University of South Florida: MUMA College of Business

Syllabus for ISM4641: Python for Business

AUTHOR
Dr. Smith

AFFILIATION
University of South Florida

Course Introduction

Welcome to **ISM4641: Python for Business**. I am Dr. Smith, and I am delighted to guide you through this asynchronous online course designed to bridge the gap between fundamental Python programming and practical business applications. Over the next 10 modules (plus an additional 2 optional), you will acquire essential Python skills, learn to manipulate and analyze data, and apply these techniques to solve real-world business challenges. This course is tailored for business students with no prior programming experience and emphasizes hands-on exercises and interactive learning. Each week, you will complete a quiz and an assignment that contribute to your final grade. Due to the course being during the summer, the final two modules are optional and will not affect your final grade. However, they will provide you with additional knowledge and skills that can be beneficial in your future career.

Instructor and Contact Information

- **Professor:** Dr. Smith
  - **Email:** smith515@usf.edu
  - **Office Hours:** Virtual - by appointment
- **Teaching Assistant:** - Name: Prince Praveen
  - Email: princepraveen@usf.edu

Course Objectives

This course is designed to:

- Introduce the fundamentals of Python programming with a focus on business applications.
- Develop your skills in data manipulation, analysis, and visualization using industry-standard libraries.
- Enhance your problem-solving abilities by integrating programming techniques with business analytics concepts.
- Equip you with the capability to design and implement modular, reusable code for automating business tasks and generating reports.

Overall Course Learning Outcomes

By the end of this course, you will be able to:

- **Grasp Fundamental Programming Concepts:** Understand and apply basic Python constructs such as variables, data types, control structures, and functions to solve common business problems.
- **Manipulate and Analyze Data:** Utilize Python's built-in data structures and the Pandas library for effective data cleaning, transformation, and analysis.
- **Develop Modular Business Applications:** Write modular and reusable code that automates tasks like financial calculations, inventory processing, and report generation.
- **Handle Files and Visualize Data:** Read from and write to common file formats (e.g., CSV) and create compelling visualizations using Matplotlib to communicate business insights.
- **Apply Business Analytics Concepts:** Integrate programming with foundational business analytics techniques to interpret data and support strategic decision-making.

Prerequisites

- **Technical Requirements:** Basic computer skills and familiarity with

operating a laptop (MacOS or Windows). No prior programming experience is required, though familiarity with spreadsheets is beneficial.
- **Academic Readiness:** A keen interest in learning programming and its applications in the business world is highly recommended.

Course Description and Required Resources

Course Description: ISM2411: Python for Business is a 9-week asynchronous summer course that introduces Python programming with an emphasis on its application in solving business problems. The course covers Python basics, data collections, file handling, data analysis with Pandas, and data visualization with Matplotlib. Practical exercises and real-world examples ensure that you not only learn theoretical concepts but also gain hands-on experience through interactive exercises and assignments. Each module includes a quiz and an assignment to reinforce learning and assess understanding.

Course Pace and Format: – This is an asynchronous online course, meaning you can complete the modules at your own pace within the specified deadlines. All modules will be released during the first week of the course. **Each module will have deliverables** (i.e. quizzes, assignments, discussion forum postings), and each will have dates you must complete the deliverable by. With such flexibility, it is important to stay on track and manage your time effectively. The course is designed to be completed in 9 weeks, with each module taking less than one week to complete. However, you can work ahead if you prefer.

TextBook: - No textbook is required for this course. All course materials will be provided online through the course website.

Software: - Thonny IDE for coding in Python (thonny.org) This is a free IDE (integrated development environment) that is easy to use and suitable for beginners. It is recommended for this course as it provides a simple interface for writing and running Python code. You can download it from the official website.

Subscription: - We will use CodeGrade for assignments. You will need

to create an account and subscribe to the service. The subscription fee is $10 for the entire course. A link and further details will be provided during the first week of class. You will need to have this subscripption to submit your assignment for Module02.

Hardware: - A personal laptop (MacOS or Windows) with at least 8 GB RAM.

Online Platforms: - Canvas for course materials, quizzes, assignments, and announcements.

Course Policies

Attendance Policy

- **First Day Attendance:** As this is an asynchronous online course, you must complete the first-day attendance quiz within six days of the course start date. Failure to do so may result in removal from the course per university policy.

Grading Scale

Score	Grade
>= 97	A+
>= 94	A
>= 90	A-
>= 87	B+
>= 84	B
>= 80	B-
>= 77	C+
>= 74	C

>= 70	C-
>= 67	D+
>= 64	D
>= 60	D-
Below 60	F

Evaluation Components

Component	Weight
Quizzes	45%
Discussion Forums	10%
Assignments	45%

Quizzes: Each module will have a quiz consisting of 5 multiple-choice questions. Students have up to 3 attempts for each quiz. A new set of questions related to the module's material will be generated for each attempt. The highest score out of the 5 attempts will be recorded.

Discussion Forums: Each module will have a discussion forum where students respond to a reflective question, share ideas, and engage in discussions. Students are encouraged to participate actively and to respond to their peers' posts.

Assignments: Each module will have an assignment that contributes to the final grade.

Late Submission Policy

- Late quizzes will not be accepted. Assignments submitted late will incur a 25% deduction per day.

Incomplete Grade Policy

- An “I” grade may be awarded with demonstration of significant completion of the course work, prior arrangements, valid reasons, and a clear plan for completion. If unresolved by the end of the following semester, the “I” converts to an “IF.”

Communication Policies

Email

- Primary communication is via Canvas email. Expect responses within approximately 24 hours during weekdays.

Canvas

- Canvas serves as the hub for distributing course materials, quizzes, and assignments. For technical issues, contact USF IT support.

HonorLock

- HonorLock is used for quizzes and exams. Ensure your webcam and microphone are functional, and complete the HonorTest quiz as instructed.

Academic Integrity

- Academic integrity is essential. All submitted work must be your own. Any form of academic dishonesty will result in disciplinary action.

Course Schedule

The course is organized into 12 modules. Each module focuses on a key aspect of Python programming for business applications. Each module includes a quiz and an assignment.

Module 1: Introduction to Python and Its Business Applications

- **Content Summary:** Overview of Python, its role in business,

comparison with other tools, Thonny IDE setup, and basic programming concepts like comments, print, variables, and simple debugging.

- **Learning Outcomes:**
  - Explain Python's evolution, advantages, and its pivotal role in modern business applications like data analysis, automation, and integration.
  - Install, configure, and navigate the Thonny IDE, identifying and using key components such as the editor, shell, and variable inspector.
  - Write, save, and execute basic Python scripts incorporating fundamental programming concepts.
  - Understand and apply fundamental Python concepts including variables, comments, basic arithmetic operations, and the `print()` function for displaying output.
  - Utilize basic debugging techniques in Thonny, such as stepping through code line-by-line and inspecting the values of variables during execution.

Module 2: Python Fundamentals – Variables, Data Types, and Arithmetic Operations

- **Content Summary:** Covers variables, core data types (integers, floats, strings, booleans), type conversion, arithmetic operations, order of operations, and an introduction to the `math` library with business examples.
- **Learning Outcomes:**
  - Declare, assign, and manipulate variables using appropriate Python syntax and naming conventions (snake_case).
  - Identify and differentiate between fundamental Python data types (int, float, str, bool) and select the appropriate type for various business data scenarios.
  - Perform arithmetic calculations accurately using Python's range of operators (+, -, *, /, **, %, //).
  - Apply the correct order of operations (PEMDAS/BODMAS) when evaluating complex expressions and use parentheses to ensure clarity and correctness.
  - Convert data between different types (e.g., string to integer, integer to float) using Python's built-in functions and recognize potential conversion errors.

- Import and utilize basic functions from the Python `math` library (e.g., `pow`, `ceil`, `exp`) to perform more advanced mathematical and financial calculations.
- Implement fundamental coding best practices, such as using descriptive variable names and adding comments, to create understandable and maintainable code.

Module 3: Working with Text – Strings, Input, and Output

- **Content Summary:** Focuses on string fundamentals including immutability, indexing, slicing, basic operations, string methods for processing, user input with `input()`, input handling/validation, and output formatting with f-strings.
- **Learning Outcomes:**
  - Access specific characters (indexing) and subsequences (slicing) within strings using correct Python syntax.
  - Explain the concept of string immutability and correctly predict the outcome of operations that appear to modify strings (i.e., they return new strings).
  - Apply common string methods (e.g., `.strip()`, `.lower()`, `.upper()`, `.replace()`, `.split()`, `.find()`, `.join()`) to effectively clean, parse, transform, and standardize textual data for business applications.
  - Perform string concatenation (`+`) and repetition (`*`), and determine the length of a string using `len()`.
  - Obtain textual input from a user via the console using the `input()` function with clear, descriptive prompts.
  - Recognize that `input()` returns strings and reliably convert user input to numeric types (`int`, `float`) when needed for calculations, handling potential `ValueError` exceptions using `try-except` blocks.
  - Employ f-strings (`f"..."`) with various format specifiers (e.g., `:.2f`, `:,`, `:.n%`, alignment) to produce clear, professional, and well-formatted output combining text and variable values.
  - Write simple, interactive programs that integrate user input, string manipulation, data type conversion, basic calculations, error handling (for invalid input), and formatted output to solve basic business problems.

Module 4: Organizing Data – Lists and Tuples

- **Content Summary:** Introduces collections, focusing on lists (ordered, mutable) and tuples (ordered, immutable). Covers creation, accessing elements (indexing, slicing), modification methods for lists, tuple unpacking, and choosing between lists and tuples.
- **Learning Outcomes:**
  - Recognize scenarios where grouping data into collections (lists or tuples) is more effective than using individual variables.
  - Create lists using `[]`, access elements using indexing and slicing, and modify lists effectively using index assignment and methods like `.append()`, `.insert()`, `.pop()`, `.remove()`, `.sort()`, and `.reverse()`, understanding their in-place nature.
  - Create tuples using `()` or commas (including single-item tuples), access elements using indexing and slicing.
  - Explain the concept and benefits of tuple immutability (`TypeError` on modification attempts).
  - Apply standard sequence operations (`len()`, `+`, `*`, `in`, `not in`) correctly to both lists and tuples.
  - Utilize tuple unpacking to assign elements to individual variables efficiently.
  - Distinguish clearly between mutable lists and immutable tuples and make informed decisions on when to use each based on data requirements (changeability vs. fixedness).
  - Identify and avoid common pitfalls associated with list and tuple manipulation (e.g., `IndexError`, `TypeError`, `ValueError`, misunderstanding in-place methods).

Module 5: Data Lookup with Dictionaries & Basic Aggregation

- **Content Summary:** Introduces dictionaries for key-value mapping and efficient data lookup. Covers dictionary creation, operations (accessing, adding, modifying, removing entries), dictionary views, and basic data aggregation functions (`len()`, `sum()`, `min()`, `max()`) applied to collections.
- **Learning Outcomes:**
  - Explain scenarios where dictionary key-value lookup is more

advantageous than list/tuple index-based access.
    - Create dictionaries using `{}` syntax with appropriate unique, immutable keys (like strings or numbers) and various value types.
    - Retrieve values from dictionaries using key lookup (`my_dict[key]`) and anticipate `KeyError` if a key might be missing.
    - Add new key-value pairs and update existing values in dictionaries using assignment (`my_dict[key] = value`), demonstrating dictionary mutability.
    - Remove specific key-value pairs using `.pop(key)` and retrieve the removed value.
    - Check for the presence or absence of keys within a dictionary using the `in` and `not in` operators.
    - Determine the size (number of key-value pairs) of a dictionary using `len()`.
    - Obtain dictionary keys, values, or items using `.keys()`, `.values()`, and `.items()`, and convert these views to lists when needed.
    - Apply built-in aggregation functions (`len()`, `sum()`, `min()`, `max()`) correctly to lists, tuples, and appropriate dictionary components (e.g., `list(dict.values())`) to compute basic summary statistics.
    - Model simple key-based business data (e.g., product information, configurations) using dictionaries.

Module 6: Control Structures – Conditional Logic and Branching

- **Content Summary:** Focuses on implementing decision-making using `if`, `if-else`, and `if-elif-else` structures, comparison and logical operators for conditions, nested conditionals, and a brief introduction to loop constructs (`for`, `while`) and control statements (`break`, `continue`) for repetition, applied to business contexts.
- **Learning Outcomes:**
    - Implement decision-making logic in Python programs using `if`, `elif`, and `else` statements based on evaluated conditions.
    - Construct boolean expressions for conditional statements using comparison (`==`, `>`, `<`, etc.) and logical (`and`, `or`, `not`)

operators.
    - Write `for` loops to iterate effectively over sequences (e.g., lists, tuples, `range()` outputs) and `while` loops to perform repetitions based on runtime conditions.
    - Control loop execution flow strategically using `break` to terminate loops early and `continue` to skip iterations based on specific criteria.
    - Apply combinations of conditional statements and loops to automate processes and implement data-driven logic for common business scenarios (e.g., calculating tiered rates, processing records based on status, simulating changes over time).

Module 7: Control Structures – Looping and Repetition

- **Content Summary:** Details loop structures (`for` and `while`) for automating repetitive tasks, including definite iteration with `for` loops (and `range()`) and indefinite iteration with `while` loops. Covers loop control (`break` , `continue`), nested loops, and integrating loops with conditional statements for practical business problem-solving.
- **Learning Outcomes:**
    - Implement `for` loops to effectively iterate over various sequences (including lists, strings, and `range()` objects).
    - Implement `while` loops to execute code blocks repeatedly based on the evaluation of a boolean condition, ensuring proper termination.
    - Identify and prevent potential infinite loops in `while` constructs.
    - Control the flow of loop execution using `break` to exit early and `continue` to skip iterations.
    - Construct nested loops to handle iterations involving multiple dimensions or combinations of data.
    - Combine loops with conditional statements (`if` , `elif` , `else`) to process items within a collection differently based on specific criteria.
    - Apply loop structures to solve practical business problems involving repetitive calculations, data processing, and simulations.

Module 8: Functions and Modular Programming

- **Content Summary:** Explains the rationale for functions (DRY principle, organization, reusability), defining functions (`def` , docstrings), calling functions, using parameters/arguments (positional, keyword, default), returning values, variable scope (local vs. global), lambda functions, and modular design principles.
- **Learning Outcomes:**
  - Explain the benefits of using functions for code organization, reusability (DRY), readability, and maintainability.
  - Define user-defined functions in Python using the `def` keyword, including parameters and descriptive docstrings (`"""..."""`).
  - Call functions correctly, passing data using positional and keyword arguments.
  - Define functions with default parameter values to create optional arguments.
  - Implement functions that return single or multiple values using the `return` statement and capture these results in variables.
  - Clearly differentiate between the purpose and effect of `print()` and `return` within functions.
  - Describe the difference between local variables (defined within functions) and global variables (defined outside) and predict variable accessibility based on scope.
  - Define and use simple anonymous functions for single-expression tasks using the `lambda` keyword.
  - Apply modular programming principles by breaking down problems into smaller, well-defined, reusable functions.
  - Develop reusable functions that encapsulate common operations applicable to business logic.

Module 9: Introduction to Data Analysis with Pandas

- **Content Summary:** Introduces the Pandas library for data manipulation and analysis, focusing on its core data structures: Series (1D labeled array) and DataFrames (2D labeled table). Covers creation, inspection, data selection/filtering (including boolean indexing), basic descriptive statistics, and identifying missing data.

- **Learning Outcomes:**
  - Explain the advantages of using the Pandas library for handling and analyzing tabular data in business contexts compared to standard Python types.
  - Import the Pandas library using the conventional alias `pd`.
  - Create Pandas Series and DataFrames from Python dictionaries and lists.
  - Load data into a Pandas DataFrame from a CSV file using `pd.read_csv()`.
  - Inspect and understand the structure, dimensions, data types, and content of a DataFrame using attributes and methods like `.shape`, `.info()`, `.dtypes`, `.columns`, `.head()`, and `.tail()`.
  - Select specific columns, rows (by label using `.loc` and position using `.iloc`), and subsets of data from a DataFrame using various indexing techniques.
  - Filter DataFrame rows based on single or multiple conditions using boolean indexing.
  - Calculate basic descriptive statistics (e.g., count, mean, sum, min, max) for DataFrame columns or Series using methods like `.describe()`, `.sum()`, `.mean()`, etc.
  - Identify the presence and count of missing values (`NaN`) within a DataFrame using `.isnull().sum()`.

Module 10: Data Manipulation and Cleaning with Pandas

- **Content Summary:** Focuses on data wrangling techniques, including handling missing data (dropping or filling `NaN` values), correcting data types (`.astype()`, `pd.to_numeric()`, `pd.to_datetime()`), modifying DataFrame structure (adding, removing, renaming columns), applying custom transformations (`.apply()`, `.map()`), string manipulation with the `.str` accessor, sorting data, and managing duplicates.
- **Learning Outcomes:**
  - Implement strategies for handling missing data (`NaN`) in Pandas DataFrames using `.dropna()` and `.fillna()`.
  - Identify incorrect data types in DataFrame columns and convert them to appropriate types (numeric, string, datetime, category) using `.astype()`, `pd.to_numeric()`, and

`pd.to_datetime()`, including handling potential errors.
    - Add new calculated columns, remove unnecessary columns using `.drop()`, and rename columns for improved clarity using `.rename()`.
    - Apply custom transformations to DataFrame rows or columns using `.apply()` (often with lambda functions) and perform element-wise mapping on Series using `.map()`.
    - Clean and manipulate string (text) data within Pandas Series using the `.str` accessor and relevant string methods.
    - Sort DataFrames based on the values in one or more columns (`.sort_values()`) or by the index labels (`.sort_index()`).
    - Identify and remove duplicate rows from DataFrames using `.duplicated()` and `.drop_duplicates()`, considering subsets of columns if necessary.
    - Employ various data wrangling techniques using Pandas to prepare clean, consistent, and analysis-ready datasets from raw business data.

Module 11: Data Visualization with Matplotlib

- **Content Summary:** Introduces data visualization concepts and the Matplotlib library. Covers creating common chart types (line, bar, scatter, histogram, pie), customizing plots (titles, labels, legends, colors, styles), and integrating Pandas plotting capabilities for quick visualizations.
- **Learning Outcomes:**
    - Explain why data visualization is a critical tool for understanding data patterns, communicating business insights, and supporting decision-making.
    - Import the `matplotlib.pyplot` module using the standard alias `plt`.
    - Generate common chart types—including line charts, bar charts (vertical and horizontal), scatter plots, histograms, and pie charts—using Matplotlib's `pyplot` functions.
    - Create basic plots directly from Pandas Series and DataFrames using the `.plot()` method and specifying the plot `kind`.
    - Customize Matplotlib plots by adding titles, axis labels, legends, grids, and adjusting colors, styles, and figure size to improve clarity and effectiveness.

    - Select appropriate chart types (line, bar, scatter, histogram, pie) based on the type of data being analyzed and the specific question being addressed.
    - Apply plotting techniques to visualize data stored and manipulated in Pandas DataFrames.

Module 12: Business Analytics Fundamentals and Applications

- **Content Summary:** Introduces the business analytics framework (descriptive, diagnostic, predictive, prescriptive analytics), core statistical concepts (measures of central tendency and dispersion, correlation), an introduction to predictive modeling (simple linear regression concepts), and the process of applying analytics and communicating findings.
- **Learning Outcomes:**
    - Define business analytics and explain its importance in supporting data-driven decision-making within organizations.
    - Differentiate between descriptive, diagnostic, predictive, and prescriptive analytics, providing examples of relevant business questions for each category.
    - Calculate and interpret common descriptive statistics (mean, median, standard deviation, range) using Pandas, explaining their significance in describing business data.
    - Calculate and interpret correlation coefficients using Pandas (`.corr()`) to quantify the strength and direction of linear relationships between variables, while recognizing the crucial distinction between correlation and causation.
    - Explain the fundamental concept of simple linear regression, including its purpose for modeling relationships and how the slope and intercept are used for interpretation and basic prediction.
    - Outline the key steps involved in a typical business analytics project, from defining the question to communicating the findings.
    - Identify key principles for effectively communicating analytical results and insights to various business stakeholders, leveraging narrative, interpretation, and visualization.

University Policies and Additional Guidelines

All policies outlined in this syllabus adhere to USF's academic guidelines and core syllabus policy statements. Students are expected to maintain academic integrity, adhere to deadlines, and engage in respectful communication. For further details, please refer to USF Core Syllabus Policy Statements.

Dr. Smith University of South Florida