**Python Introduction**

**AWS Cloud9** provides a cloud-based IDE which can run code from the command line with the format *$python [name].py* with the following benefits:

|  |  |  |
| --- | --- | --- |
| *Start Projects Quickly & Code with only a Web Browser.* | *Code Together in Real-Time.* | *Build Serverless Applications with Ease.* |
| *Live Syntax Checking.* | *Automatic Indentation.* | *Automatic Completion.* |
| *Code Folding.* | *Splitting Panes.* | *Version Control Integration.* |

**AWS Lambda** can take uploaded code to trigger from an event such as a user visiting your website. It only functions from being triggered and only pay for the compute time used, supporting multiple languages & AWS Cloud9 is included within the Lambda interface. It has the following benefits:

|  |  |  |
| --- | --- | --- |
| *No Servers to Manage.* | *Run Code for any App/Backend Service.* | *Code can Automatically Trigger from other AWS Services.* |

**Shell Scripting Commands** can run directly from an OS Command Line, available on any Machine/OS without needing to install software i.e. – *Bash, Zshell, etc.* They can be compared to Python:

|  |  |
| --- | --- |
| **Shell Scripting** | **Python** |
| Powerful Tool for System Administration. | Better for Complicated Data Structures. |
| Requires many lines for certain actions such as creating a HTTP Server compared to Python. | Complete Programming Language with many Libraries & Resources. |

**Python Programming Introduction**

**Automation** refers to any technology removing human interaction from a system/equipment/process. It often uses **Scripts** to automate labour-intensive tasks & streamline workflows.

**Software** can be written with either:

* Text files using Text Editors (i.e. – VSCode, Vi/Vim, Nano, GNU emacs, Notepad++, TextEdit, etc)
* Computer Languages (i.e. – Python, JavaScript, C#, C++) each with their own Grammar/Syntax/Common Uses/Community.

**Integrated Development Environments** tell you incorrect spelling/syntax and unclear phrases, giving you a possible solution. Some IDEs only work with one language. They use either compilers or interpreters

|  |  |
| --- | --- |
| **Compilers** | (*C/C++, Basic, GoLang*) take high-level languages into low-level machine code after changes are made before it runs the code. |
| **Interpreters** | (*Python, Ruby, JavaScript*) do the same process one step at the time while the code is running. |
| **Variables** | Identifiers in code representing a value in memory, the name helps human remember what it means. |
| **Switches** | Included in most Programming Languages to conveniently handle multiple cases of a value. |
| **Execution Paths** | Sequence of Steps a Programmer Performs when it Runs, coming to an either-or/multiple choice(s). |

The **American Standards Association for Information Interchange (ASCII)** is a System Associating Encoding Characters into Computers.

*There are different data types:*

|  |  |
| --- | --- |
| **Primitive Data Type** | Built into Coding Language without Modifications. |
| **Composite Data Type** | Combines Multiple Data Types into Single Unit. i.e. - *A Movie with the form name + year + viewed status.* |

*There are also different collection group types:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Arrays*** | ***Vector*** | ***Lists*** | ***Set*** |
| ***Queue*** | ***Deque (Double-Ended Queue)*** | ***Hashes*** | ***Dictionaries*** |

**Version Control Systems** are Version Tracking Software (i.e. – *Git/GitHub/GNU Arch/Mercurial*) for Code & Documents as you update them. It can be locally done on your PC or through a Dedicated Website. It offers the following advantages:

|  |  |  |
| --- | --- | --- |
| *Ease of Access to Changes.* | *Error Tracking.* | *Security.* |

**GitHub** is a Repository Hosting Service using repositories & the same commands as the terminal.

**Python Basics**

|  |  |
| --- | --- |
| **Identifiers** | Works as a name for entities (i.e. - *Class, Functions, Variables*) to differentiate them from each other. |
| **Functions** | Tell Computers to do a Specific Task, fitted with names and parentheses taking various arguments. |
| **Comments** | Serve as notes to yourself & others, describing the contents of a program & how it works using the # symbol. |

|  |  |
| --- | --- |
| **Mutable Data Types** | **Immutable Data Types** |
| List, Set, Dictionary, Byte Array. | Integer, Float, Complex, String, Tuple, Frozen Sets. |

|  |  |
| --- | --- |
| **Operators** | |
| **Arithmetic Operators** | +, -, \*, /, %, \*\*, // |
| **Comparison Operators** | <, >, <=, >=, ==, != |
| **Assignment Operators** | =, +=, -=, \*=, /=, %=, //=, \*\*=, &=, |=, ^=, >>=, <<= |
| **Logical Operators** | And/Or/Not |
| **Membership Operators** | In/Not In |
| **Identity Operators** | Is, Is Not |

**System Administration**

**SysAdmin** is the management of hardware/software systems which ensures all related services are okay through various means (i.e. – Installing New Hardware/Software, Creating/Managing User Accounts, Maintaining Servers/Databases & Dealing with System Outages).

|  |  |  |
| --- | --- | --- |
| *Increases Efficiency* | *Identify & Resolve Problems before the affect business.* | *Show System Stability and Smoothness.* |

**Modules & Libraries**

**Libraries** are collections of Custom/Built-In Modules that make programming easier by removing the need to rewrite common commands.

**Modules** can be created by others (External Sources), Prepackaged with Python or created by you.

|  |  |  |
| --- | --- | --- |
| **Exception Handling Cases** | | |
| *Trying to read a file which might not exist.* | *Truing to understand an incorrectly saved file.* | *Trying to write to an existing file which is out of space.* |
| *Trying to handle user input which may be invalid.* | *Trying to make a HTTP Request without Internet.* |  |

**OS** is part of the Python Standard Library providing Operating System Functionality, commonly used for File Manipulation, Directory Traversal and Process Management.

|  |  |
| --- | --- |
| **OS Capabilities** | |
| **Getlogin** | **Return name of logged-in user.** |
| **Getgrouplist** | **Return list of Group IDs user belongs to.** |
| **Getenv** | **Return environment variable value passed in.** |
| **Uname** | **Returns information to identify current OS.** |
| **System** | **Run commands in a subshell of the system.** |

**JavaScript Object Notation (JSON)** is a standard file format transmitting data object, it’s language independent and is used to serialize objects turning code intro string transmittable over a channel.

* *Dump & Dumps turn structured data into strings which can be written to a file.*
* *Load & Loads turn strings back into structured data.*
* *Dump & Load work directly with files, Dumps & Loads work with Strings.*

**Debugging & Testing**

**Debugging** identifies defects in the code through finding errors & fixing them. It’s quite difficult so languages have tools to mitigate this.

* The Python Debugger is activated with *Python -m pdb <filename>*.

|  |  |
| --- | --- |
| **Debugging Forms** | |
| **Static Analysis** | Continuously during Development Process, reporting Syntax/Semantic Errors, aided by IDEs (i.e. – *nesting, function calls, code complexity*).  Exact Issue Location Manual Analysis is Time Consuming.  Faster Fix Turnaround Time. Automation False Positives/Negatives.  Later Tests have Less Issues. Automation takes Security for granted. |
| **Dynamic Analysis** | Analyses Running Applications in IDE Debugging mode, writing out values/conditions happening in a running log file.  Finds Issue in Runtime Environment. Automation takes Security for granted.  Analyse Apps without seeing Code. Can’t Guarantee Full Coverage.  Proves/Confirms false Negatives. Can be more difficult Isolating Code Issues.  Can be Used with Every App. |

**Assertions** are conditions that check application values, used by Dynamic Analysis to raise errors when conditions occur and monitor logs.

**Log Monitoring** is done within a text file to keep track of errors in a running program & the last time it was run for later reviews (*seeing what went wrong in specific areas, etc*.) Python uses *import logging*

|  |  |  |
| --- | --- | --- |
| *Where did it occur?* | *What time did it occur?* | *What were the Arguments?* |
| *What is State of Important Resources?* | *Capture all Exceptions + Inner Exceptions.* | *Capture all Traceback Objects such as Stack Traces.* |

**Software Testing**

|  |  |
| --- | --- |
| **Software Testing Tiers** | |
| **Unit Testing** | Smallest & Most Basic Testable part of any software, a few inputs with a single output i.e. – *verifying individual functions of a program*. |
| **Integration Testing** | Tests Interaction between different software parts to identify issues, performed by developers frequently through using dedicated testers. |
| **System Testing** | Complete & Integrated App Tested to see whether it meets specific reqs. |
| **Acceptance Testing** | Formalized Test considering User/Business Needs & whether software is acceptable for delivery to final user. |

**DevOps & Continuous Integration**

**DevOps** is a software engineering culture/practice aiming to unify software development & operations, advocating for automation & monitoring at all steps of software construction. It bridges the gap between traditional IT Software Development & Quality Assurance.

**DevOps Culture** considers the performance of the entire system, creating & amplifying right-to-left feedback loops. They foster continual experimentation, taking risks & learning through repetition & practice.

|  |  |
| --- | --- |
| **Types of Automation & Risks** | |
| **Over-Automation** | Automated steps reduce creativity i.e. – analysing, planning & designing. |
| **Under-Automation** | Avoiding Automation to make sure things are handled correctly, should automate building, testing & deploying. |
| **Bad Automation** | Automated Process doesn’t work well, fixed by revisiting development planning stage. |

|  |  |
| --- | --- |
| **Automation Tools** |  |
| Build Automation | Automatically Compiling Code after changes are made. |
| Test Automation | Logical Tests automatically test Logic after changes to ensure it works. |
| Deployment Automation | Get Code to Usable Format for testing/use. |

**Continuous Integration (CI)** is automation of making code available for teammates, including build automation & quality assurance with the following two purposes.

1. Make sure code works with what’s already been done.
2. Make sure code is readable for who works on it after you.

**Continuous Delivery (CD)** is an extension of CI including test automation for all code submitted ensuring it works in a way that makes sense and as intended. Also ensures at any point a working version of the code can be produced immediately.

**Configuration Management**

**Project Infrastructure** speaks to how a project is organized, Architects organize *Bridge Infrastructure*, Software Developers organize *Code Infrastructure.*

|  |  |
| --- | --- |
| **Tools & Templates** | |
| **For Style** | Some Utilities (*Pylint*) runs to ensure code indented & fix formatting errors. |
| **For Logic** | Other Utilities (*PyTest*) runs test to ensure code changes still reach requirements. |

**Configuration Management** tracks versions of code as developed, enabling devs to work independently on parts then merge changes into the project through tools i.e. – Git. Enables fast rollback to working versions when updates give errors.

Configuration Management works as following:

1. Developers check out code from Repository.

*$git @<examplerepo.org>:<username>/<sourcecode>.git*

1. Developers upload changed code to Repository.

*$git –commit –m “Message about the changes.*

1. Code merged back to Main Project once it passes all tests from Git/IDE Tools (PyCharm).

*$git push*

1. Release Managers Create Distribution of Software based on Repository Contents.
2. Release Managers version the software for future rollbacks in case of errors

|  |  |
| --- | --- |
| **Configuration Management Aspects** | |
| **Accounting** | Sometimes Team Leads/Managers ask for Project Status, the team can check-in & report efforts. |
| **Security** | Repository Access must be granted and logged to help learn who/when/what checked code in/out. |