Styles

So far you may be aware of a few styles of programming:

Styles

So far you may be aware of a few styles of programming:

- Object Oriented style programming: Java and others, where you structure your code about objects (and classes)

Styles

So far you may be aware of a few styles of programming:

- Object Oriented style programming: Java and others, where you structure your code about objects (and classes)
- Procedural style: C, Python and others, where you structure about procedures (functions)

Styles

So far you may be aware of a few styles of programming:

- Object Oriented style programming: Java and others, where you structure your code about objects (and classes)
- Procedural style: C, Python and others, where you structure about procedures (functions)
- Functional Style: Haskell and others, where you structure about higher order functions in a tightly controlled manner

Styles

So far you may be aware of a few styles of programming:

- Object Oriented style programming: Java and others, where you structure your code about objects (and classes)
- Procedural style: C, Python and others, where you structure about procedures (functions)
- Functional Style: Haskell and others, where you structure about higher order functions in a tightly controlled manner
- Declarative style: Prolog and ASP (and Haskell), where you structure about the things you want to be true or want to happen

Styles

So far you may be aware of a few styles of programming:

- Object Oriented style programming: Java and others, where you structure your code about objects (and classes)
- Procedural style: C, Python and others, where you structure about procedures (functions)
- Functional Style: Haskell and others, where you structure about higher order functions in a tightly controlled manner
- Declarative style: Prolog and ASP (and Haskell), where you structure about the things you want to be true or want to happen
- No style: unstructured things like assembly language, where there is no support for structuring

Styles

You will have spent a lot of time on procedural and object oriented

Styles

You will have spent a lot of time on procedural and object oriented

The declarative and functional should be covered in other Units at some point

Styles

You will have spent a lot of time on procedural and object oriented

The declarative and functional should be covered in other Units at some point

But the point is there are many *styles* of programming

Styles

A “style” is an approach to programming or an aspect of the design of a programming language that is meant to make it easier for you to write correct programs

Styles

A “style” is an approach to programming or an aspect of the design of a programming language that is meant to make it easier for you to write correct programs

It is easy to write small programs in a slapdash manner: you can get away with it as you can hold the whole program in your head

Styles

A “style” is an approach to programming or an aspect of the design of a programming language that is meant to make it easier for you to write correct programs

It is easy to write small programs in a slapdash manner: you can get away with it as you can hold the whole program in your head

When projects get large you cannot do this

Styles

The code gets too large for you to remember all the details

Styles

The code gets too large for you to remember all the details

Or there are multiple people working on the code

Styles

The code gets too large for you to remember all the details

Or there are multiple people working on the code

So styles are invented to direct the way you write code so to make large systems written by many programmers possible

Styles

They encapsulate detail into blobs to help you keep a grasp on what is happening in your program

Styles

They encapsulate detail into blobs to help you keep a grasp on what is happening in your program

You then think “at a higher level” using blobs

Styles

They encapsulate detail into blobs to help you keep a grasp on what is happening in your program

You then think “at a higher level” using blobs

Those blobs might be objects, modules, functions or other things

Styles

They encapsulate detail into blobs to help you keep a grasp on what is happening in your program

You then think “at a higher level” using blobs

Those blobs might be objects, modules, functions or other things

Roughly speaking, the nature of the blobs is what distinguishes between the various styles

Styles

They encapsulate detail into blobs to help you keep a grasp on what is happening in your program

You then think “at a higher level” using blobs

Those blobs might be objects, modules, functions or other things

Roughly speaking, the nature of the blobs is what distinguishes between the various styles

But they all strive to make programming simpler and to control complexity

Styles

They encapsulate detail into blobs to help you keep a grasp on what is happening in your program

You then think “at a higher level” using blobs

Those blobs might be objects, modules, functions or other things

Roughly speaking, the nature of the blobs is what distinguishes between the various styles

But they all strive to make programming simpler and to control complexity

And make writing correct programs easier

Styles

If we had perfect programmers, then none of this structuring would be strictly necessary

Styles

If we had perfect programmers, then none of this structuring would be strictly necessary

> *Structure is a way of helping 3rd rate programmers to produce 2nd rate quality code*

James Davenport (probably)

Styles

If we had perfect programmers, then none of this structuring would be strictly necessary

> *Structure is a way of helping 3rd rate programmers to produce 2nd rate quality code*

James Davenport (probably)

Programmers make mistakes. So how can a programming language help the programmer to make fewer mistakes?

Styles

> *Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it*

Brian Kernighan

> *The errors you can spot easily are not the ones you need to worry about*

Anon

> *Programmers should be lazy and stupid*

Me

Styles

Programming languages themselves do have a role to play in making things easier for the programmer

Styles

Programming languages themselves do have a role to play in making things easier for the programmer

Or harder, if they are bad things or habits

Styles

Programming languages themselves do have a role to play in making things easier for the programmer

Or harder, if they are bad things or habits

Unfortunately, there is no single programming language that does everything well

Styles

There are still people out there that think their favourite programming language is perfect and try to support it by saying that “it’s the programmer’s fault” or “they weren’t using the correct features” when something goes wrong

Styles

There are still people out there that think their favourite programming language is perfect and try to support it by saying that “it’s the programmer’s fault” or “they weren’t using the correct features” when something goes wrong

For examples, see C and memory management; and C++ and almost any feature

Styles

There are still people out there that think their favourite programming language is perfect and try to support it by saying that “it’s the programmer’s fault” or “they weren’t using the correct features” when something goes wrong

For examples, see C and memory management; and C++ and almost any feature

In 2010, the iPhone 4 had problems with reception. Apple’s response was essentially “you are holding it incorrectly”

Styles

If you have a hammer, everything looks like a nail
If you have Java, everything looks object oriented

Anonymous

Styles

One thing to remember before we start: programming styles are not exclusive

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java
—though Java makes it hard to do so easily

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java
—though Java makes it hard to do so easily

You can write in an OO style in C

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java
—though Java makes it hard to do so easily

You can write in an OO style in C
—though C doesn't really provide the constructs for you to do so easily

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java
—though Java makes it hard to do so easily

You can write in an OO style in C
—though C doesn't really provide the constructs for you to do so easily

Though it may still be worthwhile to structure your C code around OO ideas

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java
—though Java makes it hard to do so easily

You can write in an OO style in C
—though C doesn't really provide the constructs for you to do so easily

Though it may still be worthwhile to structure your C code around OO ideas

Both Java and Python (and many others) have OO *and* procedural aspects

Styles

Some languages *support* certain styles

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO (to the extent of making it hard *not* to use objects)

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO (to the extent of making it hard *not* to use objects)

Other languages allow you to use a style, but you have to do the twiddly bits yourself

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO (to the extent of making it hard *not* to use objects)

Other languages allow you to use a style, but you have to do the twiddly bits yourself

You can program OO in C, but you have to do the “method” lookup yourself, i.e., pick the code for the right “method” yourself in your program

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO (to the extent of making it hard *not* to use objects)

Other languages allow you to use a style, but you have to do the twiddly bits yourself

You can program OO in C, but you have to do the “method” lookup yourself, i.e., pick the code for the right “method” yourself in your program

Sometimes this is good as it allows you to optimise for the problem in hand

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO (to the extent of making it hard *not* to use objects)

Other languages allow you to use a style, but you have to do the twiddly bits yourself

You can program OO in C, but you have to do the “method” lookup yourself, i.e., pick the code for the right “method” yourself in your program

Sometimes this is good as it allows you to optimise for the problem in hand

Java provides a general mechanism (that you don’t see) for method lookup that has to work for all kinds of situations

Styles

From the other direction, some *problems* lend themselves better to a certain style

Styles

From the other direction, some *problems* lend themselves better to a certain style

For example, a GUI application with windows and click boxes would be naturally OO

Styles

From the other direction, some *problems* lend themselves better to a certain style

For example, a GUI application with windows and click boxes would be naturally OO

For a heavily numerical application, such as a weather forecasting, objects would be just a hindrance to coding

Styles

So some problems naturally suggest a choice of language to use to implement them

Styles

So some problems naturally suggest a choice of language to use to implement them

Others problems are near impossible to program regardless of style

Styles

So some problems naturally suggest a choice of language to use to implement them

Others problems are near impossible to program regardless of style

Part of being a Computer Scientist is knowing these styles and knowing which languages support them

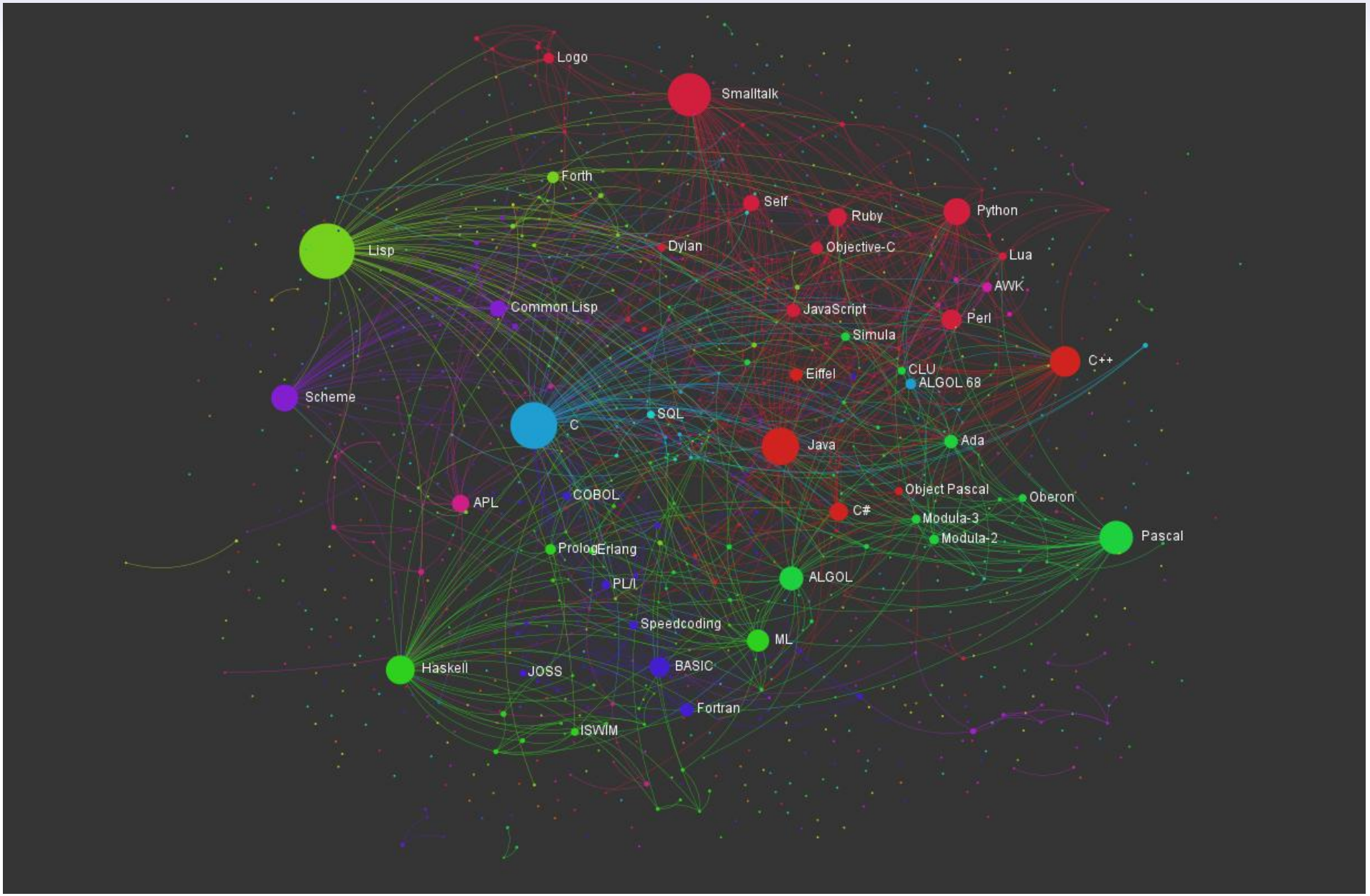
Styles

So some problems naturally suggest a choice of language to use to implement them

Others problems are near impossible to program regardless of style

Part of being a Computer Scientist is knowing these styles and knowing which languages support them

And then picking a style for a problem, then a language that supports that style



Programming Languages Influence Network

Language Families

From
`https://exploringdata.github.io/vis/programming-languages-influence-network/`

an interactive, zoomable map of languages

Language Families

Out there in the real world there are very many programming languages

Language Families

Out there in the real world there are very many programming languages

How many are actually *useful* is a difficult question

Language Families

Out there in the real world there are very many programming languages

How many are actually *useful* is a difficult question

Creating new languages is easy these days (go to a compilers unit!)

Language Families

Out there in the real world there are very many programming languages

How many are actually *useful* is a difficult question

Creating new languages is easy these days (go to a compilers unit!)

But designing a *good* and *useful* new language is much harder

Language Families

Out there in the real world there are very many programming languages

How many are actually *useful* is a difficult question

Creating new languages is easy these days (go to a compilers unit!)

But designing a *good* and *useful* new language is much harder

And almost always unnecessary

Language Families

Most “new” languages are modest variants on existing languages, often emphasising some features of interest to the language designer

Language Families

Most “new” languages are modest variants on existing languages, often emphasising some features of interest to the language designer

Very few have a radical new idea or approach

Language Families

Most “new” languages are modest variants on existing languages, often emphasising some features of interest to the language designer

Very few have a radical new idea or approach

Thus languages tend to fall into *families*, where members of the same family have many features in common

Language Families

You may have already seen (or will soon see)

- C: procedural
- Haskell, Lisp: functional
- Prolog, ASP: logic
- Python: procedural and scripting
- Java: object oriented
- etc.

Families are not clearly separated: they are fuzzy at the edges

Language Families

Feet

- C: you shoot yourself in the foot

Language Families

Feet

- C: you shoot yourself in the foot
- Lisp: You shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds. . .

Language Families

Feet

- C: you shoot yourself in the foot
- Lisp: You shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds...
- Prolog: You tell your program you want to be shot in the foot. The program figures out how to do it, but the syntax doesn't allow it to explain

Language Families

Feet

- C: you shoot yourself in the foot
- Lisp: You shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds...
- Prolog: You tell your program you want to be shot in the foot. The program figures out how to do it, but the syntax doesn't allow it to explain
- Python: You try to shoot yourself in the foot but you just keep hitting the whitespace between your toes

Language Families

Feet

- Java: You locate the Gun class, but discover that the Bullet class is abstract, so you extend it and write the missing part of the implementation. Then you implement the ShootAble interface for your foot, and recompile the Foot class. The interface lets the bullet call the doDamage method on the Foot, so the Foot can damage itself in the most effective way. Now you run the program, and call the doShoot method on the instance of the Gun class. First the Gun creates an instance of Bullet, which calls the doFire method on the Gun. The Gun calls the hit(Bullet) method on the Foot, and the instance of Bullet is passed to the Foot. But this causes an IllegalHitByBullet exception to be thrown, and you die

Language Families

There are many many many programming languages

Language Families

There are many many many programming languages

Some are used widely, but you don’t realise (Cobol, Fortran)

Language Families

There are many many many programming languages

Some are used widely, but you don't realise (Cobol, Fortran)

Some are not used much at all, but have been important influences on other languages (APL, Snobol, Algol)

Language Families

Feet

- Cobol: USEing a COLT 45 HANDGUN, AIM gun at LEG.FOOT, THEN place ARM.HAND.FINGER on HANDGUN.TRIGGER and SQUEEZE. THEN return HANDGUN to HOLSTER. CHECK whether shoelace needs to be retied

Language Families

Feet

- Cobol: USEing a COLT 45 HANDGUN, AIM gun at LEG.FOOT, THEN place ARM.HAND.FINGER on HANDGUN.TRIGGER and SQUEEZE. THEN return HANDGUN to HOLSTER. CHECK whether shoelace needs to be retied
- Fortran: You shoot yourself in each toe, iteratively, until you run out of toes, then you read in the next foot and repeat. If you run out of bullets, you continue anyway because you have no exception-handling facility

Language Families

Feet

- Cobol: USEing a COLT 45 HANDGUN, AIM gun at LEG.FOOT, THEN place ARM.HAND.FINGER on HANDGUN.TRIGGER and SQUEEZE. THEN return HANDGUN to HOLSTER. CHECK whether shoelace needs to be retied
- Fortran: You shoot yourself in each toe, iteratively, until you run out of toes, then you read in the next foot and repeat. If you run out of bullets, you continue anyway because you have no exception-handling facility
- APL: You hear a gunshot and there's a hole in your foot, but you don't remember enough linear algebra to understand what happened.

Language Families

Feet

- Cobol: USEing a COLT 45 HANDGUN, AIM gun at LEG.FOOT, THEN place ARM.HAND.FINGER on HANDGUN.TRIGGER and SQUEEZE. THEN return HANDGUN to HOLSTER. CHECK whether shoelace needs to be retied
- Fortran: You shoot yourself in each toe, iteratively, until you run out of toes, then you read in the next foot and repeat. If you run out of bullets, you continue anyway because you have no exception-handling facility
- APL: You hear a gunshot and there's a hole in your foot, but you don't remember enough linear algebra to understand what happened.
- Snobol: If you succeed, shoot yourself in the left foot. If you fail, shoot yourself in the right foot

Language Families

Feet

Continuing Exercise go and read further around these (and other) languages to discover why they have these descriptions

Language Families

Feet

Continuing Exercise go and read further around these (and other) languages to discover why they have these descriptions

Exercise for advanced students: make up jokes for the missing ones and funnier versions for existing ones

Language Families

There are hundreds of languages out there

Language Families

There are hundreds of languages out there

How do we choose which to use?

Language Families

There are hundreds of languages out there

How do we choose which to use?

Sometimes we are told by the boss, or have to fit with an existing project

Language Families

There are hundreds of languages out there

How do we choose which to use?

Sometimes we are told by the boss, or have to fit with an existing project

Sometimes we only have a restricted choice

Language Families

If we have any choice we need to have some criteria to guide the choice

Language Families

If we have any choice we need to have some criteria to guide the choice

We need to able to

Language Families

If we have any choice we need to have some criteria to guide the choice

We need to able to

- identify and assess characteristics of a given language

Language Families

If we have any choice we need to have some criteria to guide the choice

We need to able to

- identify and assess characteristics of a given language
- recognise similarities between languages

Language Families

If we have any choice we need to have some criteria to guide the choice

We need to able to

- identify and assess characteristics of a given language
- recognise similarities between languages
- recognise if a feature is unique to a language

Language Families

If we have any choice we need to have some criteria to guide the choice

We need to able to

- identify and assess characteristics of a given language
- recognise similarities between languages
- recognise if a feature is unique to a language
- take concepts from one language to another (learn one, learn 'em all)

Language Families

“If you can’t say it you can’t think it” (Orwell/Wittgenstein)

Language Families

“If you can’t say it you can’t think it” (Orwell/Wittgenstein)

Having more concepts allows more flexibility: for example, if there is no array construct in the language, you are restricted in what you can do easily

Language Families

“If you can’t say it you can’t think it” (Orwell/Wittgenstein)

Having more concepts allows more flexibility: for example, if there is no array construct in the language, you are restricted in what you can do easily

“Those who cannot remember the past are condemned to repeat it” (George Santayana)

Language Families

“If you can’t say it you can’t think it” (Orwell/Wittgenstein)

Having more concepts allows more flexibility: for example, if there is no array construct in the language, you are restricted in what you can do easily

“Those who cannot remember the past are condemned to repeat it” (George Santayana)

Avoid re-implementation and old mistakes: wise people learn from the mistakes of others

Language Families

So to help think about concepts we classify language into families

Language Families

So to help think about concepts we classify language into families

But it is important to remember that families are **not exclusive**, a language can sit comfortably in more than one family

Aside

One of the many places where we will need to think about terminology is *vectors* and *arrays*

Aside

One of the many places where we will need to think about terminology is *vectors* and *arrays*

Sometimes people use these works interchangeably, to mean the same thing

Aside

One of the many places where we will need to think about terminology is *vectors* and *arrays*

Sometimes people use these works interchangeably, to mean the same thing

Sometimes, not, e.g., array is fixed size, vector is variable sized

Aside

One of the many places where we will need to think about terminology is *vectors* and *arrays*

Sometimes people use these works interchangeably, to mean the same thing

Sometimes, not, e.g., array is fixed size, vector is variable sized

Or a vector is 1D, an array is $>$ 1D

Aside

One of the many places where we will need to think about terminology is *vectors* and *arrays*

Sometimes people use these works interchangeably, to mean the same thing

Sometimes, not, e.g., array is fixed size, vector is variable sized

Or a vector is 1D, an array is $>$ 1D

And sometimes people use “list” where others would use “vector”

Aside

One of the many places where we will need to think about terminology is *vectors* and *arrays*

Sometimes people use these works interchangeably, to mean the same thing

Sometimes, not, e.g., array is fixed size, vector is variable sized

Or a vector is 1D, an array is $>$ 1D

And sometimes people use “list” where others would use “vector”

There are many place in CS where people use different words for the same things; or the same word for different things. Be aware of such variations!