TREBALL DIRIGIT LLENGUATGES DE PROGRAMACIÓ: LLENGUATGE PONY

HashCode: 7472

ÍNDEX DE CONTINGUTS

ESTUDI DEL LLENGUATGE

1. INTRODUCCIÓ

Pony és un llenguatge de programació de codi obert creat el 2014 per Sylvan Clebsch (un desenvolupador de programari).

La idea darrere de Pony va ser crear un llenguatge de programació que fos segur, escalable, concurrent, multi-paradigmàtic, d'alt nivell i tolerant a fallades. Per assolir aquests objectius, Clebsch , que es declara fan de certs llenguatges, es va inspirar en diversos llenguatges de programació, inclosos Erlang, Haskell i Ruby per desenvolupar aquest projecte.

Avui dia, tot i ser un llenguatge relativament nou, i per tant tindre un nombre d'usuaris menor a altres llenguatges com Python o Java, està guanyant popularitat en certs àmbits, com els sistemes distribuïts i els jocs, fet que ens fa plantejar-nos el seu futur.

2. HISTÒRIA DEL LLENGUATGE

Pony és fruit de l'experiència, l'aprenentatge, l'anàlisi i altres projectes del seu creador, Sylvan Clebsch.

Durant la seva carrera com a professional, Sylvan ha treballat en diferents àmbits en els que ha après quines mancances tenen certs codis de programació. Primerament, va treballar en una companyia aèria on va desenvolupar un simulador de vol, mentre que en el desenvolupament de la seva llarga carrera, també ha estat en àmbits com el financer on va tractar amb llibreries de C, entre d'altres.

Finalment, va accedir a la Universitat on va fer un doctorat. Durant aquest, va aprendre moltes habilitats, fonamentalment sobre les llibreries de C/C++, així com alhora va conèixer ple de gent. Dins d'aquesta estància, junt amb la seva tutora de doctorat, va emprendre una empresa anomenada Casuality.

Com a fruit d'aquesta, el juny del 2014 sorgeix un nou llenguatge, que es compilarà i s'executarà el setembre. Aquesta era una primera versió del nou producte de la companyia, que fent-ne millores en el compilador i el temps d'execució, acabaren fent-ne d'aquest producte un codi font lliure. Aquesta publicació fou realitzada el maig del 2015.

Però, d'on sorgeix el nom de Pony? Sylvan afirma que ell, com bon amic que és, sempre compartia els seus pensament i propòsits amb els seus amics. Resulta que un dia, va explicar Nathan Mehl el que faria quan hagués creat un llenguatge de programació, i aquest, li va dir: "Sí, i jo vull un pony"; i d'aquí en va sortir el nom que ara bateja aquest llenguatge.

3. ANÀLISI DEL LLENGUATGE

En aquesta secció farem un anàlisi del llenguatge per entendre, entre d'altres coses, el seu funcionament, les seves habilitats, com podem explotar-lo i en quins àmbits es troba present.

3.1 PROPÒSITS DEL LLENGUATGE

Pony és un llenguatge de programació que s'enfoca en proporcionar una alternativa segura i eficient als llenguatges de programació tradicionals. Els principals propòsits d'aquest llenguatge són:

- Seguretat

Un dels principals objectius de Pony és proporcionar un entorn de programació segur que minimitzi la possibilitat d'errors fonamentals, com els de memòria i/o els de concurrència.

Respecte la memòria, no permet modificar punters, totes les excepcions són capturades i no contempla el valor *null*. Pel que fa a la concurrència, aplica un model d'actors que permet

escriure codi concurrent sense haver de preocupar-se de certs problemes com les condicions de carrera *(data-race free)* i els bloquejos *(dead-locks free).*

- Concurrència i paral·lelisme

Un dels propòsits fonamentals de Pony és facilitar la concurrència i el paral·lelisme. El llenguatge està dissenyat per permetre el desenvolupament de codis concurrents de manera segura i eficient i ho aconsegueix gràcies a la comunicació per missatges pròpia del model d'actors. I això per tant, permet poder escriure qualsevol codi concurrent sense preocupar-se per condicions de cursa o bloquejos.

- Productivitat i rendiment

Pony està dissenyat per ser ràpid i eficient. L'ús del model d'actors anomenat anteriorment permet la concurrència i l'execució en paral·lel per aprofitar al màxim els recursos del sistema. A més, el llenguatge ofereix un conjunt de característiques d' alt nivell per ajudar els desenvolupadors a escriure codi net i modular, incloent-ne la inferència de tipus.

- Interoperabilitat

Pony pot interactuar amb altres llenguatges de programació, incloent C, C + + i Python, la qual cosa el fa útil per a una gran diversitat de projectes.

- Llegibilitat

Pony està dissenyat per ser un llenguatge de programació fàcil de llegir i escriure. Això s'aconsegueix mitjançant l'ús d'una sintaxi clara i concisa i l'eliminació de construccions complexes com les excepcions.

3.2 PARADIGMES DE PROGRAMACIÓ

Tal i com s'ha dit anteriorment, Pony és un llenguatge multi-paradigma; és a dir, que combina varis paradigmes de programació. Entre aquests trobem tres dels més coneguts: la programació orientada a objectes, la funcional i la concurrent. Ara però ho explicarem amb més detall:

- Programació orientada a objectes

A Pony cada actor és un objecte especial, i per tant, té un estat, una sèrie de mètodes, una regió de memòria (de 240 bytes) i una comunicació basada en el enviament de missatges. Aquests 240 bytes reservats permeten que cadascun d'aquests actors sigui programat de manera independent ja que cadascun d'ells té una cua, una regió per emmagatzemar les escombraries i la capacitat de fer-ne la seva recollida.

A més, cada actor, pot modificar el seu estat i pot crear nous actors, i no només això, sinó que el que realment aporta Pony en comparació amb altres llenguatges de model d'actor és la idea que cal aprofitar tots els subsistemes d'un ordinador per a la interpretació, per tant, Pony intenta explotar al màxim la productivitat no només del codi, sinó del sistema o CPU que té l'usuari.

- Programació funcional

Pony tot i no ser un llenguatge purament funcional, si que suporta aquest paradigma. Inclou les funcions d'ordre superior (aquelles que permeten passar funcions com a arguments i retornar funcions), les funcions pures (aquelles que permeten dividir el codi en funcions per millorar-ne la seva llegibilitat) i la creació de variables immutables o estàtiques (no canvien el seu valor en la execució), propietats que permeten la creació de codi modular i reutilitzable.

- Programació concurrent

El model d'actors és el secret de la programació concurrent a Pony. Els actors són objectes que s'executen en paral·lel i es comuniquen mitjançant l'enviament de missatges asincrònics.

- Tipus estàtic

Pony és un llenguatge de programació de tipus estàtic, el que significa que els tipus es verifiquen en temps de compilació en lloc de temps d'execució. Això ajuda a prevenir errors en temps d'execució i millora el rendiment del codi.

3.3 SISTEMA D'EXECUCIÓ

El seu sistema d'execució de Pony està dissenyat per proporcionar una execució eficient i simultània dels programes. Per aconseguir-ho però utilitza alguns procediments i elements clau que explicarem.

- Capacitats de referència

Pony utilitza un concepte anomenat "capacitats de referència" per gestionar la seguretat de la memòria. Aquestes defineixen els permisos que té un objecte per compartir i mutar el seu estat i són: "iso" (propietat exclusiva), "val" (accés de només lectura compartit), "ref" (accés mutable compartit), "box" (objectes immutables i assignats a la pila) i tag. Aquestes capacitats permeten a Pony aplicar un accés simultani segur a la memòria.

- Recollida d'escombraries

Pony utilitza un col·lector d'escombraries (GC) per gestionar automàticament el desús de memòria. El GC a Pony és gestionat per cada actor, és a dir, cada actor recull la seva pròpia memòria deixada de usar i ho fa aplicant el protocol Orca del que en podem trobar informació gràcies a la seva importància en certs llenguatges.

- Compilador

Els programes Pony es compilen en codi màquina mitjançant el compilador Pony. El compilador realitza diverses optimitzacions i verifica la correcció del programa (mitjançant les informacions de tipus de cada variable), les regles de capacitat de referència i/o de memòria.

3.4 APLICACIONS DEL LLENGUATGE

Pony es pot fer servir en diversos àmbits on la concurrència, la seguretat i el rendiment són de vital importància. Tot i ser un llenguatge relativament nou s'està utilitzant cada vegada més en diferents aplicacions, entre elles:

- Sistemes distribuïts

Pony és especialment adequat per desenvolupar sistemes distribuïts, on la concurrència i la comunicació entre components són crucials. La seguretat d'aquesta última, permet la creació de sistemes distribuïts altament concurrents i tolerants a fallades.

- Sistemes de processament de dades en temps real

La gestió segura i eficient de la concurrència que fa Pony permet el seu ús en sistemes de grans volums de dades en temps real els sistemes de transmissió, sistemes que requereixen d'una resposta ràpida i precisa.

- Aplicacions i serveis web

Pony s'usa en aplicacions web que necessiten manejar una gran quantitat de sol·licituds concurrents. Aquest ús permet millorar el rendiment i l'escalabilitat d'aquestes aplicacions web.

- Jocs i simulacions

Els jocs i simulacions solen requerir un alt rendiment, una gestió eficient de la concurrència i una comunicació segura i ràpida entre jugadors. Pony pot ser utilitzat per aprofitar al màxim els recursos del sistema i gestionar la lògica del joc.

Aquests són només alguns exemples dels àmbits en què es pot aplicar Pony i és gràcies a la flexibilitat del llenguatge, el seu enfocament en la concurrència i la seguretat de tipus que el fan adequat per a un gran àmbit tecnològic i professional.

3.5 SISTEMES DE TIPUS

El sistema de tipus de Pony és estàtic. És a dir, Pony és un llenguatge on el compilador sap el tipus de tot el que hi ha al programa. A diferència dels llenguatges escrits dinàmicament, en Pony els errors es comproven en temps de compilació mentre que en altre llenguatge com ara Python, Lua, JavaScript i Ruby es fa en temps d'execució.

El sistema de tipus Pony ofereix moltes garanties, entre elles, ens garanteix que si el programa es compila, no es bloquejarà, sabem que es gestionaran totes les excepcions i la no-existència del valor null.

Tot i això, els aspectes i els tipus fonamentals de Pony son:

1. Classes

Igual que altres llenguatges orientats a objectes a Pony les classes es declarem amb la paraula clau *class*, i han de tenir un nom que comenci amb una lletra majúscula. Aquestes són composades per:

- Camps: hi ha tres tipus; var, let i els "camps incrustats". Per definir que un camp és privat s'utilitza la barra baixa i el funcionament d'aquest és similar a altres llenguatges com C.
- Constructors: es creen amb la paraula *new*, tenen noms i poden tindre altres paràmetres
- Funcions: Les funcions de Pony són com els mètodes de Java, C#, C++, Python entre d'altres. S'introdueixen amb la paraula clau *fun*, poden tenir paràmetres així com un resultat (si no es dona cap tipus de resultat, el valor predeterminat és *none*). També existeixen un tipus de funcions especials anomenades *finals**

*Les *finals* són funcions especials. S'anomenen _final i no prenen. En altres paraules, la definició d'una *final* ha de ser *fun* _final().

2. Primitives

Una primitiva es similar a una classe, però amb dues diferencies clau: no te camps (variables, estructures, ...), per tant són immutables i tenen una mida fixa en memòria i són úniques (només hi ha una instancia de una primitiva definida per l'usuari).

Aquestes s'usen fonamentalment com a:

- Valor marcador: Pony sovint fa servir el primitiu *none* per indicar que alguna cosa "no té valor" i d'aquesta manera crear-ne una mena de null.
- Tipus d'enumeració.
- Col·lecció de funcions: les primitives poden tenir funcions i permeten agrupar funcions.

Els mes coneguts son: enters de diverses mides (per exemple, I32 per a enters de 32 bits), números en coma flotant (F32 i F64), caràcters i booleans.

3. Structs

A Pony, les estructures tenen moltes semblances amb les classes. Ambdues es componen d'alguna combinació de: camps (es defineixen igual que les classes Pony), constructors i funcions.

4. Actors

Els actors son com les classes però amb la gran diferencia que aquests tenen **"**behaviours". Un "behaviour" és com una funció, excepte que les funcions són sincròniques i aquests són asíncrones. En altres paraules, quan crideu a una funció, el cos de la funció s'executa immediatament però en el cas dels "behaviour" s'executarà en un moment indeterminat en el futur.

Aquests s'introdueix amb la paraula clau *be*.

5. Àlies

Un àlies de tipus és només una manera de donar un nom diferent a un tipus. Les enumeracions i els tipus complexes són dos exemples d'aquest ús.

6. Expressions

Els tipus dels quals hem parlat fins ara també es poden combinar en expressions de tipus. Una expressió de tipus també s'anomena tipus de dades algebraiques. Hi ha tres tipus d'expressió de tipus:

- *Tuples* o estructures: són una seqüència de tipus que no té cap codi associat i ens permet emmagatzemar més d'un valor
- Unions: s'escriu com una tupla, però utilitza un |. Mentre una tupla representa una col·lecció de valors, una unió representa un valor únic que pot ser qualsevol dels tipus especificats. S'utilitzen per a valors opcionals, enumeracions, valors de marcadors i molt més.
- Interseccions

  Una intersecció utilitza & un entre els seus elements. Representa exactament el contrari d'una unió: és un valor únic que és tots els tipus especificats.

I a més, això va més enllà ja que les expressions de tipus es poden combinar en tipus més complexos.

7. Inferència de tipus

Pony compta amb un sistema d'inferència de tipus que pot deduir automàticament els tipus d'expressions i de variables en moltes situacions. Això permet una escriptura de codi més concisa i evita la necessitat especificar explícitament els tipus en tots els casos.

3.6 CARACTERÍSTIQUES DEL LLENGUATGE

En aquesta part de l'anàlisi no només explicarem les característiques de Pony, sinó que es farà una mena de pros i contres per poder conèixer quin potencial i quines limitacions té Pony.

AVANTATGES	MANCANÇES

Dades immutables compartides amb seguretat: qualsevol dada que sigui immutable (és a dir, que no es pot canviar) és segura d'utilitzar. Al ser és immutable, mai s'actualitza i són les actualitzacions les que causen problemes de concurrència. Les dades aïllades són segures: són aquelles que només tenen una referència. Al tindre una referència, les dades aïllades no es poden compartir, de manera que no hi ha problemes de concurrència. Un sol fil per actor: el codi d'un sol actor mai s'executa simultàniament, és a dir, dins d'un únic actor, les actualitzacions de dades no poden causar problemes. Determinista: El model d'execució de Pony s'esforça per assolir determinisme en l'execució dels programes. És a dir, amb les mateixes entrades, un programa Pony produirà sempre el mateix resultat, cosa que facilita la comprensió, depuració i prova dels programes. Rendiment, seguretat i concurrència: Pony se centra a oferir un rendiment eficient. A més, el model d'execució basat en actors pot aprofitar el paral·lelisme i la concurrència de manera eficient, evitant errors de memòria gràcies a la verificació de tipus feta a la compilació.	Falta d'estabilitat de l'API: Pony encara no ha arribat a la versió 1.0. Periòdicament tenim versions que impliquen canvis de ruptura. Aquesta manca d'estabilitat és una raó per a molts projectes per evitar l'ús de Pony. Manca de biblioteques de tercers d'alta qualitat: Un projecte tindrà èxit o fracassarà en funció de la mida de la comunitat al voltant de les eines que es facin servir. El conjunt de biblioteques de codi obert, és molt petit, és a dir, si es necessita una eina que no es troba a la biblioteca estàndard, és probable que s'hagi d'afegir "manualment", ja sigui escrivint-lo des de zero a Pony o important una biblioteca C. Corba d'aprenentatge: Pony és un llenguatge de programació relativament nou i menys conegut en comparació amb altres llenguatges. Això vol dir que hi ha menys recursos, biblioteques i comunitats de suport disponibles per als programadors de Pony. Per tant, el seu aprenentatge pot ser més difícil. Mida de la comunitat: Encara que Pony és de codi obert i té una comunitat activa, la seva mida és relativament petita en comparació amb llenguatges més populars. Això pot afectar la disponibilitat de documentació, biblioteques i exemples de codi, cosa que pot dificultar el desenvolupament de projectes.

3.7 PONY I ALTRES LLENGUATGES

Erlang és un dels llenguatges dels que Sylvan es declara fan i per això, es va basar per fer Pony. Ambdós s'utilitzen sobretot en la programació concurrent i en els sistemes distribuïts, i per això l'utilitzarem per fer-ne una visió general de les diferències:

1. Model de concurrència

A Erlang, la concurrència s'aconsegueix mitjançant processos petits que també es comuniquen mitjançant el pas de missatges. Pony tot i també usar el model d'actors, utilitza missatges asíncrons enfocant-se en la seguretat i la concurrència.

2. Tipologia

Una de les principals diferències entre Pony i Erlang és la tipologia del llenguatge. D'una banda, Pony té un sistema d'escriptura estàtica amb una forta comprovació de tipus, que ajuda a detectar errors de tipus en temps de compilació. D'altra banda, Erlang té un sistema de tipus dinàmic, on els tipus es comproven en temps d'execució. Això permet més flexibilitat, però pot introduir un major risc d'errors relacionats amb el tipus.

3. Seguretat

Pony es centra a proporcionar seguretat i correcció a nivell de tipus, evitant errors comuns com a *data-race conditions* i bloquejos. A més, la filosofia de Pony es detectar en temps de compilació els erros, per tant, aplica una política preventiva mentre que Erlang fa tot el contrari. Erlang aplica un enfocament de tolerància a errors que permet als processos reiniciar en cas d'error. Aquesta filosofia permet sistemes altament tolerants a errors.

4. Rendiment

En termes de rendiment, tant Pony com Erlang poden aconseguir una execució simultània eficient. Pony té com a objectiu proporcionar una execució eficient mentre que Erlang prioritza la tolerància a fallades i la fiabilitat.

5. Suport, eines i llibreries

Pony té una comunitat més petita en comparació amb Erlang. És un llenguatge relativament nou i per això disposa de menys biblioteques i eines disponibles. Tot i que ambdós llenguatges estan creixent exponencialment, Pony té una comunitat menor i per tant, dificulta la seva informació sobre errors, implementacions i ús de certs aspectes.

4. EXEMPLES DE CODI

ponyc / examples / helloworld / **main.pony**

```
1    actor Main
2      new create(env: Env) =>
3        env.out.print("Hello, world.")
```

En aquest primer exemple veiem com és el codi més simple possible amb el llenguatge Pony. Si ens fixem podem veure com Pony també té certes funcions implícites com *print(),* molt coneguda ja en altres llenguatges.

ponyc / examples / circle / **main.pony**

```
3    class Circle
4      var _radius: F32
5
6      new create(radius': F32) =>
7        _radius = radius'
8
9      fun ref get_radius(): F32 =>
10       _radius
11
12     fun ref get_area(): F32 =>
13       F32.pi() * _radius.pow(2)
14
15     fun ref get_circumference(): F32 =>
16       2 * _radius * F32.pi()
```

```
18   actor Main
19     new create(env: Env) =>
20
21       for i in Range[F32](1.0, 101.0) do
22         let c = Circle(i)
23
24         var str =
25           recover val
26             "Radius: " + c.get_radius().string() + "\n" +
27             "Circumference: " + c.get_circumference().string() + "\n"
28             "Area: " + c.get_area().string() + "\n"
29           end
30
31         env.out.print(str)
32       end
```

En aquest segon exemple, podem veure com es defineix una classe, quines paraules clau s'usen per declarar variables i funcions. A més, a la dreta podem veure com es creen els actors i l'alta semblança entre ambdós tipus. Amb aquest exemple podem veure tot el que s'ha comentat anteriorment.

ponyc / examples / mixed / **main.pony**

```
actor Worker
  var _env: Env

  new create(env: Env) =>
    _env = env

    var a: U64 = 86028157
    var b: U64 = 329545133

    var result = factorize(a*b)

    var correct =
      try
        (result.size() == 2) and
          (result(0)? == 86028157) and
          (result(1)? == 329545133)
      else
        false
      end
```

```
  fun ref factorize(bigint: U64): Array[U64] =>
    var factors = Array[U64](2)

    if bigint <= 3 then
      factors.push(bigint)
    else
      var d: U64 = 2
      var i: U64 = 0
      var n = bigint

      while d < n do
        if (n % d) == 0 then
          i = i + 1
          factors.push(d)
          n = n / d
        else
          d = if d == 2 then 3 else (d + 2) end
        end
      end

      factors.push(d)
    end

    factors
```

Finalment, amb aquest codi es pot veure una mica més quins són els tipus que hi ha a Pony, com es relacionen els mètodes amb les variables, com es creen primitives de caràcter privat i com es poden fer certes operacions no només matemàtiques.

5. CONCLUSIONS PERSONALS

Finalment, després d'informar-m'he i explicar de la millor manera possible el que s'ha après sobre Pony, n'extrauré les meves conclusions exposant-ne una mica el que en penso sobre l'ús, el futur i l'impacte que pot tindre Pony tant en el present com en el futur.

Sí que és cert que Pony ens permet estalviar-nos problemes amb memòria i per tant, evitar-nos maldecaps alhora de programar i encara més si volem explotar el paral·lelisme i el màxim rendiment de les nostres CPU's. Tanmateix, crec que el fi no justifica els mitjans i òbviament a tots ens agrada tindre sempre el màxim rendiment en tot el que fem, però a quin preu?.. Per tant, per a mi, hi ha aspectes que s'han de tindre en compte i que inclús podrien ser més rellevants.

Cal remarcar però que tindre un llenguatge que s'enfoqui en explotar al màxim el paral·lelisme i la concurrència aplicant-ne certa seguretat és digne d'admirar. A més, permetre que el programador es centri en desenvolupar el codi i no tant en pensar amb problemes de coherència, entre d'altres, és un aspecte que pot fer-te decantar-te entre aquest tipus de llenguatge o un altre.

Però com hem vist, no tot és perfecte perquè sinó tothom utilitzaria Pony i els llenguatges alternatius estarien exempts. És cert que retreure el fet de les llibreries limitades tampoc és just, ja que aquest llenguatge tot just acaba de néixer i per tant, probablement en un futur no gaire llunyà, en tindrà moltes més i el suport que tindrà Pony per a desenvolupar projectes serà major que en la actualitat.

Per tant, jo seria partidari de donar una oportunitat a aquest llenguatge sobretot en petits projectes dins del món laboral. A partir d'aquí les valoracions dels usuaris i el feedback que donarà la comunitat podrà fer-nos valorar si realment té potencial o no.

ESTUDI BIBLIOGRÀFIC

1. ANÀLISI DE LA INFORMACIÓ

En aquesta secció cal remarcar la dificultat que ha tingut la recerca de informació. Bé és cert que en la pàgina oficial de Pony hi havia molta informació, i que junt amb les entrevistes trobades i els fòrums de la comunitat d'usuaris s'ha pogut realitzar el treball. Tanmateix, s'ha trobat molt poca informació contrastada sobre el llenguatge i s'ha hagut de recórrer a certes eines per verificar i poder contrastar una mica més la informació.

Per tant, és cert que la informació brindada en la pàgina oficial doncs té totes les garanties de ser certa i que per tant, es una font fiable. Per tant, la qualitat de la informació és bona.

2. REFERÈNCIES I CITACIONS

Pony - Pony. (s. f.). Pony. https://www.ponylang.io/

Ponylang. (s. f.). GitHub - ponylang/ponyc: Pony is an open-source, actor-model, capabilities-secure, high performance programming language. GitHub. https://github.com/ponylang/ponyc

TNG Technology Consulting GmbH. (2015, 27 agosto). *Pony - A new, race-free, safe & fast actor programming language* [Vídeo]. YouTube. https://www.youtube.com/watch?v=fNdnr1MUXp8

SL, A. S. (2016, 2 abril). *Pony: Actores, Objetos y Alto Rendimiento*. Altenwald Blog. https://altenwald.org/2016/04/02/pony-actores-objetos-alto-rendimiento/

Allen, S. T. (s. f.). *Introduction to the Pony programming language*. Opensource.com. https://opensource.com/article/18/5/pony

Drossopoulou, S. (2020, 30 agosto). Pony, Actors, Causality, Types, and Garbage Collection. *InfoQ*. https://www.infoq.com/presentations/pony-types-garbage-collection/

Sean T. Allen. (2018, 26 octubre). *Pony: How I learned to stop worrying and embrace an unproven technology* [Vídeo]. YouTube. https://www.youtube.com/watch?v=GigBhej1gfI

Sylvan Clebsch on the Actor-Model Language Pony, Garbage Collection, Capabilities, Concurrency. (s. f.). InfoQ. https://www.infoq.com/interviews/clebsch-pony/

Overview of Rust and Pony | Schematron. (s. f.). https://schematron.com/offtopic/overview_of_rust_and_pony.html

Introducción al lenguaje de programación Pony - programador clic. (s. f.). https://programmerclick.com/article/3365211216/

Garbage Collection with Pony-ORCA - Pony Tutorial. (s. f.). https://tutorial.ponylang.io/appendices/garbage-collection.html

Clebsch, S. (2016, 5 junio). Using Pony for Fintech. *InfoQ*. https://www.infoq.com/presentations/pony/

Slant - Erlang vs Pony detailed comparison as of 2023. (s. f.). Slant. https://www.slant.co/versus/11675/15833/~erlang_vs_pony

What is your experience with using Pony programming language? (s. f.). Quora. https://www.quora.com/What-is-your-experience-with-using-Pony-programming-language

Pony Programming Language | Hacker News. (s. f.). https://news.ycombinator.com/item?id=33970547