**Corporate tax fraud detection using Machine Learning**

**Objective**

The primary objective is to create an advanced machine learning system for detecting potential corporate tax fraud. This system aims to automate the identification process, using cutting-edge techniques like boosting algorithms to enhance accuracy. By accurately distinguishing between legitimate and potentially fraudulent tax filings, the goal is to minimize erroneous returns, protect government revenue, and optimize resource utilization. Ultimately, the project aims to contribute to a more secure and efficient income tax assessment process.

**Scope of The Project**

The project's scope involves gathering corporate tax datasets, implementing boosting algorithms for fraud detection, and evaluating their performance. It aims to automate fraud identification, improve accuracy in distinguishing legitimate and fraudulent tax filings, and contribute to a more secure and efficient tax assessment process. Ultimately, the project focuses on leveraging machine learning techniques to detect potential income tax fraud effectively.

**Abstract**

This project focuses on using boosting algorithms within machine learning to identify potential instances of income tax fraud. It involves gathering and utilizing income tax datasets for training and testing. The primary objective is to optimize fraud detection accuracy using boosting algorithms and visualize the critical attributes that aid in fraud identification. Evaluation will rely on key metrics like accuracy, sensitivity, specificity, and precision. Ultimately, the aim is to automate fraud detection, accurately differentiate between legitimate and potentially fraudulent tax filings, minimize errors, and enhance the efficiency and security of income tax assessments.

**Existing System**

The current approach to detecting income tax fraud involves methods such as data normalization, cluster analysis, and neural networks. These techniques aim to standardize data, identify patterns, and train models for fraud detection. However, despite these efforts, the accuracy of detecting fraudulent tax filings remains around 50%. This signifies limitations in accurately identifying potential instances of income tax fraud using the existing methodologies based on normalization, clustering, and neural network training.

**Disadvantage:**

* Limited Accuracy
* Failure to Address Emerging Fraud Patterns
* Resource Intensiveness
* Limited Improvement in Cost Measures

**Proposed System**

The proposed system for corporate income tax fraud detection aims to address limitations by implementing boosting algorithms, emphasizing financial attributes like income sources and expenses. It seeks to enhance accuracy by automating the detection process, optimizing performance metrics, and adapting to emerging fraud patterns. Ultimately, the system aims to significantly improve the identification of potential fraudulent tax filings, ensuring a more robust and efficient detection mechanism.

**Advantage:**

* Boosting algorithm ranks the importance of variables in a regression or classification problem in a natural way can be done by Random Forest.
* Improved Accuracy
* Adaptability to New Fraud Patterns
* Enhanced Efficiency

**System Architecture**

Test data

dataset

Machine learning model

Feature extraction

pre-processing

Classifier Section

Result

Performance analysis

**Modules**

**DATA COLLECTION**

**DATA PRE-PROCESSING**

**FEATURE EXTRACTION**

**EVALUATION MODEL**

**DATA COLLECTION**

Data collection is a process in which information is gathered from many sources which is later used to develop the machine learning models. The data should be stored in a way that makes sense for problem. In this step the data set is converted into the understandable format which can be fed into machine learning models.

Data used in this paper is a set of weather data with 25 features . This step is concerned with selecting the subset of all available data that you will be working with. ML and nueral networks problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called labelled data.

**DATA PRE-PROCESSING**

Organize your selected data by formatting, cleaning and sampling from it.

Three common data pre-processing steps are:

Formatting: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

Cleaning: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

Sampling: There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

**FEATURE EXTRATION**

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data. The chosen classifiers were Random forest. These algorithms are very popular in text classification tasks.

**EVALUATION MODEL**

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid over fitting, both methods use a test set (not seen by the model) to evaluate model performance.

Performance of each classification model is estimated base on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs.

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

**Data Flow Diagrams**

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.

It is a graphical tool, useful for communicating with users ,managers and other personnel. it is useful for analyzing existing as well as proposed system.

It provides an overview of

• What data is system processes.

• What transformation are performed.

• What data are stored.

• What results are produced , etc.

Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

The Data Flow Diagram has 4 components:

Process Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence

Data Flow Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.

Warehouse The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updating.

Terminator The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

DATA FLOW DIAGRAM

LEVEL 0

LEVEL 1

LEVEL 2

UML Diagram - Use Case Diagram:

UML Diagrams

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

● actors

● business processes

● (logical) components

● activities

● programming language statements

● database schemas, and

● Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

Use case diagram:

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously putting effort to make a truly industry standard.

UML stands for Unified Modeling Language.

UML is a pictorial language used to make software blue prints

**Data Flow Diagrams**

**DFD** is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.

It is a graphical tool, useful for communicating with users ,managers and other personnel. it is useful for analyzing existing as well as proposed system.

It provides an overview of

* What data is system processes.
* What transformation are performed.
* What data are stored.
* What results are produced , etc.

Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

**The Data Flow Diagram has 4 components:**

**Process** Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence

**Data Flow** Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.

**Warehouse** The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updating.

**Terminator** The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

**DATA FLOW DIAGRAM**

**LEVEL 0**

LEVEL 1

LEVEL 2

**UML Diagram - Use Case Diagram:**

**UML Diagrams**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

* actors
* business processes
* (logical) components
* activities
* programming language statements
* database schemas, and
* Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

**Use case diagram:**

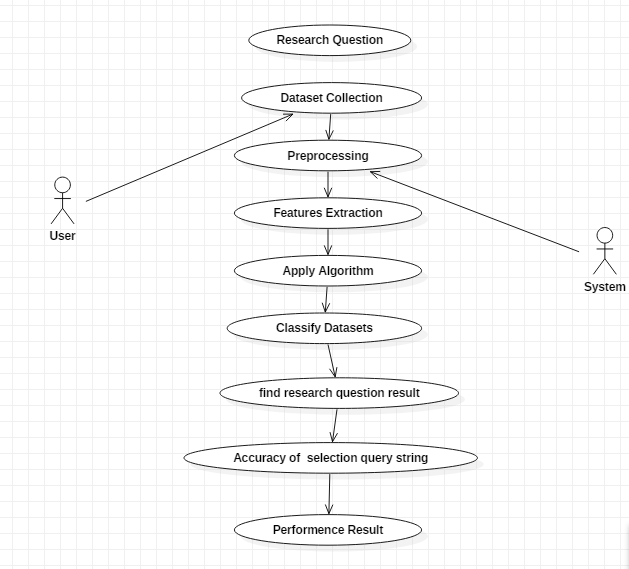
UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously putting effort to make a truly industry standard.

UML stands for Unified Modeling Language.

UML is a pictorial language used to make software blue prints



**Class Diagram:**

**Class diagram**

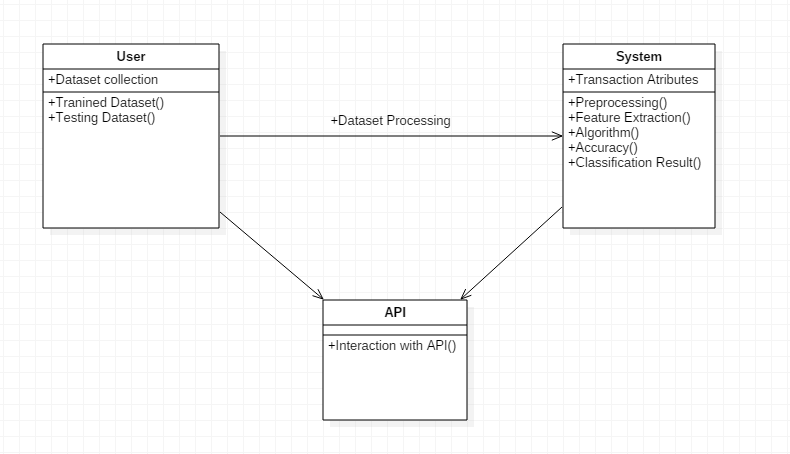
The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.

The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.

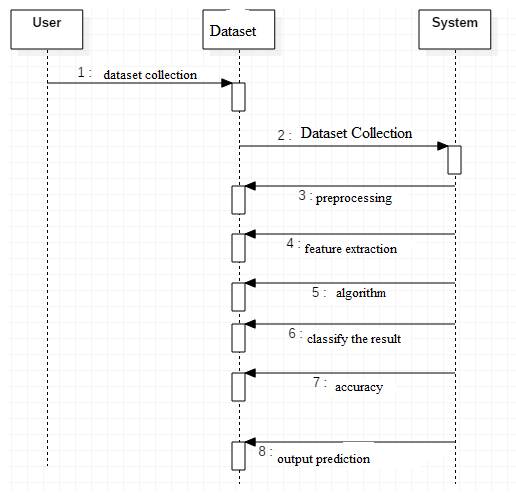
The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.



**Sequence Diagram:**

**Sequence Diagram:**

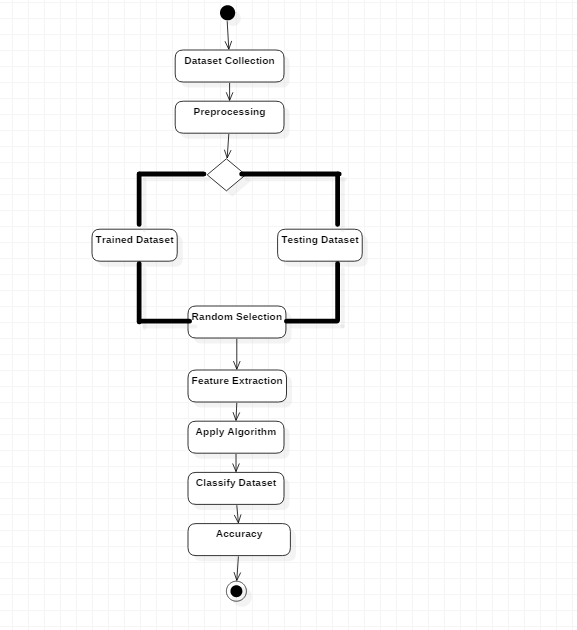
Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.



**Activity Diagram:**

**Activity Diagrams-:**

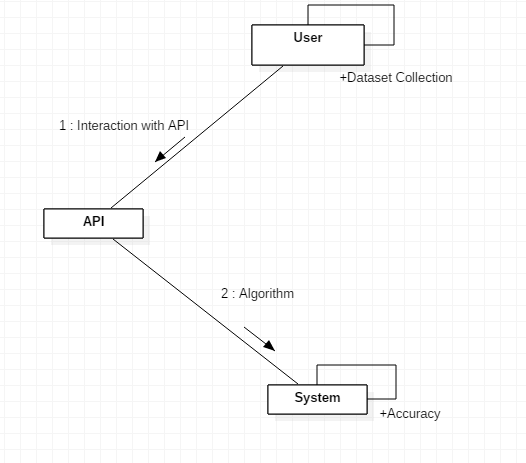
Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**Colloboration Diagram:**

**COLLABORATION DIAGRAM**

Another type of interaction diagram is the collaboration diagram. A collaboration diagram represents a collaboration, which is a set of objects related in a particular context, and interaction, which is a set of messages exchange among the objects within the collaboration to achieve a desired outcome.



**Software and Hardware Requirements:**

**Hardware:**

* OS – Windows 7, 8 and 10 (32 and 64 bit)
* RAM – 4GB

**Software:**

* Python
* Anaconda

**Literature survey:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| s.no | **Year** | **Title** | **author** | **Algorithm** | **Disadvantages** |
| 1 | **2019** | **Abnormal Group-Based Joint**  **Medical Fraud Detection** | **Chenfei Sun;**  **Zhongmin Yan;**  **Qingzhong Li;**  **Yongqing Zheng;**  **Xudong Lu;**  **Lizhen Cui** | **linear regression** | **It is often quite prone to noise and overfitting** |
| 2 | **2023** | **Generative Adversarial Networks-Based Novel Approach for Fraud Detection for the European Cardholders 2013 Dataset** | **Fahdah A. Almarshad;**  **Ghada Abdalaziz Gashgari;**  **Abdullah I. A. Alzahrani** | **K-means clustering** | **It requires very large amount of data in order to perform better than other techniques** |
| 3 | **2020** | **A Systematic Literature Review of**  **Fraud Detection Metrics in**  **Business Processes** | **Badr Omair;**  **Ahmad Alturki** | **Random forest (RF)** | **Complexity: Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs** |
| 4 | **2018** | **CoDetect: Financial Fraud Detection With Anomaly Feature Detection** | **Dongxu Huang;**  **Dejun Mu;**  **Libin Yang;**  **Xiaoyan Cai** | **Naϊve Bayes** | **The assumption of linearity between dependent and independent variables** |
| 5 | **2020** | **Taxonomy of Fraud Detection Metrics for Business Processes** | **Badr Omair;**  **Ahmad Alturki** | **regression model** | **Require high memory – need to store all of the training data** |

Domain overview:

PYTHON OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

• Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

• Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

• Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

• Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

Easy-to-read: Python code is more clearly defined and visible to the eyes.

Easy-to-maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

• IT supports functional and structured programming methods as well as OOP.

• It can be used as a scripting language or can be compiled to byte-code for building large applications.

• It provides very high-level dynamic data types and supports dynamic type checking.

• IT supports automatic garbage collection.

• It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

LANGUAGE USED:

1.What is Python:

Python is an object-oriented, high level language, interpreted, dynamic and multipurpose programming language.

Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas.

Python supports multiple programming pattern, including object oriented programming, imperative and functional programming or procedural styles.

Python is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to declare an integer value in a variable.

Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

2. Python Features

1) Easy to Use:

Python is easy to very easy to use and high level language. Thus it is programmer-friendly language.

2) Expressive Language:

Python language is more expressive. The sense of expressive is the code is easily understandable.

3) Interpreted Language:

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform language:

Python can run equally on different platforms such as Windows, Linux, Unix , Macintosh etc. Thus, Python is a portable language.

5) Free and Open Source:

Python language is freely available(www.python.org).The source-code is also available. Therefore it is open source.

6) Object-Oriented language:

Python supports object oriented language. Concept of classes and objects comes into existence.

7) Extensible:

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in your python code.

8) Large Standard Library:

Python has a large and broad library.

9) GUI Programming:

Graphical user interfaces can be developed using Python.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10) Integrated:

It can be easily integrated with languages like C, C++, JAVA etc.

3. Python History

• Python laid its foundation in the late 1980s.

• The implementation of Python was started in the December 1989 by Guido Van Rossum at CWI in Netherland.

• ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.

• Python is influenced by programming languages like:

• ABC language.

• Modula-3

4. Python Version

Python programming language is being updated regularly with new features and support. There are a lot of updation in python versions, started from 1994 to current date.

A list of python versions with its released date is given below.

Python Version Released Date

Python 1.0 January 1994

Python 1.5 December 31, 1997

Python 1.6 September 5, 2000

Python 2.0 October 16, 2000

Python 2.1 April 17, 2001

Python 2.2 December 21, 2001

Python 2.3 July 29, 2003

Python 2.4 November 30, 2004

Python 2.5 September 19, 2006

Python 2.6 October 1, 2008

Python 2.7 July 3, 2010

Python 3.0 December 3, 2008

Python 3.1 June 27, 2009

Python 3.2 February 20, 2011

Python 3.3 September 29, 2012

5. Python Applications

Python as a whole can be used in any sphere of development.

Let us see what are the major regions where Python proves to be handy.

1) Console Based Application

Python can be used to develop console based applications. For example: IPython.

2) Audio or Video based Applications

Python proves handy in multimedia section. Some of real applications are: TimPlayer, cplay etc.

3) 3D CAD Applications

Fandango is a real application which provides full features of CAD.

4) Web Applications

Python can also be used to develop web based application. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

5) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

6) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

6. Python Example

Python code is simple and easy to run. Here is a simple Python code that will print "Welcome to Python".

A simple python example is given below.

1. >>> a="Welcome To Python"

2. >>> print a

3. Welcome To Python

4. >>>

Explanation:

• Here we are using IDLE to write the Python code. Detail explanation to run code is given in Execute Python section.

• A variable is defined named "a" which holds "Welcome To Python".

• "print" statement is used to print the content. Therefore "print a" statement will print the content of the variable. Therefore, the output "Welcome To Python" is produced.

Python 3.4 Example

In python 3.4 version, you need to add parenthesis () in a string code to print it.

1. >>> a=("Welcome To Python Example")

2. >>> print a

3. Welcome To Python Example

4. >>>

7. How to execute python

There are three different ways of working in Python:

1) Interactive Mode:

You can enter python in the command prompt and start working with Python.

Press Enter key and the Command Prompt will appear like:

Now you can execute your Python commands.

Eg:

2) Script Mode:

Using Script Mode , you can write your Python code in a separate file using any editor of your Operating System.

Save it by .py extension.

Now open Command prompt and execute it by :

NOTE: Path in the command prompt should be where you have saved your file. In the above case file should be saved at desktop.

3) Using IDE: (Integrated Development Environment)

You can execute your Python code using a Graphical User Interface (GUI).

All you need to do is:

Click on Start button -> All Programs -> Python -> IDLE(Python GUI)

You can use both Interactive as well as Script mode in IDE.

1) Using Interactive mode:

Execute your Python code on the Python prompt and it will display result simultaneously.

2) Using Script Mode:

i) Click on Start button -> All Programs -> Python -> IDLE(Python GUI)

ii) Python Shell will be opened. Now click on File -> New Window.

A new Editor will be opened . Write your Python code here.

Click on file -> save as

Run then code by clicking on Run in the Menu bar.

Run -> Run Module

Result will be displayed on a new Python shell as:

8. Python Variables

Variable is a name of the memory location where data is stored. Once a variable is stored that means a space is allocated in memory.

Assigning values to Variable:

We need not to declare explicitly variable in Python. When we assign any value to the variable that variable is declared automatically.

The assignment is done using the equal (=) operator.

Eg:

Output:

1. >>>

2. 10

3. ravi

4. 20000.67

5. >>>

Multiple Assignment:

Multiple assignment can be done in Python at a time.

There are two ways to assign values in Python:

1. Assigning single value to multiple variables:

Eg:

1. x=y=z=50

2. print x

3. print y

4. print z

Output:

1. >>>

2. 50

3. 50

4. 50

5. >>>

2.Assigning multiple values to multiple variables:

Eg:

1. a,b,c=5,10,15

2. print a

3. print b

4. print c

Output:

1. >>>

2. 5

3. 10

4. 15

5. >>>

The values will be assigned in the order in which variables appears.

Basic Fundamentals:

This section contains the basic fundamentals of Python like :

i)Tokens and their types.

ii) Comments

a)Tokens:

• Tokens can be defined as a punctuator mark, reserved words and each individual word in a statement.

• Token is the smallest unit inside the given program.

There are following tokens in Python:

• Keywords.

• Identifiers.

• Literals.

• Operators.

Tuples:

• Tuple is another form of collection where different type of data can be stored.

• It is similar to list where data is separated by commas. Only the difference is that list uses square bracket and tuple uses parenthesis.

• Tuples are enclosed in parenthesis and cannot be changed.

Eg:

1.>>> tuple=('rahul',100,60.4,'deepak')

2.>>> tuple1=('sanjay',10)

3.>>> tuple

4.('rahul', 100, 60.4, 'deepak')

5.>>> tuple[2:]

6.(60.4, 'deepak')

7.>>> tuple1[0]

8.'sanjay'

9.>>> tuple+tuple1

10.('rahul', 100, 60.4, 'deepak', 'sanjay', 10)

11.>>>

Dictionary:

•Dictionary is a collection which works on a key-value pair.

•It works like an associated array where no two keys can be same.

•Dictionaries are enclosed by curly braces ({}) and values can be retrieved by square bracket([]).Eg:

1.>>> dictionary={'name':'charlie','id':100,'dept':'it'}

2.>>> dictionary

3.{'dept': 'it', 'name': 'charlie', 'id': 100}

4.>>> dictionary.keys()

5.['dept', 'name', 'id']

6.>>> dictionary.values()

7.['it', 'charlie', 100]

8.>>>

9.Python Keywords

10.

True False None And as

Asset Def class Continue break

Else Finally elif Del except

Global For if From import

Raise Try or Return pass

Nonlocal In not Is lambda

11. Keywords are special reserved words which convey a special meaning to the compiler/interpreter. Each keyword have a special meaning and a specific operation. List of Keywords used in Python are:

10. Identifiers

Identifiers are the names given to the fundamental building blocks in a program.

These can be variables ,class ,object ,functions , lists , dictionaries etc.

There are certain rules defined for naming i.e., Identifiers.

I. An identifier is a long sequence of characters and numbers.

II.No special character except underscore ( \_ ) can be used as an identifier.

III.Keyword should not be used as an identifier name.

IV.Python is case sensitive. So using case is significant.

V.First character of an identifier can be character, underscore ( \_ ) but not digit.

**Conclusion:**

In summary, income tax fraud detection is essential for maintaining financial system integrity. Advanced technologies, such as machine learning, enhance detection capabilities, but ongoing evolution is necessary to combat sophisticated fraud tactics. Collaboration among tax authorities, law enforcement, and financial institutions is crucial for effective prevention, ensuring fairness and transparency in tax systems.

**Reference:**

1.Zhao Yue, "Taxation analysis of university individual tax based on the new individual tax law [J]", Taxation, vol. 000, no. 007, pp. 33-33, 2019.

2.Li Benxuan, "Discussion on personal income tax planning under the new tax law[J]", Chief Accountant of China, vol. 189, no. 04, pp. 76-79, 2019.

3.Deng Yue and Zhang Wei, "Personal income tax planning of college teachers under the new tax law[J]", Nanguo Expo, vol. 094, no. 05, pp. 72-72, 2019.

4.Hou Meng, "Analysis of the personal income tax of administrative institutions based on the new tax law [J]", Taxation, vol. 15, no. 7, pp.

5.Xue Yuexiang, "Research on Personal Income Tax Tax Planning Based on the New IIT Policy [J]", Taxation, vol. 000, no. 034, pp. 35-36, 2019.

6.Su Qiyan, "Personal income tax analysis of public hospitals based on the new tax law".

7.Li Yilian, Mi Wenhong and Ji Yunrui, "Personal income tax planning for university faculty and staff based on the new tax law ——Taking K universities as an example [J]", Economic Research Guide, no. 21, 2020.

8.Qin Xiaobo and Lu Jingbo, "Research on Personal Income Tax Planning under the New Tax Law [J]", Business Intelligence, no. 52, pp. 338-338, 2012.

9.Zhao Danmei, "Research on year-end bonus tax planning based on the background of new personal income tax [J]", Finance and Accounting Study, vol. 000, no. 011, pp. 30-31, 2020.

10.Dong Weifeng, "Research on my country's Personal Income Tax from the Perspective of Equity [D]", Shanghai Normal University, 2012.