Gossip Programing16 NOV 2025 AD by Nnamdi Michael Okpala

Okay, so basically this is not recording, my audacity is recording, three minutes on this, everything is recording, it's going to be fine. Okay, now, basically guys, this video is very important to me. It's the most first polyglot of networking programming language.

The architecture has infrastructure compliance, structure by design, which is the minimum principle of the Gaussian ecosystem. So basically, we've got to have this thing, it's called the Rifters Way, which is the manifesto of the Rifters programming language. The gossip.gossi programming language, which is the word for polyglot, networking, architectural, or programming language.

It's not assembly, because assembly is an architectural language, but it's the gossi language, it's a gossi. So basically, this is just the Rifters manifesto. It's basically the principles of the guiding infrastructure for something called the Rif, which is a regular expression, a bidirectional system, where you use regular expression to create tokens and parsers, etc.

If you want to go to my riftheyoutube.com slash oblankers playlist, you'll see how it aligns with the overview. But I'm going to develop a rift and gossi lang. The rift is just a part of the rift ecosystem, which is a tool for matching systems, for tokenizer, for systems.

Rift is a tool, it's part of the tool chain, to use something called an R syntax, a regular expression. So you have like an R character single quote, like an RC, or RX, matching like a character set of a regular expression. Or like an R double quote in C, C11, C19, C19, C16, C21.

It doesn't matter, but rift is a flexible translator. It's meant to compete with YAC, with alignment with objectives, like, you know, parting tokens across public good infrastructures. VM 1 has got hybrid and public good systems.

Basically, this is just Rifters way. I'm going to explain to you, like, you know, this I want you guys to remember. And I've got my notes already, so I want you guys to really engrave into your souls, okay.

So basically, this is called rift. The rift is way. So basically, this is the disc of Python.

You know, when you type Python, or .she, in the terminal, you type in Python, .she, import this. You see the Python one. But this is the rift one.

You're going to see it, basically. Let me just give you an example here. So here on the back of my terminal, here, so let me just try to do this.

So I'm on my terminal. I type in Python. So basically, I'm going to go to Python.

.she, that means compile power, import this. Okay. This is called a zen of Python.

Now, the zen of Python is the how it should, how the Python standard should work, you know, for people who are new to Python or to Python. This is zen of Python by team Ben Peters, you know. So this is what the Python one is.

But this is the zen, the beauty, the zen simplicity of Python. Beauty is better than ugly. Explicit is better than implicit, you know.

But these have a deep meaning for the ecosystem, for the fandom, for the people, for the Pythonistas and the Pythonias. You know, because, you know, if you break your philosophy of this, your code is not working, something like that. And you don't have to open the whole specification, everything, you know.

This is what the Python system really is. But I'm not going to talk about the disc of Python. I'm going to talk about the Python math one.

I'm going to go to the scene two. Here. So we've got the Riftless way.

Now, this is for the Riftful chain. So basically, import disc, not data. Okay.

Now, the disc is memory, the structure, the logic. And not the logic, you know, the logic, you know, how the system works, the models, you know, the full-throat models, the error correct protocol, the disc, not the data. Not the stream of bytes, not the tokens, you know, you know, the structure you're representing.

Using sparse graph, or sparse data, sparse, or sparse byte path that's pipelatted graph with K and N clustering, you know. It won't be really formal. Because sparse means there's any way you can do it, model it, to get that result.

The model infrastructure, the struct, the pattern, the struct in C, the cloud looping across, based on verb noun modelling, that's the naming conversion, you know. It refers to the state of the machine, the state, you know, the colour of the state of the machine, the weightless of the machine, using observer to consume a model, you know. How the machine sees the state, you know, and the development of preservation, the pause and resuming, the capacity of port to the frontend.

When you run import disc, you're not loading files, guys. You're restoring the context, you know, the context of the system. You know, it's basically a schema, like, you know, a schema to restore context.

Every time, every time a programme ends in return, you return. Input disc helps you resume your extension, you know. Statement, expression, evaluation.

Statement, statement can be true or false. Boolean evaluation. True or false.

Statement, solution, or false. Expression is true or false, or can express a value, like a number or something, like an object, a model, a structure, an intent, you know, the intent, why the purpose of doing that. Now, guys, guys, guys, this is beautiful, guys.

I want you guys to guide you through, guys, the ripper's way, you know. This is just the first one. Now, let the bytecode hear what the human couldn't say, you know, you know, sometimes we don't know what we can't do, we want to do, you know, we want to do, you know, we'll just try to code the mofiber product, we don't know how to solve it, you know.

Now, the bytecode is the core machine, you know, it didn't go to analytics, you know, you look at the bytecode, like the assembly instructions, the doctors, and see how the system meets in somatic context, you know, you know. Threshold compilers often mean away, you don't optimise any meaning, basically, it's about you human-centric, you're the programmer or the AI system, about you, the human intent, why you wrote that code, you know, it doesn't matter about good or evil, you know, is it good or evil, does it not work against you in the long run, you know, and you're going to have to roll back many updates, you know, allowing bytecode to reflect, but expression, expression of intent, you know, your intent, what do you want to do, and the rifters works forward. Now, the rifter works forward means, you know, you're a rifter, you know, you're the, I'm basically go to rift, test, r-f-i-t-t-e-s, the rifter testing library, you know, you can test rift expressions, now you've got rift, rift, rift is like, for example, you've got, what else, go to rift, r-f-i-t, which is the patimacha, then you've got the rift test, the testing patimacha, you know, then you've got the other ones, like rift, rift, rift is a flexible regulation as firmware, on github.com slash ob-nexis, rift-raff, etc, etc, you know, this is a rift tool change, you know, now rift-raff is for production, well, rift-raff, rift-raff, you know, things that rift are essentially you can use when a client, so match the library systems, you know, it's about positive systems that are coherent, you know, basically, the rift, the riftal walks through, you know, a thread of yarn, my yarn book, the riftal walks forward like a thread to a pin, a pin to a pin, you know, that's not a, that's not metaphoric, that's, oh, that's quite literal, you know, like, you know, I take a thread of a yarn and I'm trying to pin it to something, have an objective, I have a goal, you know, I want to do something, you know, I have a one, I'm doing something to write, you know, a thread is like a, basically, you are certified with thread, a thread is both a process and a story, so every thread you pin, you know, you pin a process, but that process is now part of an ID, PID file, PID.log, you know, and then when you pin that thread, your strainer process, you know, you're removing that observer bubble or the consumer bubble, you know, and then you are now exposing it to danger, you know, the rift doesn't think you look back and it's unnecessary, they move forward with an intention structure, you know, and then we have one pass, sorry, this is my thing, one pass, oh, this is just not pin, one pass, no, okay, one pass, no recursion, to recurse is to break the weave, you know, so every time you've woven a thread, you know, you don't go back, you don't know pin a thread, you know, basically single pass, this is a single pass compiler, you know, it doesn't go through many phases, you know, it seems to be minimal and optimal for the human in the pipeline structure, you don't have to get a bloated system to build on top of it, you know, it doesn't contain stability because, you know, you want basically, you have one pass, one time to do something, you have nothing to break it, you know, you don't have anything, instead of your structure-related, no looped, you know, complete on top of the directed by path of drag resolution, which I'll come back to later, but you know, basically, if you go to github.com, which should be here, if I've opened it correctly, yes, this is something called a semantic region X, this is about works on a bidirectional mapping of layers of, you know, something called a by-path that's a dag grife, you know, the thing about this by-path is a mode of consciousness, you know, it's a mode of consciousness, I won't say to you, this conscious model is very coherent, it works for you and everyone else, you know, this is server X, but you've got a rust server X, but you know, I'm gonna start reading from here, yeah, so go like, it's for governance, you know, so the by-path, wait, I'm just trying to hear, so basically, in Ghostland, we have something called an actor model and a witnessing model, you know, now actor is an actuator, you know, the accent is on behalf, you know, it's axe on its own intent, you know, the accent, no student expression is intent, but you know, it's not enough for it to do it to itself, so basically, Ghostman called a by-path, resolution, so if you remember my by-informatics repository, you got a by-path, you know, when you have an upload, you can use a, I'm just trying to get this done, yes, basically, so what's the motivation in X versus like, you know, mapping of structures or by-paths, graphs or KNN clusters or data, then you go to Hamiltonian, Hamiltonian graph is the dead fresh search, three first and last by-path infrastructure, that's the first thing, that's the stack, now we go to the stack, the stack goes one way, you know, and then you go to the queue, the queue is the first in, first out of the structure, you know, if you want to go to the bio informatics, you can bio informatics, which is like the subset of structural information, that structures means the structure of the substitution, you can go to VVP, which has a vision, VVP, which stands for the MVP, you can go to sparse exits, you can go to, you can run this file called Hamiltonian sync functor main, you can run this file python, just start with python, but you know, this is sync, you can sync 100% of the Hamiltonian and then the third FI, hello, and FI for the reason bio informatics, and you get one more with server X, which is like a by-path tag, so basically, get like a PLP, a modular layer, so it's an observer called muscle model, and basically, if the Hamiltonian algorithm is used to first in, it's come out last, so basically, it's used for uploading, you know, the first in to all you put comes out last, or isn't a stack, then the other one, the Elorian one, Elorian cycles, or the W, is using FOI first in first out to download the data over the server, this is not good enough, so you have a hundred one, so basically, you go live, you got to have a node, which is like a little by-path code for you, a local to your remote repository, because you know, you're using the smarty version X on the, on the github.com, I'm sorry, on the HTTPS R.OBNXS.org, you know, the bypass relationship, you know, you have your own channel, you speak to your own channel, or by-path channel, so basically, you put your channel, or your local channel, you didn't have your remote channel, for me, I have my OBNXS Namdi account, that's my new channel, and I can put my local repository, my computer, to my remote repository, and then push it to the world, you can upload and download, you know, based on my preference, you know, basically, it's not fragmented, because you know, I have my Hamiltonian, I don't, I don't upload to the world, if I don't push my remote, if I don't release my remote, you know, I wish like my remote repository, but this is the, this is the Python code here, Hamiltonian sync functor, now, basically, input is the part, you know, this is like the functor notation here, so, Python sync is basically syncing the operations in Hamiltonian, and you know, which are, which are just ways to transfer the graphs, the graph to the structures, you know, the goal of semantic version is to map a major minor patch, since, symbolically, and literally, to semantic version X, LTS, LTS, or stable model, legacy, and have experimental, using monogot, which is like one ecosystem, one polyglot, one ecosystem, like Python, PyPy, brass, packaged cargo systems, or like a C corner, you know, one system, hybrid, you know, shared, shared across all the infrastructures, recommended, things that are really fragmented, but didn't get them better, based on good, ruined adaptor patterns, and you get polyglots, polyglot infrastructures, talk to any language, any library, you know, that legacy-lang is the world-source polygraph language, okay, now, back to the reason that working as a key principle, so, see that then, have I done the thing? I saw that it's gonna crash, okay, let me just unplug it, it should not crash, okay, wait for it, wait for it, it didn't crash, okay, it should not crash, you know, okay, I open this one, now, so, we got this, each breath, took it in a breath, each precious truth, you know, basically, the tokens are, like, can I say just to start breathing, you know, like, you know, when you have a token, you're speaking words, trying to make words, so, you shouldn't speak your breathing, you should, each token carries a semantic word, you know, what I'm trying to say, not just syntax, you know, the structure, you know, that's why Gussie is a tokenizer, to favour prefix, to make clear intention, tokens are not part of units, they express decision, you know, because this is polyglot, it's talking to a lot of languages, you know, not talking to any, just one language, you have to understand why you're breathing, even, why you're uttering a, a tone, a token, in a tone, in one dialect, you know, ES2s are not roots, but, I'm not true, it's by root of intention, so, basically, this is, like, the standard of token type, token value to come in, more formally, phenomological or genonomological, which means, like, the purpose of talk to the world, doing something, and the structure, the genome, of a makeup, or to the model, to this past structures to do that, you know, that's what it is, in Gussie language, the ES2s don't pass the trees, there's the intense types of values every day, so to be seen, they ensure, this ensures the reflection of programmer's choice, you know, the ES2s, how you want to represent your information, how the system stands, the difference between a Python ES2, Python interpreter, you know, bytecode interpreter language, and a C compound, or Russian compiler language, all drivers are bound, not all binds are drivers, you know, basically, not all, relationships are functions that are not returned, they remain the whole, the bind, you know, relationships are functions, that's what that means, but I didn't put that there, so the relationships are functions, you know, you don't need to map, you don't need to return an output, you know, into a decision, to resolve anything with a DAG, or byte DAG, or bipartite DAG, you know, the bind behaviour, rather than computing, basically, to resolve the system, your burden, you know, in a cycling way, without, you know, continuous IOL, without giving you a method to execute it, you know, it's like a server just observing itself and then fix its own errors, you know, because, you know, it knows how to track its own problems, okay, my maths is low now, okay, all drivers are, all binds are drivers, not drivers are binds, binding refers to attaching functionality to data, you know, you know, drivers are forcible of the binding, like USB interface, or gosilab, polygon models, you know, some attach automatically, some are, some are totally bounded, some drivers behave externally, so basically, anything that drives logic, you know, is predictable, you know, you have to, you know, you can discuss everything like binds, it's just a mapping, so you don't need to execute it, you know, you don't need to execute it, you just have to, let's not just interface it, you know, because it doesn't need to be hardware, you know, it's just, it's just mapped, you know, and gosilab, we don't bind our fear, we bind out of care, like, hands to, you know, and the cosy labs, which are probably, probably gosilab, we do not bind our fear, you know, we don't panic, you know, we bind or to fix our code quickly, we basically, we're not bureaucrats, but these are anti-breaker systems, you know, we're trying to bind out of care, you know, like our hands are doing, you know, we hear the one using the device, you know, it reminds me of my treatment machine, you know, if I try to fix that treatment machine, you know, basically the cosy language, like, you know, I was living on a machine, and I decided not to use it because it wasn't quite safe, you know, then I started this project, because cosy language is a very safe language, you know, it's 100% perfect, you know, you just, there's no error, don't panic anything, and how that works is because you bubble all the errors to where the error came from, you know, you don't propagate any errors, you know, all the errors go upstream to the, like, events, probably, and then, and the observers, they track the errors through the programme with a tag, you know, this that means resolve and don't, you know, and this you want to see, and then you don't, because, you know, it's a, it's a by-path of, it works just like all the angles of the errors, like a ring topology system, you know, the goal of the by-path model is to resolve the errors in the gossip lab, you know, which is developing the gossip lab, which is gochub.com, GOSI labs, it's also gossip languages in our labs, so basically you have ordered this here, and then you get to plan you now, you know, so like, the P2 gossip, the PYG gossip, the system is connected to gossip language, you know, to do something, but they're just for, you know, there is a schema to it, basically, if you want, like, a PHP gossip for some type, like a PHP gossip for a PHP package, you know, but you can use a PHP binding, which you can map, you can use a PHP plug-in, which is essentially of a binding to extend the plug-in for the HD, which is for each case in that PHP system, and then you can get a PHP software domain kit, you can get a PHP platform, different bucket, or SDK, depending on the Windows, Linux, or Macintosh systems, you know, depending on what the things are really about, you know, it's joined as a degradation between all the ecosystems, and everything is deliberately binded, because you need it, you don't even want it, you don't want it there, you know, I mean basically, concur, basically, you know, how Gosilang stuff was, you know, it's about thread-safe, everything, Gosilang is 100% thread-safe, why is it 100% thread-safe? Because, you know, you don't, you don't, you don't map into it, you don't need to observe it, you can see the state of your programme, and the big thing with concurrency, and thread panics, you know, it's because they are synchronous, you have to time them everything, you have to, but you know, you don't, you feel probably everything, that's frustration, you don't even need to, if the source is the error code, you know, you don't, you don't resolve it with the wrong tag, you go the wrong way, and you know, the big thing with thread-safe is 100% thread-safe means, like, you know, there's no risk in concurrency, or parallelism, around two times, there's no attacks, or lapping times, and how I saw this, because literally, you know, you can run this in a tag, you know, in a bad head of a tag, but, you know, you bubble up, you win at the state, you observe the state, yes, because there's no thread involved, even, there's no thread, you know, everything is just a, like, you know, a message passing, you know, basically, if I send some data, or some, it's a message, you know, that message is not, it fails, it's not getting processed, you know, it failed, and you can fix the process before you can do it, because, you know, the data, the stream of information, it's not, it's not blocking thread, it's just locks, there's no panicking, there's no, it's asynchronous locks, you know, it's just time, it's just, you know, the message is coherent enough, and the timing is, it's past, because, you know, there's no, I'm saying there's no thread issues there, because the sleeping machine will have a lot of thread issues because of locks, everything, and try to manage two resources at a time, but it's not easy to unlock one resource and use another resource, you know, you know, it's just time consuming, a lot of resources, so you just crush all that money to do together, you know, basically, how this works is based on the model of the system here, I'm going to give you now, so basically, you have like a, like, you know, like a filmmaker's token, it's like a, you know, filmmaker's memory, which can be nil, new state, it just means Iota state, which can be the default state of that system, when it doesn't evolve, you know, just like, nil means nothing, no memory, no token type, no value, empty, so basically, the nil type is a void pointer, you see, it points to nothing, because it's not into a pointer, you get it? You know, so it's just, it's a waypoint opportunity, the top type, no, it's void, means that void data, no, void data means void token, for focal token, void token, but that token is not present, or some information is present, or token value, token type, void token type is like the type of the data, and the type of the token, and the type of token data, the data, the data, the data, and then that token value, the value of the data, the character, or the string, or the, what they do, but, you know, we, in our model, with the PLP layer, the general type model, it's just, there's no, there's no ambiguity in the structure of the schema.

Now, the general type model is really important here, for Gussie UML. Now, the Gussie UML is very important, because the Gussie UML is like, the UML layer of the architecture, you know, we stay, we listen, we compile both, we don't, we don't, basically, the thing about this, is the Gussie UML isn't like a planning diagram, but, you know, every time I type a Gussie programme, Gussie UML language, you know, I gossip with something, I get to Gussie, like, you know, I, I get tokens to me, but, you know, I get the, the architectural blueprints, you know, what I'm wanting, you know, I get the structure to, or what the, biological system looks like, you know, if I had it, like, in silicon, you know, like, in the architecture, you know, basically, when I type in Gussie UML code, I get the, my goal is to have this system, give me the blueprint, why I type code, it would have come to me then, you know. So, when I export a code, I look at my blueprint and say, oh, yes, yes, this is what I modelled in my, in my framework, this is what I got, unless I can improve, you know, it's really new to do this, but, you know, it's architectural as infrastructure, so you can have the, like, me, unless I'm building the architecture on the evaluation layer, and I'm writing a programme, you know, to call it autocross itself, if I am trying to implement a healing artefact, but, you know, I don't know about the system, it's, it's based on, you know, execution, and, you know, which is just this here, I'm just trying to not take a picture here, because it's, I've got five more minutes before my things stop.

Diagram, the architecture is random access memory. So, we've got, we've got the planning UML diagram with architectural blueprints, the big document is the silicon that maps to the infrastructure. We are via bi-tactical position, and the diagram is so much of vision as a policy.

Now, the policy and the standard application and everything that's implemented is just like a, like, you know, a declarator, a declarator function, a function that maps across all functions, you know, with the functional frameworks, and gives up the computational benefits of functional frameworks. So, you're going to have that in place to ensure that a segmentary is, a segmentation, which is like a segmentary memory using, or from magic memory. It's kind of with the architecture, it's a bi-tactical relationship, like, based on algorithmically, the second three nodes, it's like, you know, the three nodes for them.

So, basically, you've got, like, a phenology of token type, token memory. Look, basically, it's like a structure, basically, it's like a two-bit, like, something with, like, a struck colour. Colour is 255 range, or 256.

If it's an unsigned integer, you know, but, you know, that colour, 0 to 256, once it's 125, it's a range of colours, but that model of colour is going to be used to do some architectural thing, like, you know, if you want to, you know, if you want, like, an RGB, like, to do randomness and kind of like, even where you see, where you get, or where you see, what you mean by direction of pipeline. But, you know, when I type the code, I get the light from there, I get the silicon resolution of the architecture. So, it can be debugged, it can be actually done, you know, if I'm not with the system, you know.

So, basically, they don't see anything. The architecture is very important for our system to remain coherent, because, you know, we are the ones, we are humans, we know computers. Basically, the segment tree is the node that it segmented.

It's a big segment, it's like a segment of memory, a segment of a token, a segment of a value, a data value, using, you know, a segment section, like, you know, a section of memory. So, you can defrag yourself, you can de-segment yourself, and you can go into your, into your tree, which, like, your time, it needs to run, which I used to ensure that the system does what it does in time, you know. So, you can go into your tree node, silver balancing node, you can measure its own state.

27, most parameters, I'll have to stop by here today. But, the goal of this, I'm going to show you the code in the next video, but, you know, that I've not done today. So, we've got the PRP's parse mode, and the infrastructure is networking.

The computational geometry of the system that can evolve, you know, we have NMS, NMS, Sibley language model, and Sibley Amazon Atlas for sleep handling systems. Or, like, basically, you can try to, when it's gone over, after my PhD, when it's gone over in two years, after my PhD, when it's language is complete, I'll try to build my own sleep value machine with it. You know, for myself, my sake, you know, and I'll build, which is, you know, which is not, which is

100% really safe.

And this consciousness, you know, conscientious plasma systems, like, I think that, like, for system people, or even for people who are conscious, you know, I knew there were people who are, who are conscious, and needed the system to remain for them or call them in a sleep to really, their own coherence state of mind, of awareness while they sleep, so they can be cautious, you know, like, you know, put a mirror on for a child, it's not good enough anymore. So, that's what I'm saying, you know. Based on physical architecture, you know, computational work.

So, basically, the goal is for this programme language to remain coherent, okay, now, to nine minutes, okay. Now, this is with this way, but, you know, we've got the one here. We will stay, we'll listen, we'll compile while it's gone, and we've got, that means the gosilang are based on rift policies, but the rift policies are just for a rift ecosystem, so the gosilang is put on rift, so basically, that's like, you know, the rift policy is not introduced or anything.

No burnout, it's like a rift policy file, and now, no burnout, no overclock, and no just rhythm, you know, basically, it's about thinking of the nexus linker, and it's showing that in any link, links everything, which, you know, the accesses are unbloated, you know, there's no burnout in space and time. We have an app, a rift app, promodoro by promodoro, okay. Goal, a goal, a breath, a push, a rest, you know.

Two truck of philosophy, foundational truck, and aspiration truck, you work with two truck of philosophy all the time. Okay, now, guys, my government stuff, I've got a great time. You don't need permission to breathe, need permission to take a break, only to read, okay.

I'm going to collect a couple of videos and I'll finish, thank you guys. This is the medium article, let me stop it now, three, two, one. Three, two, one.

Okay. Okay, it's done, this is done, this is done now. Okay, that's fine.

That's fine, that's fine, that's fine. Okay, it's not done. Okay, this should be fine.