Tecnología de la programación

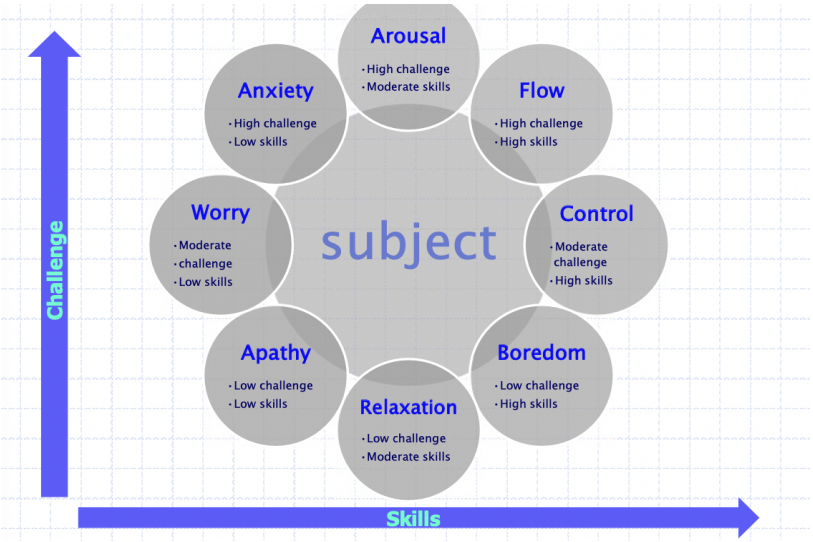
El mejor lenguaje de programación es el lenguaje que te haga escribir menos dependiendo del problema.

Hacer más con menos

Elegir un lenguaje

- Cuanto más expresivo de alto nivel sea el lenguaje, menos errores cometeremos
- La alta expresividad suelen conllevar peores prestaciones
  - Balancear el coste del programador
- Puntos importantes
  - El **paradigma** del lenguaje de programación debe ser adecuado al problema y a los programadores
  - Cuanto más rico sea el lenguaje en **estructuras de datos**, más expresividad permite
  - Un buen sistema de **tipado** bien entendido y usado

Challenges vs skills



Paradigms of programming languages

- Imperative / Procedural
- Functional / Aplicative
- Logic
- Object-Oriented

- Concatenative / Stack-based
- Concurrent
- Scripting

In reality, very few languages are "pure"
- Most combine features of different paradigms

Where do paradigms come from?
- Paradigms emerge as the result of social processes in which people develop ideas and create principles and practices that embody those ideas
- Programming paradigms are the result of peoples ideas about how programs should be constructed
    - linguistic mechanisms for expressing them
        - no se puede hacer una gramática formal del español
    - software engineering principles and practices for using the resulting programming languages to solve problems

Imperative Paradigm
- Imperative (Procedural) programs consists of actions to effect state change, principally through assignment operations or side effects
    - Fortran, algol, cobol, ada, C
- Serie de acciones que cambian los estados del ordenador. El ordenador es una máquina de estados.
- Secuencia de ordenes
- Instrucciones que hace el ordenador

Object-Oriented Programming (OOP)
- Is not always imperative, but most OO languages have been imperative
    - Simula, Smalltalk, C++, Java
- Objetos que interactuan entre si
- Objeto es algo con propiedades y acciones

Functional and logic paradigms
- Focuses on function evaluation; avoids updates, assignment, mutable state, side effects
- Not all functional languages are "pure"
    - in practice, rely on non-pure functions for input/output and some permit assignment-like operators

    - SQL is functional
    - motores no compiladores
- Logic programming is based on predicate logic
    - Targeted at theorem-proving languages, automated reasoning, database applications (Prolog)
    - Recent trend: declarative programming (XSLT)

Concatenative and Stack-based Paradigms

Concatenative

- Programming language in which terms correspond to functions and in which the juxtaposition of terms denotes the composition of functions
- Concatenative programming replaces function application, common in other programming style, with function composition as the default way to build subroutines

Stack-based

- Programming languages operate upon one or more stacks, each of which may serve different purposes
- Programming constructs in other programming languages may need to be modified for use in a stack-oriented programming language

Examples: FORTH, factor, PostScript, min

Concurrent and Scripting Languages

- concurrent programming cuts across imperative, object-oriented and functional paradigms
    - Pascal
- Scripting is a very "high" level of programming
    - Rapid development: glue together different programs
    - Often dynamically typed with only int,float,string and array as the data types; no user-defined types
    - Often weakly typed: a variable "x" can be assigned a value of any type at any time during execution
- Very popular in Web Development
    - Especially scripting active web pages

Unifying concepts

- Unifying languages concepts
  - Types(both built-in and user-defined)
    - Specify constraints on functions data
    - Static vs dynamic typing
    - Conjunto determinado de reglas para trabajar con valores
  - Expressions (example: arithmetic, boolean, strings)
    - combinar diferentes variables para conseguir un resultado
  - Functions/procedures
    - Function
      - devuelve valor
    - Procedimiento
      - no devuelven valor
  - Commands
    - palabra reservada
    - if

First-class entities

- Programming language theorist define a first-class entity as a program entity that can be:
  - Created at runtime
  - Assigned to a variable or element in a data structure
  - Passed as an argument to a function
  - Returned as the result of a function
- Java
  - objetos
    - instancia una clase

Integers, strings, and sometimes real numbers are typical examples of first-class entities

But classes, functions, objects are not always first-class entities in several programming languages

Le metes nuevos comandos añades vocabulario

Lenguaje normal y tiempo real

- Normal
  - dar respuesta correcta

- tiempo real
    - tienes tiempo para decir la respuesta

Estructuras de datos

- Un lenguaje es más rico cuanto mejores son sus tipos de datos

Definición de dato

- Dato
    - Unidades de tratamiento dentro de un sistema informático.
    - Los datos de entrada junto con los datos de salida constituyen lo que se denomina información
- Diseño de un programa
    - Definir estructuras de datos
    - Definir operaciones sobre los datos
- Componentes de un dato
    - IDENTIFICADOR
        - Nombre utilizado para referenciar al dato
    - TIPO
        - Rango de valores que puede tomar el dato
    - VALOR
        - Elemento que pertenece al rango de valores según el tipo
    - int num = 5;
        - int - tipo
        - num - identificador
        - 5 - valor

Clasificación de los datos

- Datos básicos
    - Numéricos
    - entero real
- Datos derivados
    - puntero
- Datos estructurados
    - internos
    - externos
    - compuestos
    - estáticos
    - dinámicos

    - lineales
    - no lineales
    - tabla
    - registro
    - base de datos

Datos lógicos

- Un dato lógico o booleano describe a una magnitud que tiene sólo dos valores o estados
    - verdadero-falso, si-no, etc
- Su representación depende del lenguaje de programación
    - true, false, 0/1
- Se basan en el álgebra de Boole
    - Operaciones
        - Suma lógica (+)
        - Producto lógico (.) AND
        - Negación Lógica (-) NOT

Almacenamiento de datos multibyte

- Endianness
    - El término inglés endianness ("extremidad") designa el formato en el que se almacenan los datos de más de un byte en un ordenador
    - Tipo de endianness
        - Little-endian
            - intel
                - CISC
        - Big-endian
            - tcp/ip
        - Bi-endian
            - ARM (ACORN RISC MACHINE)
                - RISC
- Datos numéricos
    - Numéricos
        - Numérico entero con y sin signo
        - Numérico real
        - Otros: racional, monetario, complejo,etc
    - Numérico entero (sin signo)
        - Binario puro

        - longitud de la palabra
    - Decimal codificado en binario (BCD)
        - Agrupaciones de 4 bits por cada dígito decimal
- Numérico entero (con signo)
    - Binario (Módulos y signo MS)
        - signo en el bit más a la izquierda
    - Complemento a 1 (C-1): igual a MS para positivos
        - Negativos
            - cambio 0s por 1s
    - Complemento a (C-2)
        - Igual a MS para positivos
            - Negativos
                - C-1 y suma 1
    - Exceso a 2^n - 1
        - No hay bit para el signo
            - valor = número + exceso
    - Decimal desempaquetado, decimal empaquetado
- Número real
    - Números muy grandes o pequeños
        - Representación
            - Punto decimal
                - dígitos del 0 al 9 más el punto decimal
            - Notación científica o exponencial
                - Torres Quevedo
                    - Numero = mantisa + base^característica
                    - Mantisa
                        - número real
                    - Base de exponenciación (E) Base decimal
                    - Característica
                        - Exponente correspondiente a un número entero con su signo
                - Numérico Real
                    - norma IEEE 754

- Datos de carácter
    - Utilizados para representar un carácter dentro del juego de caracteres definido por el fabricante del ordenador

---

- Datos de carácter
  - Utilizados para representar un carácter dentro del juego de caracteres definido por el fabricante del ordenador
- Juegos de caracteres clásicos
  - ASCII
    - American Standard Code for Information Interchange
    - El código ASCII es un código de 7 bits que se suele almacenar en 8 bits
  - EBCDIC
    - Extended Binary Coded Decimal Interchange Code
    - El código EBCDIC es un código de 8 bits
- UNICODE
  - Representa caracteres (characters) mediante números (code points) que se dibujan mediante ideograma o glifos (glyphs)
  - Todos los lenguajes del mundo están incluidos
  - La más usada y conocida es UTF - 8

UNICODE

- Tipos de datos: datos derivados
  - Se emplean para referenciar otro dato (de otro tipo) mediante algún medio, por ejemplo usando la dirección de memoria del otro dato (puntero)
  - Puntero
    - Llegar a otro dato
    - punteros tienen tipo
    - llegar al contenido
    - Punteros son una dirección en memoria que apuntan a un dato para poder llegar al contenido
  - Referencias
    - Apuntan a donde esta algo pero sin saberlo
  - Características de los datos de tipo puntero
    - Tipo
      - Mismo tipo que el del dato al que apunta
    - Contenido
      - La dirección del dato al que apunta
    - Indirección
      - Referencias indirectas al dato que apunta
    - Reutilización

            - Se puede utilizar para apuntar a otro dato del mismo tipo que el puntero
- Tipos de datos: datos estructurados
    - Internos
        - residen en la memoria principal del ordenador
            - RAM
    - Externos
        - residen en la memoria auxiliar del ordenador
            - Disco
            - SSD
        - Dispositivo de almacenamiento masivo
    - Estáticos
        - Su tamaño no puede cambiar durante la ejecución del programa
    - Dinámico
        - Su tamaño puede cambiar durante la ejecución del programa
    - Lineales
        - Sólo puede estar enlazados a un elemento anterior y uno posterior
    - No lineales
        - pueden enlazarse a más de un elemento, de diferentes formas
    - Compuestos
        - Formados por el programador mediante agregaciones de datos básicos y derivados
- Tipos de datos: estructuras estáticas
    - Tablas
        - Estructuras estática de datos constituida por un número fijo de elementos del mismo tipo en direcciones de memoria contiguas
            - Random access
                - Secuencial
        - Clasificación
            - Unidimensionales
                - vectores
                - arbitrario
                    - el que me de la gana
            - Bidimensionales
                - matrices
            - Multidimensionales
                - poliedros
    - Lista

        - Estructura dinámica de datos constituida por elementos (habitualmente, pero no necesariamente, del mismo tipo) denominados nodos. El orden de los elementos es significativo
            - PILA
                - Lista basada en el criterio LIFO
                    - Last in first out
                        - el último que entra es el primero que sale
                - Los elementos se añaden apilándose unos sobre otros y solo se tiene acceso al que esta en la parte superior
                - Solo puedes ir al último elemento
            - COLA
                - Lista basada en el criterio FIFO
                    - First in first out
                        - el primero que entra es el primero que sale
                - Los nuevos elementos se añaden en la parte de atrás y la extracción de elementos se realiza por la de delante
    - Conjunto
        - Estructura dinámica constituida por elementos (habitualmente, pero no necesariamente, del mismo tipo).
        - Los elementos no están ordenador y solo es significativa su pertenencia o no al conjunto y la posibilidad de que estén repetidos o no
        - acceso secuencial
        - idempotente
            - lo meto 3 pero solo esta 1
    - Diccionario o MAPA
        - Estructura dinámica de datos formada por una serie de parejas
            - clave = > valor
        - El orden de las parejas puede ser significativo o no, así como la repetición de claves
        - Las claves son únicas y no pueden cambiar
            - clave tiene que tener función de comparación
    - Árbol
        - Representación de estructuras jerarquizadas
        - cada nodo tiene un padre
        - Un árbol esta balanceado cuando todas las capas menos la última están completas
        - Recorrer por profundidad o anchura

    - Grafo
        - conjunto de nodos conectados por arcos

Type Systems

- Introduction to type systems
    - Assigning datatypes ("typing") gives meaning to collections of bits
    - A type system stipulates the ways typed programs may behave and makes behaviour outside these rules illegal
- In computer science, a type system defines how a programming language classifies values and variables into types, how it can manipulate those types and how they interact
- Propiedades de un type systems
    - type checking
        - static
            - tira hacia el static pero no al 100%
            - Every name is bound both to a type (at **compile time**, via a data declaration) and to an object.
            - Solo puede ser asociado a un objeto con el mismo tipo que el nombre
            - int n = 3
                - 3 es objecto
                - int tipo
                - n es nombre o referencia
                - n es de tipo int
                - el objecto 3 es de tipo int
                - float n = 3
                    - error
                    - chequeo estático de tipos
                - Una variable se declara para indicarle al programa a partir de qué lugar empieza a existir, qué nombre tendrá y qué tipo de datos almacenará. La asignación de un valor inicial se llama inicialización.
                - No se comprueban los valores
                - ventajas de hacer el chequeo en tiempo de compilación
                    - sabes que todos los errores de tipado ya estan solucionado
                - desventajas
                    - fiar toda la seguridad en un lugar en el compilador

- dynamic
  - Every name is (unless it is null) bound only to an object
  - execution time
  - nombres no tienen tipo
  - tienen sus cositas static
  - a = 3
    - a no tiene tipo
    - 3 tiene tipo de entero
  - puede compilar o no dar error a la hora de escribirlo
    - los lenguajes interpretados no se compilan
  - el error lo da cuando ejecutamos
    - REPL
      - Read - Eval - Print Loop
  - makes metaprogramming more powerful and easier to use
    - mas haya de la programación
    - los programas los escriben los programas
- Static vs Dynamic type checking
  - Benefits of static type checking
    - error/ warning in your editor
      - comprueba los tipos antes de ejecutar
    - Generics (but this can be also a negative)
      - templates
        - tipo de un arrayList por ejemplo
    - ability to follow a chain and figure out what type of object is required at each step
    - refactoring
    - common in compiled languages, considered "safer"
  - Benefits of dynamic type checking
    - more concise and precise, less typing
    - no badly implemented generics
    - closures/functions
      - clausura
        - es una función
        - def cm(n)
          - def mi(x)
            - return (x*n)
          - return mi
        - p3 = cm(3)

            - p3(5) = 15
        - clausura son común en js
    - REPL: Read-Eval-Print Loop
    - common in interpreted languages
- Escribir el tipo en static
    - int
    - declarar
- safety
    - type cast y conversion
        - cast
            - fuerzas al tipo a cambiar
            - es una trampa
        - conversion
            - cambiar el tipo
            - cambiarlo en memoria
    - unsafe
        - if allows these conversions (type coercion)
    - safe
        - if it does not allow operations or conversions which lead to erroneous conditions
            - a = 3
            - b = "hola"
            - a + b = "3hola"
        - La coerción es la conversión automática o implicita de valores de un tipo de dato a otro (Ejemplo: de cadena de texto a número). La conversión es similar a la coerción porque ambas convierten valores de un tipo de dato a otro pero con una diferencia clave - la coerción es implícita mientras que la conversión puede ser implícita o explícita.
        - hace cosas por debajo de la mesa
    - JS es seguro
    - una conversion viola o no los tipos
        - no
- accuracy
    - weak
        - si las permite
    - strong

        - languages doesn't allow an operation to succeed on arguments which have the wrong type
    - type equivalence
        - a type system has always an own definition of equivalence
        - structural
            - types must have the same structure in order to be equal
            - cantidad de bits iguales y mismos
        - duck
            - duck typing
                - the language "guesses" which type is meant
                    - python
                - se hace en tiempo de ejecución y ve que puedes hacer
        - nominative
            - type systems
                - java, c
                - si la familia son iguales
                    - herencia
                - types must have the same name in order to be equal

Obejct-Oriented Programming (OOP)

- Imperative programming vs OOP
    - So what are the paradigms of programming?
        - Imperative programming is the "traditional" model of computation
            - state
            - variables
            - assignment
            - loops
- A processing unit is separate from memory, and "acts" upon memory

Paradigm know as Object-Oriented Programming (OOP)

- A Class
    - Is a description used to instantiate objects
    - Identifies properties (attributes that belong to all objects of the class and behaviors (methods))
- An Object
    - Is an instance of a class, it has a name, attributes and their values, and methods
        - ejemplar de una clase

        - comparten propiedades
    - Models an idea found in reality (tangible or abstract)
- Attributes
    - properties of the object
- Methods (services, messages): behaviors
    - servicio
        - hace algo
- Information hiding and encapsulation
    - a technique in which an object reveals as little as possible about its inner working
- other concepts
    - inheritance, polymorphism, overloading, templates
        - inheritance
            - reutilización
        - polymorphism
            - varias formas
            - una operación con el mas funciona con no solo número pero mas
            - para un objetos de clases diferentes usan el mismo método
        - overloading
            - le hacemos algo a la clase alumno para que pueda sumar
            - sobrecargar un operador
        - overwrite
            - rescribir un método en una subclase
        - Templates
            - plantillas

Class vs objeto

- Las clases crean objetos
- los objetos se pueden modificar clases no
- clases son plantilla
- las que hacen cosas son

=

- asignar referencias

Métodos vs funciones

- método es función dentro de la clase
- función es una función fuera de una clase

Assignment, equality and identity
En c

- x = 3
- y = x
- y = 3
- copia el valor

- Assignments create / manipulate object references
- x = y
  - asignación
  - does not make a copy of y
  - nombre
- x = y
  - makes x reference the object y reference
- A test for identity asks whether two reference refer to exactly the same object
  - hagan referencia al mismo objecto
  - ==
    - equal to
- A test for equality ask whether two references refer to values that are equivalente
  - the meaning of equivalent is inherently domain specific
  - ===
    - equal value and equal type

```
>>> a = [1, 2, 3]
>>> b = a
>>> a.append(4)
>>> print(b)
[1, 2, 3, 4]
```

- El array tiene dos nombres a y b

Kay's description of OOP

- Object- oriented programming is based on the principle of recursive design
  - everything is an object
    - actions are performed by agents, called instances or objects

- objects perform computation by making requests of each other through the passing of the messages
    - actions are produced in response to requests (messages)
    - an instance may accept a message, and in return will perform an action (method) which use its data (attributes) and return a value (usually another object)
- Every object has it's own memory, which consists of other objects (encapsulation)
    - each object is like a miniature computer itself - a specialized processor performing a specific task
- Every object is an instance of a class, so a class:
    - groups similar objects
    - is a repository for behavior associated with an object

Elements of OOP

- Inhertiance
    - Ability to define classes that are extensions of other classes with new and/or specialized attributes and methods
- Overriding
    - Subclasses can alter or overrride information inherited from parent classes
- Polymorphism
    - Ability to use same name for methods
    - Allows to manipulate objects without knowing their exact type but only their common property
- Overloading
    - An existing operator, such as + or = is given the capability to operate on a new data type

Que hace que un lenguaje sea orientado a objetos

- Propiedad básica de un un OOP
- Dynamic binding
    - Process of identifying at run time what code should be executed as a result of a message
    - The ability of invoke a derived class method via a base class reference is called virtual method invocation or dynamic binding
- This is the one true characteristic of OOP

    - A language with classes, attributes, methods, inheritance, polymorphism, overloading, templates, etc is said to be object bases, but not object-oriented

OOP Design - SOLID

- Single Responsibility Principle (SRP)
    - A class should have only one reason to change
    - Cuando una clase haga mas de una cosa
    - Objetos tienen que ser sencillos dedicarse a una cosa
- Open-Closed Principle (OCP)
    - Software entities (classes, modules, functions, etc) should be open for extension but closed for modification
    - No puedes modificar lo que ya tienes
- Liskov Substitution Principle(LSP)
    - Subtype must be substitutable for their base types
        - basic OOP discusses inheritance with IS-A
    - Donde pasa una persona puede pasar un alumno
- Interface Segregation Principle(ISP)
    - Clients should not be forced to depend on methods they do not use
    - Otro objeto hace la petición
    - Dependo de tu interfaz
- Dependency Inversion Principle (DIP)
    - High-level modules should not depend on low-level modules. Both should depend on abstractions
    - Abstractions should not depend on details. Details should depend upon abstraction.
    - Usando solo interfaces para programar
    - Independencia
    - Persona no depende de erasmus pero erasmus depende de persona

Covariance and contravariance

- Formal definition: within the type system of a programming language, a typing rule or a type constructor is:
    - covariant if is preserves the ordering of typed which orders types from more specific to more generic
    - contravariant if it reverses this ordering
    - bivariant if both of these apply
    - invariant or nonvariant if neither of these applies

- Example arrays
  - First consider the array type constructor from the type Animal we can make the type Animal [ ] ("array of animals") Should we treat this as
    - Covariant: A cat [ ] is an Animal [ ]
    - Contravariant: an Animal [ ] is a Cat [ ]
    - or neither (invariant) ?

Law of Demeter

- Also called Don't Talk to Strangers
  - Each class should only use a limited set of other classes: only units closely related to the current unit
  - Each class should only talk (send messages) to its friends or "Don't talk to sytrangers"
  - if objects traverse long object structure paths and send messages to distant, indirect (stranger) objects, the system is fragile with respect to changes on the object structures - a common point of instability in systems
  - LoD help us avoid creating such designs