## Node < char > \* starterNode std::vector< Edge</li> char > \* > edges+ Model() + char \* RandomWalk(Markov ::Random::RandomEngine \*randomEngine, int minSetting, int maxSetting, char \*buffer) + void AdjustEdge(const char \*payload, long int occurrence) + bool Import(std::ifstream \*) + bool Import(const char \*filename) + bool Export(std::ofstream \*) + bool Export(const char \*filename) + Node < char > \* StarterNode() + std::vector< Edge< char > \* > \* Edges()+ std::map< char, Node < char > \* > \* Nodes() + void OptimizeEdgeOrder() Markov::API::MarkovPasswords std::ifstream \* datasetFile std::ofstream \* modelSavefile std::ofstream \* outputFile + MarkovPasswords() + MarkovPasswords(const char \*filename) + std::ifstream \* OpenDataset File(const char \*filename) + void Train(const char \*datasetFileName, char delimiter, int threads) + std::ofstream \* Save (const char \*filename) + void Generate(unsigned long int n, const char \*wordlistFileName, int minLen=6, int maxLen=12, int threads=20) + void Buff(const char \*str, double multiplier, bool bDontAdjustSelfLoops =true, bool bDontAdjustExtendedLoops=false) void TrainThread(Markov ::API::Concurrency::ThreadShared ListHandler \*listhandler, char delimiter) · void GenerateThread (std::mutex \*outputLock, unsigned long int n, std ::ofstream \*wordlist, int minLen, int maxLen) Python.Markopy.MarkovModel + def Import(str filename) + def Export(str filename) + def Train(str dataset, str seperator, int threads) + def Generate(int count, str wordlist, int minlen, int maxlen, int threads) Python.Markopy.MarkovPasswordsCLI + model + def init (self, bool add help=True) def generate(self, wordlist) Python.Markopy.MarkopyCLI + args + cli + def init (self, add help=False) + def add arguments(self) + def help(self) + def parse(self) + def init post\_arguments (self) + def parse fail(self)

Markov::Model < char >

std::map< char, Node</li>char > \* > nodes

## Python.CudaMarkopy.CudaMarkopyCLI

+ def init\_post\_arguments(sel)  $\Delta$ 

+ def process(self) + def stub(self) + def evaluate(self,

str filename)

- + args + cli
  - CII
- + None \_\_init\_\_(self) + def help(self)
- + def parse(self) + def parse fail(self)