = False
List	x = [10, 20, 30]
Tuple	x = ("Ed", "Edd", "Eddy", 2009)
Dictionary	x = {‘one’: 1, ‘two’: 2}
List	x = ["Ed", "Edd", "Eddy", 2009]

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Tuple / List

- If we look at Tuple and List in our Data Types table

Type	Example
Tuple	x = ("Ed", "Edd", "Eddy", 2009)
List	x = ["Ed", "Edd", "Eddy", 2009]

Tuple / List

- If we look at Tuple and List in our Data Types table

Type	Example
Tuple	x = ("Ed", "Edd", "Eddy", 2009)
List	x = ["Ed", "Edd", "Eddy", 2009]

- We can see that the content of both types is identical

Tuple / List

- If we look at Tuple and List in our Data Types table

Type	Example
Tuple	x = ("Ed", "Edd", "Eddy", 2009)
List	x = ["Ed", "Edd", "Eddy", 2009]

- We can see that the content of both types is identical
- The only physical difference being the ( ) of Tuple

Tuple / List

- If we look at Tuple and List in our Data Types table

Type	Example
Tuple	x = ("Ed", "Edd", "Eddy", 2009)
List	x = ["Ed", "Edd", "Eddy", 2009]

- We can see that the content of both types is identical
- The only physical difference being the ( ) of Tuple
- And [ ] of List

Tuple / List

- If we look at Tuple and List in our Data Types table

Type	Example
Tuple	x = ("Ed", "Edd", "Eddy", 2009)
List	x = ["Ed", "Edd", "Eddy", 2009]

- We can see that the content of both types is identical
- The only physical difference being the ( ) of Tuple
- And [ ] of List
- But List is mutable, it's content can be changed…

List

- In List we can use all the functions we have seen for Tuple:

List

- In List we can use all the functions we have seen for Tuple:

  We can get a value based on index number – <list>[index]

List

- In List we can use all the functions we have seen for Tuple:

We can get a value based on index number – <list>[index]

We can get the length of a List – len(<list>)

List

- In List we can use all the functions we have seen for Tuple:

We can get a value based on index number – <list>[index]

We can get the length of a List – len(<list>)

We can add to a List – mutable – <list>[4] = value

List

- In List we can use all the functions we have seen for Tuple:

We can get a value based on index number – <list>[index]

We can get the length of a List – len(<list>)

We can add to a List – mutable – <list>[4] = value

We can count how many times a value appears in a List – <list>.count(value)

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

List

- In List we can use all the functions we have seen for Tuple:

We can get a value based on index number – <list>[index]

We can get the length of a List – len(<list>)

We can add to a List – mutable – <list>[4] = value

We can count how many times a value appears in a List – <list>.count(value)

And we can get the index based on value - <list>.index(value) (throws exception) if the value does not exist in the List

List

- In List we can use all the functions we have seen for Tuple:

We can get a value based on index number – <list>[index]

We can get the length of a List – len(<list>)

We can add to a List – mutable – <list>[4] = value

We can count how many times a value appears in a List – <list>.count(value)

And we can get the index based on value - <list>.index(value) (throws exception) if the value does not exist in the List

Oh and we can use - if value in <list>:

List

- In List we can use all the functions we have seen for Tuple:

We can get a value based on index number – <list>[index]

We can get the length of a List – len(<list>)

We can add to a List – mutable – <list>[4] = value

We can count how many times a value appears in a List – <list>.count(value)

And we can get the index based on value - <list>.index(value) (throws exception) if the value does not exist in the List

Oh and we can use - if value in <list>:

Let's look at some examples:

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

Create a List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print(great_show)
print(type(great_show))

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# <class 'list'>
```

Lists – examples…

Create a List

Print the List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print(great_show)
print(type(great_show))

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# <class 'list'>
```

Lists – examples…

Create a List

Print the List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print(great_show)
print(type(great_show))

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# <class 'list'>
```

Square Brackets

Lists – examples…

Create a List

Print the List

Type the List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print(great_show)
print(type(great_show))

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# <class 'list'>
```

Lists – examples…

Create a List

Print the List

Type the List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print(great_show)
print(type(great_show))

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# <class 'list'>
```

Lists – examples…

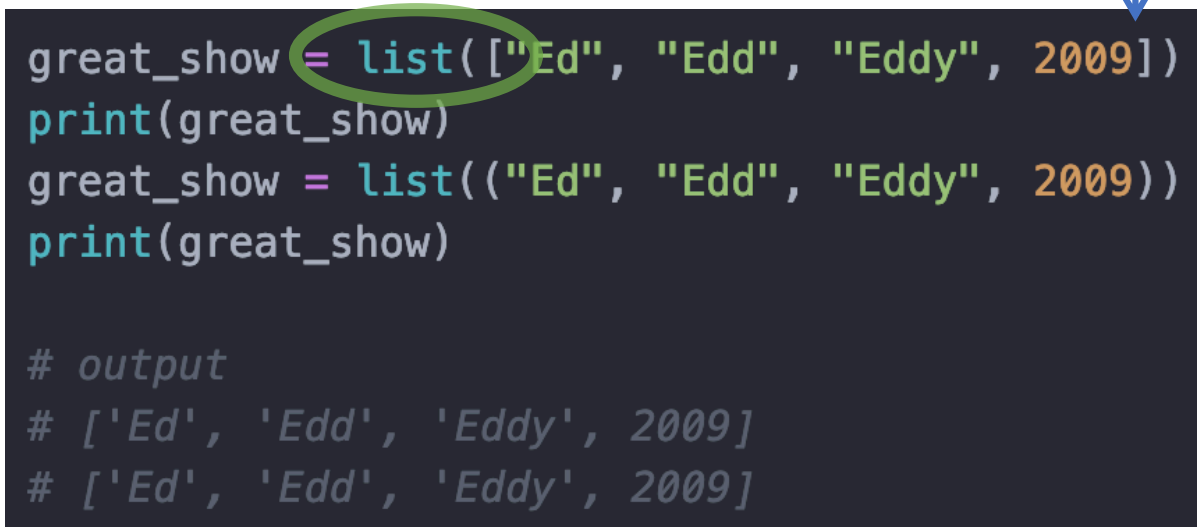
Create a List using list()

```
great_show = list(["Ed", "Edd", "Eddy", 2009])
print(great_show)
great_show = list(("Ed", "Edd", "Eddy", 2009))
print(great_show)

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# ['Ed', 'Edd', 'Eddy', 2009]
```

Lists – examples…

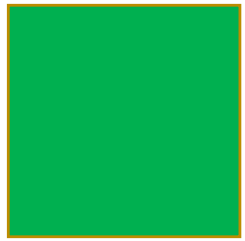
Create a List using list()

Square Brackets



```
great_show = list(["Ed", "Edd", "Eddy", 2009])
print(great_show)
great_show = list(("Ed", "Edd", "Eddy", 2009))
print(great_show)

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# ['Ed', 'Edd', 'Eddy', 2009]
```

Lists – examples…



Create a List using list()

Create a List from a Tuple using list()

```
great_show = list(["Ed", "Edd", "Eddy", 2009])
print(great_show)
great_show = list(("Ed", "Edd", "Eddy", 2009))
print(great_show)

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# ['Ed', 'Edd', 'Eddy', 2009]
```

Lists – examples…

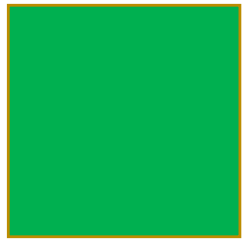
Create a List using list()

Create a List from a Tuple using list()

```
great_show = list(["Ed", "Edd", "Eddy", 2009])
print(great_show)
great_show = list(("Ed", "Edd", "Eddy", 2009))
print(great_show)

# output
# ['Ed', 'Edd', 'Eddy', 2009]
# ['Ed', 'Edd', 'Eddy', 2009]
```

Round Brackets

Lists – examples…



We can even create a List from a String using list()

```
great_show = list("Ed and Edd")
print(great_show)

# output
# ['E', 'd', ' ', 'a', 'n', 'd', ' ', 'E', 'd', 'd']
```

Lists – examples…

Get the index value from a List using <list_name>[<index>]

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print(great_show[1])

# output
# Edd
```

Lists – examples…

Get the index value from a List using <list_name>[<index>]

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print(great_show[1])

# output
# Edd
```


Square Brackets

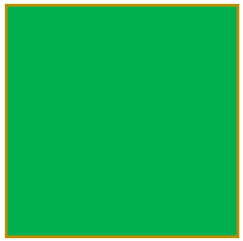
Lists – examples…

Get the number of elements in the List using len(<list_name>)

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print("There are "+str(len(great_show))+" items in this list")

# output
# There are 4 items in this list
```

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



Get the number of elements in the List using len(<list_name>)

Each value in the List is known as an element

```
great_show = ["Ed", "Edd", "Eddy", 2009]
print("There are "+str(len(great_show))+" items in this list")

# output
# There are 4 items in this list
```

Lists – examples…

Add an element to the List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
great_show[4] = "plank"

# output
# IndexError: list assignment index out of range
```

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

Add an element to the List

Oops….

```
great_show = ["Ed", "Edd", "Eddy", 2009]
great_show[4] = "plank"

# output
# IndexError: list assignment index out of range
```

Lists – examples…

Add an element to the List

If the index does not exist, we get an error

```
great_show = ["Ed", "Edd", "Eddy", 2009]
great_show[4] = "plank"

# output
# IndexError: list assignment index out of range
```

Lists – examples…

Add an element to the List

If the index does not exist, we get an error

```
great_show = ["Ed", "Edd", "Eddy", 2009]
great_show[4] = "plank"

# output
# IndexError: list assignment index out of range
```

Index 4 does not exist

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

So, we use append() to add to the end of a List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
great_show.append("plank")
print(great_show[4])
print(great_show)

# output
# plank
# ['Ed', 'Edd', 'Eddy', 2009, 'plank']
```

Lists – examples…

So, we use append() to add to the end of a List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
great_show.append("plank")
print(great_show[4])
print(great_show)

# output
# plank
# ['Ed', 'Edd', 'Eddy', 2009, 'plank']
```

Lists – examples…

Here we append() to the end of the List

```
great_show = ["Ed", "Edd", "Eddy", 2009]
great_show.append("plank")
print(great_show)
great_show[4] = "Jonny"
print(great_show[4])
print(great_show)

# output
# ['Ed', 'Edd', 'Eddy', 2009, 'plank']
# Jonny
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

Lists – examples…

Here we append() to the end of the List

And then we can use index allocation to change the value

```
great_show = ["Ed", "Edd", "Eddy", 2009]
great_show.append("plank")
print(great_show)
great_show[4] = "Jonny"
print(great_show[4])
print(great_show)

# output
# ['Ed', 'Edd', 'Eddy', 2009, 'plank']
# Jonny
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

Lists – examples…

Here I use append() or index allocation depending on List length

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 4
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use append
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

Lists – examples…

Here I use append() or index allocation depending on List length

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 4
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use append
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

Lists – examples…

Here I use append or index allocation depending on List length

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 4
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use append
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

As CS indexing starts at zero

Lists – examples…

As CS indexing starts at zero

len() will always return one more than the highest index number

Lists – examples…

As CS indexing starts at zero

len() will always return one more than the highest index number

```
great_show = ["Ed", "Edd", "Eddy", 2009]
```

Lists – examples…

As CS indexing starts at zero

len() will always return one more than the highest index number

```
great_show = ["Ed", "Edd", "Eddy", 2009]

len(great_show) -> 4
```

Lists – examples…

As CS indexing starts at zero

len() will always return one more than the highest index number

great_show = ["Ed", "Edd", "Eddy", 2009]

len(great_show) -> 4

Index number of value 2009 is 3

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

As CS indexing starts at zero

len() will always return one more than the highest index number

```
great_show = ["Ed", "Edd", "Eddy", 2009]
              0      1       2       3
len(great_show) -> 4
```

Index number of value 2009 is 3

Lists – examples…

As CS indexing starts at zero

len() will always return one more than the highest index number

```
great_show = ["Ed", "Edd", "Eddy", 2009]
              0      1       2       3
len(great_show) -> 4
```

Index number of value 2009 is 3

So max index is one less than len()

Lists – examples…

As CS indexing starts at zero

len() will always return one more than the highest index number

```
great_show = ["Ed", "Edd", "Eddy", 2009]
              0      1       2       3
len(great_show) -> 4
```

Index number of value 2009 is 3

So max index is one less than len()

```
len(great_show )-1
```

Remember this…

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

Here I use append or index allocation depending on List length

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 4
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use append
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

Lists – examples…

Here I use append or index allocation depending on List length

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 4
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use append
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

Lists – examples…

Here I use append or index allocation depending on List length

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 4
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use append
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

Lists – examples…

Here I use append or index allocation depending on List length

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 4
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use append
# ['Ed', 'Edd', 'Eddy', 2009, 'Jonny']
```

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

Here I use append or index allocation depending on List length

Change index to add to 2

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 2
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use index allocation
# ['Ed', 'Edd', 'Jonny', 2009]
```

Lists – examples…

Here I use append or index allocation depending on List length

Change index to add to 2

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 2
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use index allocation
# ['Ed', 'Edd', 'Jonny', 2009]
```

Lists – examples…

Here I use append or index allocation depending on List length

Change index to add to 2

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 2
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use index allocation
# ['Ed', 'Edd', 'Jonny', 2009]
```

Lists – examples…

Here I use append or index allocation depending on List length

Change index to add to 2

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 2
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show[index_to_add] = "Jonny"
    print("I use index allocation")

print(great_show)

# output
# I use index allocation
# ['Ed', 'Edd', 'Jonny', 2009]
```

I can also use great_show.insert(index, value)

Lists – examples…

Here I use append or index allocation depending on List length

Change index to add to 2

```
great_show = ["Ed", "Edd", "Eddy", 2009]
index_to_add = 2
if len(great_show) <= index_to_add:
    great_show.append("Jonny")
    print("I use append")
else:
    great_show.insert(index_to_add, "Jonny")
    print("I use index allocation")

print(great_show)

# output
# I use index allocation
# ['Ed', 'Edd', 'Jonny', 2009]
```

I can also use great_show.insert(index, value)

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

<list_name>.count(<value>) returns
how many times <value> occurs in the List

```
great_show = ["Ed", "Ed", "Ed", 2009]
print(str(great_show[0]) + " occurs "
      + str(great_show.count(great_show[0]))
      + " time(s) in this list")

# output
# Ed occurs 3 time(s) in this list
```

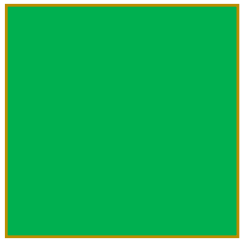
Lists – examples…

<list_name>.index(<value>) returns
the index number <value> first occurs at, in the List

```
great_show = ["Ed", "Ed", "Ed", 2009]
print("Ed occurs at index "
      + str(great_show.index("Ed"))
      + " in this list")

# output
# Ed occurs at index 0 in this list
```

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

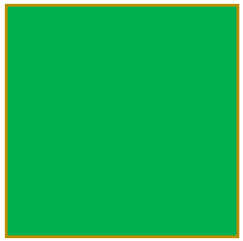
Lists – examples…



Looking for a <value> using <list_name>.index(<value>) that does not occurs in the List will cause an error

```
great_show = ["Ed", "Ed", "Ed", 2009]
# print("ed occurs at index " + str(great_show.index("ed")) + " in this list")

# output
# ValueError: 'ed' is not in list
```

Lists – examples…



We can use try/except to catch these errors

```
try:
    great_show = ["Ed", "Ed", "Ed", 2009]
    print("ed occurs at index "
          + str(great_show.index("ed"))
          + " in this list")
except Exception as e:
    print("ed does not occurs in "
          + str(great_show))

# output
# ed does not occurs in ['Ed', 'Ed', 'Ed', 2009]
```

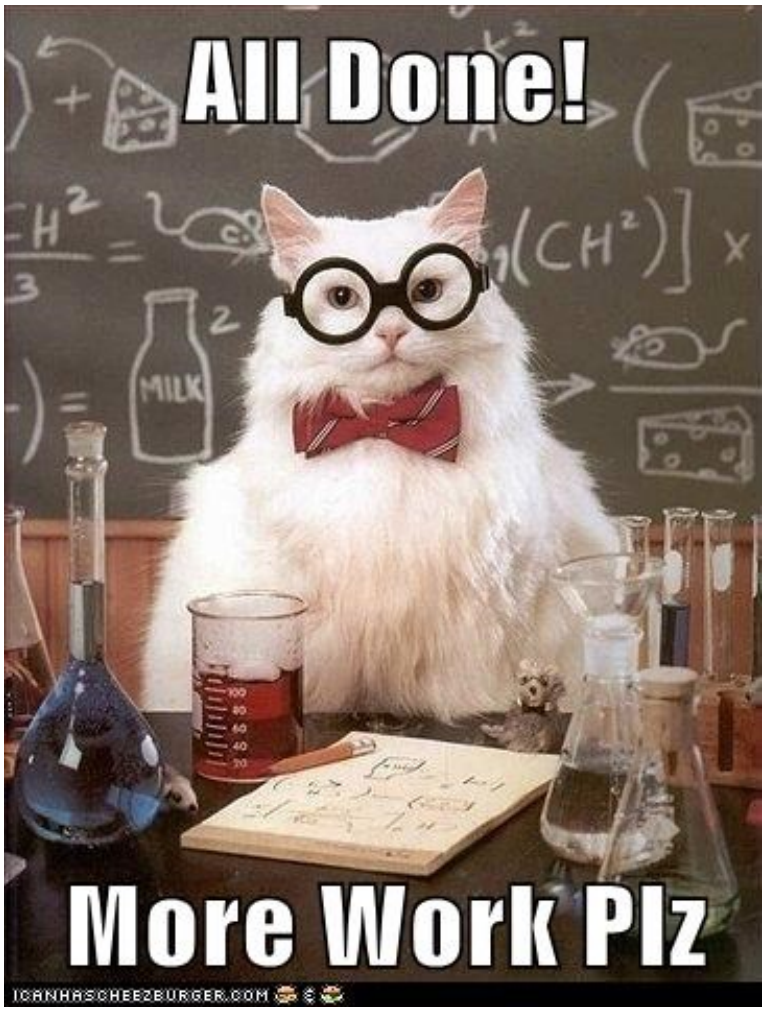
Lists – examples…

Finally, we can use in and not in

```
great_show = ["Ed", "Ed", "Ed", 2009]
value_to_find = "ed"

if value_to_find in great_show:
    print(value_to_find + " occurs in "
          + str(great_show))
else:
    print(value_to_find + " does not occurs in "
          + str(great_show))

if value_to_find not in great_show:
    print(value_to_find + " does not occurs in "
          + str(great_show))

# output
# ed does not occurs in ['Ed', 'Ed', 'Ed', 2009]
# ed does not occurs in ['Ed', 'Ed', 'Ed', 2009]
```

Canvas Student App

Let’s Sign into this lecture now



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

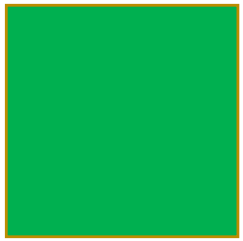
Lists – examples…

So, we've seen all the functionality common between tuple and list

Let's look at new functionality for list

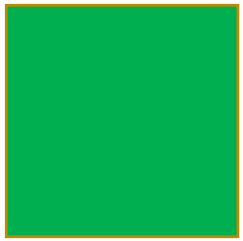
[1,2,3] * 3 -> [1,2,3,1,2,3,1,2,3]

Lists – examples…



We can use negative indexes to get values from the end of a list

Lists – examples…



We can use negative indexes to get values from the end of a list

my_list = [“a”, “b”, “c”]

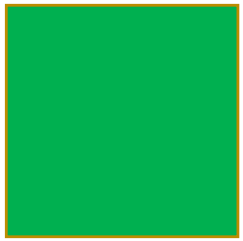
Lists – examples…

We can use negative indexes to get values from the end of a list

my_list = [“a”, “b”, “c”]

my_list[2] -> “c”

Lists – examples…



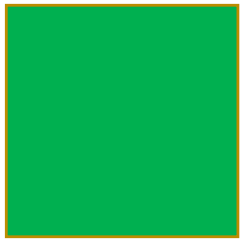
We can use negative indexes to get values from the end of a list

my_list = [“a”, “b”, “c”]

0 1 2

my_list[2] -> “c”

Lists – examples…



We can use negative indexes to get values from the end of a list

my_list = [“a”, “b”, “c”]

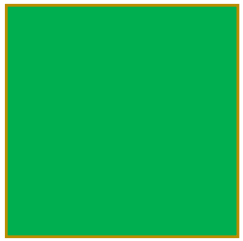
0 1 2

my_list[2] -> “c”

my_list[-1] -> “c”

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



We can use negative indexes to get values from the end of a list

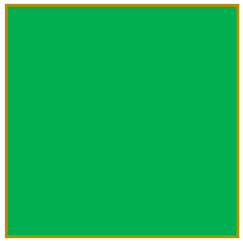
-3	-2	-1
my_list = [“a”, “b”, “c”]		
0	1	2

my_list[2] -> “c”

my_list[-1] -> “c”

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



We can use + operator to concatenate lists together

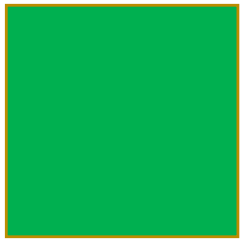
my_list_1 = [“a”, “b”, “c”]

my_list_2 = [“d”, “e”, “f”]

my_list_1 = my_list_1 + my_list_2

my_list_1 -> [“a”, “b”, “c”, “d”, “e”, “f”]

Lists – examples…



We can use += operator to concatenate lists together

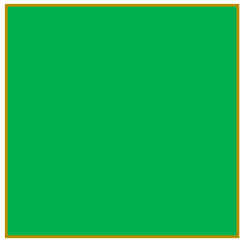
my_list_1 = [“a”, “b”, “c”]

my_list_2 = [“d”, “e”, “f”]

my_list_1 += my_list_2

my_list_1 -> [“a”, “b”, “c”, “d”, “e”, “f”]

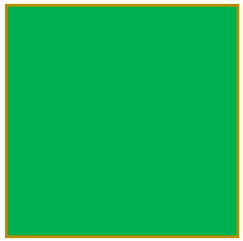
Lists – examples…



We can use sort() to sort the contents of a list from lowest to highest value

```
my_list_1 = [“a”, “B”, “7”]
my_list_1.sort()
my_list_1 -> [“7”, “B”, “a”]

my_list_2 = [“9”, “i”, “g”]
my_list_2.sort()
my_list_1 -> [“9”, “g”, “i”]
```

Lists – examples…



We can use remove(value) to remove the first occurrence of a value from a list

my_list_1 = [“a”, “B”, “7”]

my_list_1.remove(“7”)

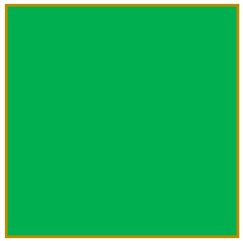
my_list_1 -> [“a”, “B”]

my_list_2 = [“9”, “i”, “g”]

my_list_2. remove (“7)

Error - ValueError: list.remove(x): x not in list

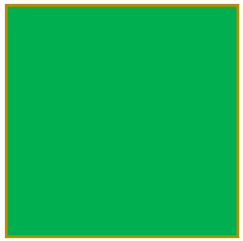
Lists – examples…



We can use del list [index] to remove the index from a list

```
my_list_1 = [“a”, “B”, “7”]
del my_list_1[1]
my_list_1 -> [“a”, “7”]

my_list_2 = [“9”, “i”, “g”]
del my_list_2[0]
my_list_2 -> [“i”, “g”]
```

Lists – examples…



We can use reverse() to reverse the list

my_list_1 = [“a”, “B”, “7”]

my_list_1.reverse()

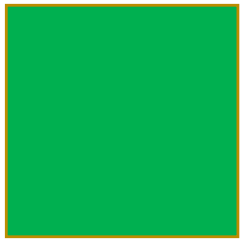
my_list_1 -> [“7”, “B”, “a”]

my_list_2 = [“9”, “i”, “g”]

my_list_2. reverse()

my_list_1 -> [“g”, “i”, “9”]

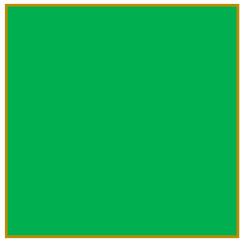
Lists – examples…



We can use min()/max() operator to get min and max values in a list

```
my_list_1 = [“a”, “B”, “7”]
min_val = min(my_list_1)
min_val -> “7”

max_val = max(my_list_1)
max_val -> “a”
```

Lists – examples…



Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, "d”, "4"]

0 1 2 3 4

new_list = my_list_1[1 : 4]

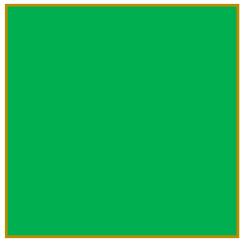
new_list -> [“B”, “7”, "d"]

list[*start* : *end*]

Returned list does NOT include the end index!!!!

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, "d”, "4"]

0 1 2 3 4

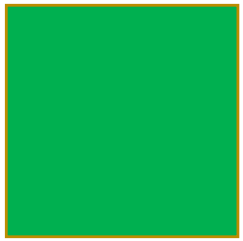
new_list = my_list_1[: 4]

new_list -> [“a”, “B”, “7”, "d"]

If we don’t specify the start index
0 is used by default

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, “d”, “4"]

0 1 2 3 4

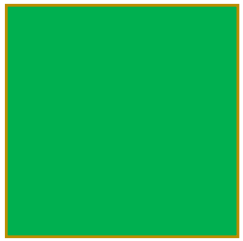
new_list = my_list_1[1:]

new_list -> [“B”, “7”, "d", "4"]

If we don’t specify the end index
len(list) is used by default

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, “d”, “4"]

0 1 2 3 4

new_list = my_list_1[1:]

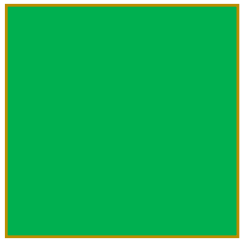
new_list -> [“B”, “7”, "d", "4"]

If we don’t specify the end index
len(list) is used by default

Why don’t we use len(list)-1??

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, "d”, "4"]

0 1 2 3 4

new_list = my_list_1[:]

new_list -> [“a”, “B”, “7”, "d", "4"]

If we don’t specify the start or end index
The entire list is returned by default

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…

Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, "d”, "4"]

-5 -4 -3 -2 -1

new_list = my_list_1[: -1]

new_list -> [“a”, “B”, “7”, "d"]

We can also use the negative indexes

new_list = my_list_1[: -4]

new_list -> [“a”]

Lists – examples…

Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, "d”, "4"]

-5 -4 -3 -2 -1

new_list = my_list_1[0 : -1]

new_list -> [“a”, “B”, “7”, "d"]

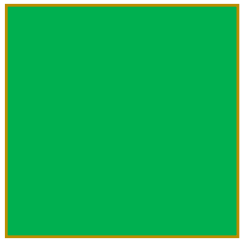
We can actually use any combinations of positive and negative values

new_list = my_list_1[2: -2]

new_list -> [“7”]

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, "d”, "4"]

-5 -4 -3 -2 -1

new_list = my_list_1[3 : 1]

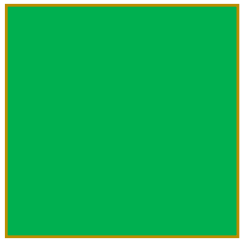
new_list -> []

If start is larger than end

You get an empty list

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lists – examples…



Slicing – extracting a subset of the list values

my_list_1 = [“a”, “B”, “7”, "d”, "4"]

-5 -4 -3 -2 -1

new_list = my_list_1[3 : 1]

new_list -> []

But we can add a third step value to step over the list

new_list = my_list_1[3 : 1 : -1]

new_list -> [“d”, ”7”]

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append()

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort()

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value)

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value),

del list[index]

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value),

del list[index], list.reverse()

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value),

del list[index], list.reverse(), min/max(list)

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value), del list[index], list.reverse(), min/max(list), insert(index,value)

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value), del list[index], list.reverse(), min/max(list), insert(index,value)

Operators - *

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value), del list[index], list.reverse(), min/max(list), insert(index,value)

Operators - *, +

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value), del list[index], list.reverse(), min/max(list), insert(index,value)

Operators - *, +, +=

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value), del list[index], list.reverse(), min/max(list), insert(index,value)

Operators - *, +, +=

Slicing – list[start:end]

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

Functions - list.append(), list.sort(), list.remove(value), del list[index], list.reverse(), min/max(list), insert(index,value)

Operators - *, +, +=

Slicing – list[start:end], list[start:end:step]

List Recap

In List we can use all the functions we have seen for Tuple, plus we can use:

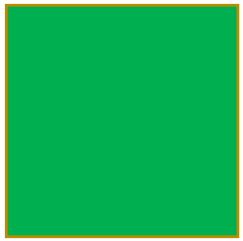
Functions - list.append(), list.sort(), list.remove(value), del list[index], list.reverse(), min/max(list), insert(index,value)

Operators - *, +, +=

Slicing – list[start:end], list[start:end:step]

Negative indexing – list[-2]

Slicing

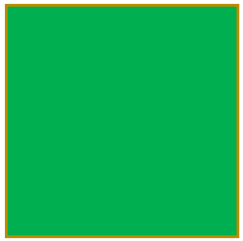


Does Slicing only work for Lists?

```
my_list_1 = ["a", "B", "7", "d", "4"]
new_list = my_list_1[2:]
print(new_list)

# output
# ['7', 'd', '4']
```

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Slicing

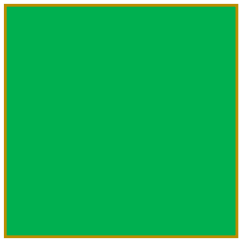


Does Slicing only work for Lists?

```
my_tuple_1 = ("a", "B", "7", "d", "4")
new_tuple = my_tuple_1[2:]
print(new_tuple)

# output
# ('7', 'd', '4')
```

Works for Tuples…

Slicing

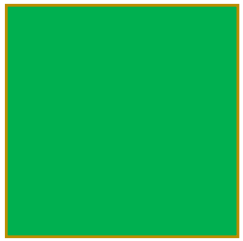


Does Slicing only work for Lists?

```
string_1 = "hello world"
new_string = string_1[2:]
print(new_string)

# output
# llo world
```

Works for Strings…

Slicing



Does Slicing only work for Lists?

```
int_1 = 123456
new_int = int_1[2:]
print(new_int)

# output
# TypeError: 'int' object is not subscriptable
```

Does not work for Integers….

What about Floats?

Slicing

Does Slicing only work for Lists?

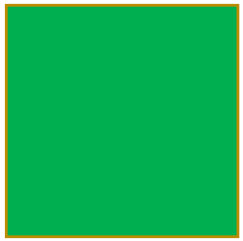
```
float_1 = 123.456
new_float = float_1[2:]
print(new_float)

# output
# TypeError: 'float' object is not subscriptable
```

Does not work for Integers or Floats…

But….

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Slicing

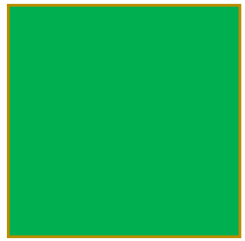


Does Slicing only work for Lists?

```
int_1 = str(123456)
new_int = int_1[2:]
print(new_int)

# output
# 3456
```

Cast int to String, and then slicing works ☺

Slicing

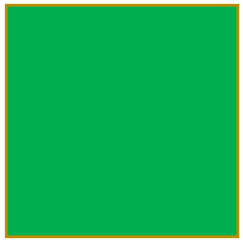


Does Slicing only work for Lists?

```
float_1 = str(123.456)
new_float = float_1[2:]
print(new_float)

# output
# 3.456
```

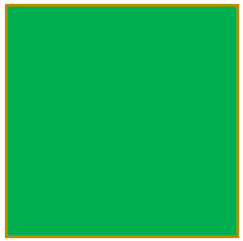
Cast float to String, and then slicing works ☺

Slicing



So Slicing work for a host of different Data Types?

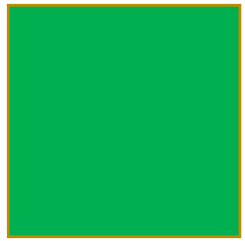
Slicing



So Slicing work for a host of different Data Types?

When the Data Type does not support indexing

Slicing

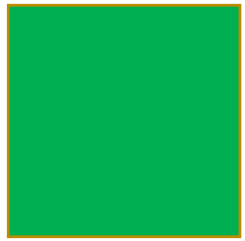


So Slicing work for a host of different Data Types?

When the Data Type does not support indexing

We can cast to a Data Type that does support indexing

Slicing



So Slicing work for a host of different Data Types?

When the Data Type does not support indexing

We can cast to a Data Type that does support indexing

Do our Slicing

Slicing

So Slicing work for a host of different Data Types?

When the Data Type does not support indexing

We can cast to a Data Type that does support indexing

Do our Slicing

And then cast back to the original Data Type

☺

Live Coding Time…

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh