Welcome to FINS5546: Toolkit for Fin Mkt Decisions

Breno Schmidt

- In preparation for this lecture please check that your microphone is muted.
- We'll be taking question by chat

This lecture will be recorded and made available for your ongoing reference.

UNSW SYDNEY

Instructors

Breno Schmidt

-Lecturer-in-charge

- breno.schmidt@unsw.edu.au

Yiping Lin

-Lecturer

- yiping.lin@unsw.edu.au

Consultation on Fridays (TBD)

Today

- Course information
- ED, Python, and PyCharm
- Lecture 1
- Part of Lecture 2

Course Information

An applied ***finance*** course that develops Python skills for working effectively and competitively in the modern job market

Our Philosophy

The most valuable commodity…

- The information revolution
  - Easy access to a daunting amount of (relevant) information
  - Potential for more informed decisions by market participants
- Challenges
  - Collecting, storing, processing vast amounts of data
  - Analysis
- Tools of the trade
  - Finance theory
  - Empirical methods
  - Computers and programming languages
    - Python as a tool

This course

- This is an applied Finance course
  - How to use effectively use technology in financial analysis
- Project based approach to programming
  - Finance:
    - Finance theory
    - Analytical methods such as event studies
  - Coding
    - Embed concepts within the context of projects
    - Emphasis on practicality
    - How to start thinking like a (Python) programmer
  - Develop critical reasoning and problem analysis

Why Python?

Python is a de facto standard for Finance professionals

- **Pros**
  - Reasonably straightforward syntax (easy to learn)
  - Clean, powerful constructs (e.g. not verbose, yet not limiting)
  - Standard paradigm (e.g. object oriented)
  - Excellent support for many cutting-edge technologies (e.g. machine learning)
- **Cons**
  - **Very easy to write codes that will not behave as expected**
  - Slow (relative to compiled languages like C++, Rust, Haskell, etc…)
  - Issues with multithreading, GIL, type-checking…

Required programming background

- None
  - You need not have used Python before.
  - You need not have programmed before.
- We will teach as if
  - You have not used Python before.
  - You have not programmed before.

What you will learn

- Basics of Python syntax
    - Including how to create functions, modules, packages
- PyCharm: A fully functional programming environment
    - Setup and third-party libraries
    - Executing Python programs in PyCharm
- How to organise and manipulate data using Pandas
- A different approach to financial analysis
    - Design and use Python libraries and modules
        - Break down a complex task into intermediary steps
        - Combine user-defined and third-party codes
- **How to avoid common mistakes and bad programming habits**

What you will NOT learn

- You will **not** become a proficient Python programmer in ten weeks
  - Takes a long time to become a competent Python programmer
- There are many important concepts **we will not cover**
  - Creating your own classes
  - Data visualisation tools
  - Doctests, test-driven development
  - Serialization, databases, REST APIs, ML
- Other topics **not covered** include
  - Inheritance, metaprogramming (decorators, ABC, metaclasses), concurrency, optimisation, design patterns, etc...

Resources and Technology

1. Moodle
   - Log into ED
   - Links to lectures/recordings (Zoom)
2. Zoom
   - All lectures will be recorded
3. ED
   - Course materials
   - Assessments
   - All course communication (Discussion Board)

ED: Lessons and Quizzes

- Lessons
  - Self-contained and include the required reading material
  - No required textbook
- Weekly quizzes and code challenges
  - Must be completed individually
  - Solutions provided one week after due date
- Other resources?
  - Official Python/Pandas docs
  - Online videos (Youtube, etc…)?

The Discussion Board

- All course communication will take place on ED
- Please post all questions to the discussion board
  - Please **do not** post questions related to Assessments
- Feel free to answer questions posted by other students
  - Multiple students can answer/comment same question
  - No penalty for incorrect answers
- I will provide answers/clarifications as required

PyCharm

Development environment for Python

- Download and install:
  - Instructions provided in ED
  - Community edition
- You must use PyCharm in this course
- Why PyCharm
  - Real-world as opposed to the educational environment in ED
  - Ensures that you will be able to actually implement projects
  - Better at handling large projects and debugging than Jupyter

Please go to ED via the Moodle Link

UNSW
SYDNEY

Important course information

1. Log into Moodle
2. Click on the ED link
3. Navigate to the Lessons page
4. Open the “Important Course Information” lesson

Lecture 1

Our first project

Downloading stock price data in bulk

Finance objectives include:

- A tool that maintains current return data is useful for many tasks in Finance
- The tool should:
    - Download multiple stocks
    - Update when necessary

Programming objectives include:

- Learn the basics of Python syntax
- Understand the basics of:
    - Objects
    - Assignment statements
    - Methods
    - Flow control

Manually downloading stock return data

Using a browser

Manually downloading return data:

- Navigate to Yahoo! Finance.
- Enter the stock ticker for Qantas.
- Select the correct time period
- Click on Download
- Save the file

Observations:

- Steps need to be executed in order
- Some steps would not change for Wesfarmers (WES.AX), others would

Using Python

We’ll use the following code in Python. It is also posted in Ed.

```
import yfinance
tic = "QAN.AX"
start = ’2020-01-01’
end = None
df = yfinance.download(tic, start, end)
df.to_csv(’qan_stk_prc.csv’)
```

Basic program flow in Python is to execute each statement from top to bottom.

Understanding the code

- There are certain concepts you need to know first
  - We be covered in the first couple of weeks
- However…
  - Try to run the code yourself before next week
  - This will ensure PyCharm is working as expected
  - Detailed instructions in ED
    - Under “Python & PyCharm”

Lecture 2

Objects

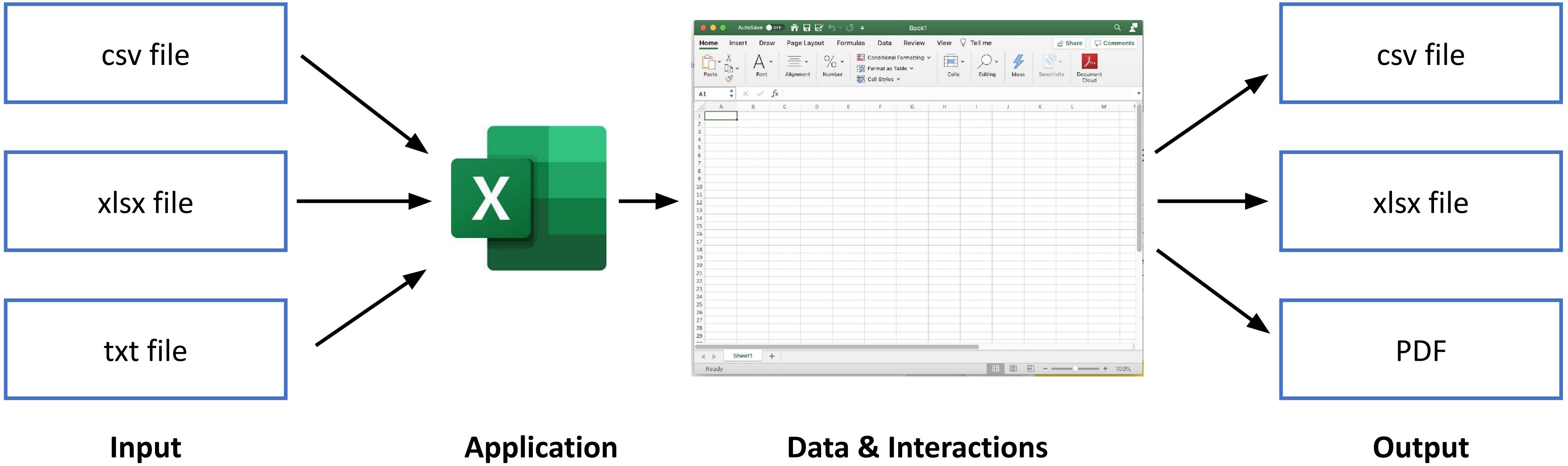
A proper definition

Definition of an instance in Python

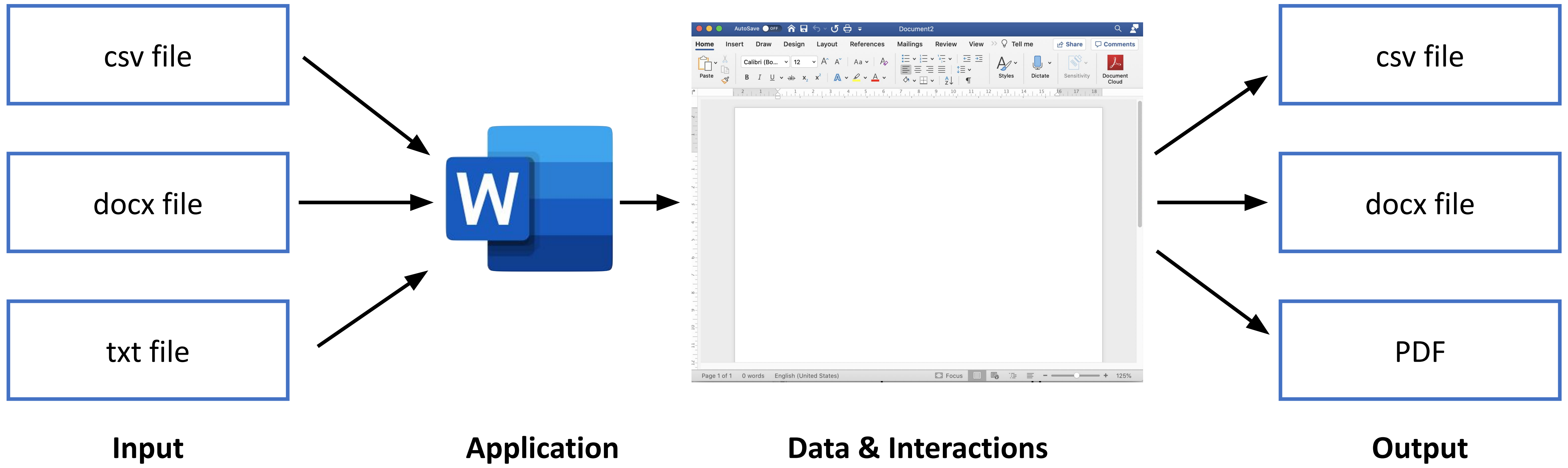
An instance of a class is a Python object with arbitrarily named attributes that you can bind and reference. An instance object implicitly delegates to its class the lookup of attributes not found in the instance itself. The class, in turn, may delegate the lookup to classes from which it inherits, if any.

-- Python in a Nutshell, 3rd Edition

Office application process flow



Office application process flow


csv file
docx file
txt file
Document2
Home Insert Draw Design Layout References Mailings Review View
Page 1 of 1   0 words   English (United States)
csv file
docx file
PDF
Input
Application
Data & Interactions
Output

Office application analogy

Applications define

- (i) the type of information, and
- (ii) possible interactions with the information

Each file we use as input creates a new ***instance*** of the application.

- Instances have the same functionality regardless of the input.
- The only thing that changes across instances is the information (data)

Interactions occur through ***menus and functions***

Objects in programming

Classes define

- The type of the objects it creates
- How these objects interact with other objects
    - "What these objects can do"
- The functionality ("functions") available to instances

Each time we create a new ***instance*** of a class:

- Instances have the same functionality as other instances created by the same class.
    - The functions inherited from the class are called **methods**
- The only thing that changes across instances is the information (data)

Intuitive definition

Intuition:

Classes are factories that create instances of their type. If Excel was a Python class, “opening” an Excel document tells the Excel class to create an instance of the Excel type. The way you interact with one Excel instance is similar to how you interact with any other Excel instance (that is, each instance will have the same the same methods). But it is different than how you interact with instances created by a “Word class”...

This is the definition you should understand

Microsoft Excel “class”

csv file

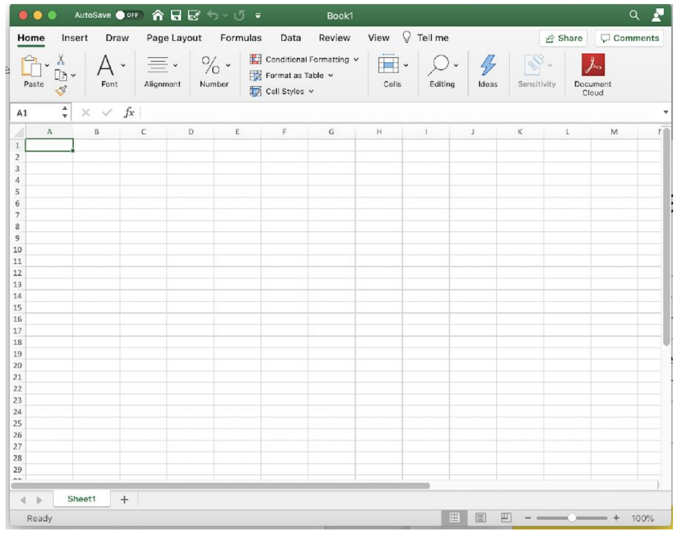
xlsx file

txt file

Input

Class

Instance



Data (attributes)

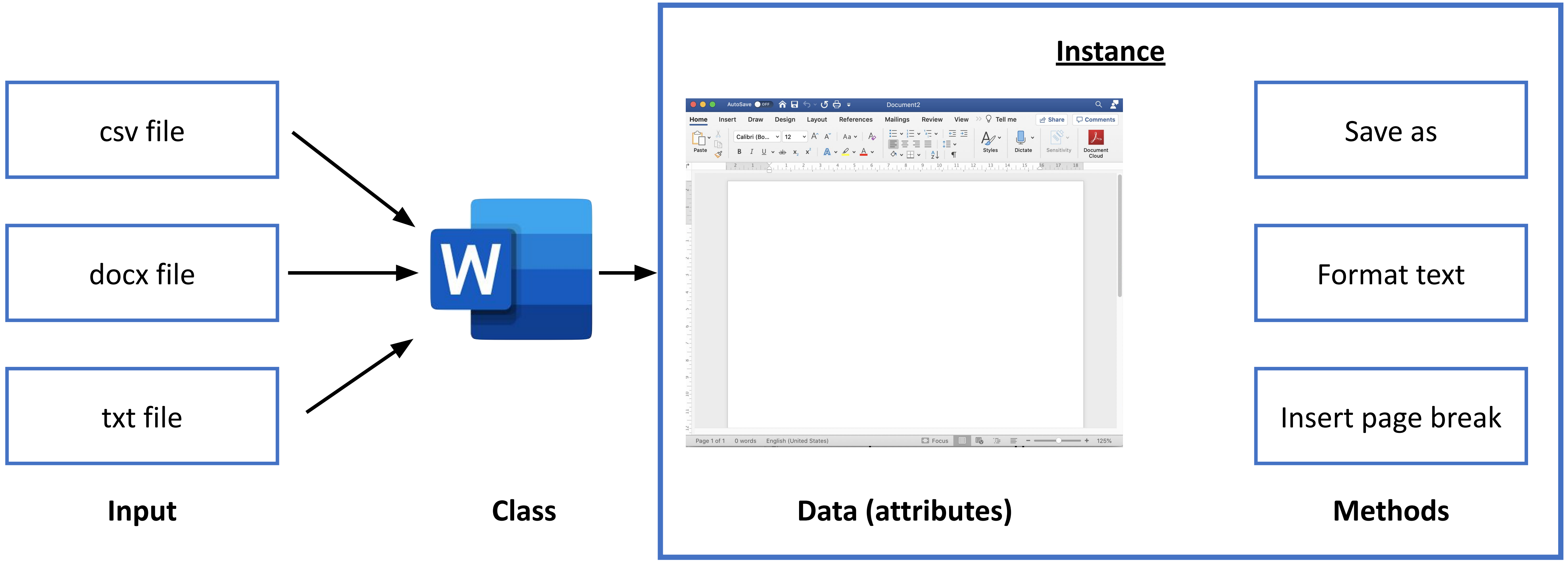
Save as

Sort

Mathematical functions

Methods

Office application process flow



Example (Python)

Two common classes in Python

- “str” → Create instances of the “str” type
    - These are “strings” of characters
    - Used to represent text
- “int” → Creates instances of the “int” type
    - These are integers (whole numbers)

These classes provide different functionality

Variables and assignment

Using Python

```
import yfinance
tic = "QAN.AX"
start = ’2020-01-01’
end = None
df = yfinance.download(tic, start, end)
df.to_csv(’qan_stk_prc.csv’)
```

Assignment

Variables:

- Store information that can be used or changed in your code.
- Are ***assigned*** objects using a single equals sign in Python

variable = expression

Assignment statements

Assignment statements have the form:

variable = expression

Python interprets these statements by:

- Checking if ***variable*** is a valid name
- Evaluating the ***expression*** and determining the result object
- Assigning the result of the ***expression*** to the ***variable***

Once assigned, Python uses the object whenever it sees the variable.

Variable names

Valid names must:

- Consist of any combination of letters, numbers, or underscores, _.
- Have a ***first*** character this is a letter or an underscore, _.
- Not be a keyword reserved by Python.

Names are ***case sensitive:***

- ***variable, Variable, VARIABLE, vArlaBle*** are all different

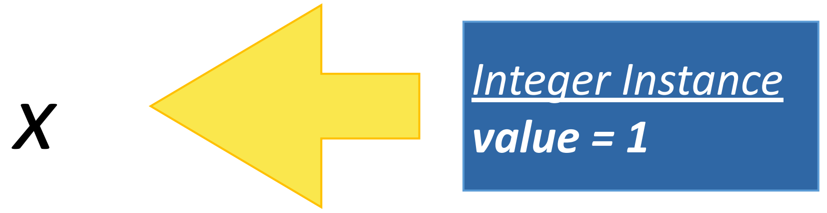
Expression evaluation

Expressions are evaluated by manipulating objects:

$$x \quad = \quad 1$$

Expression evaluation

Expressions are evaluated by manipulating objects:



- Python ensures *x* is a valid variable name
- Sees a number without quotation marks or a decimal places and identifies the corresponding object
- Assigns the object to the variable

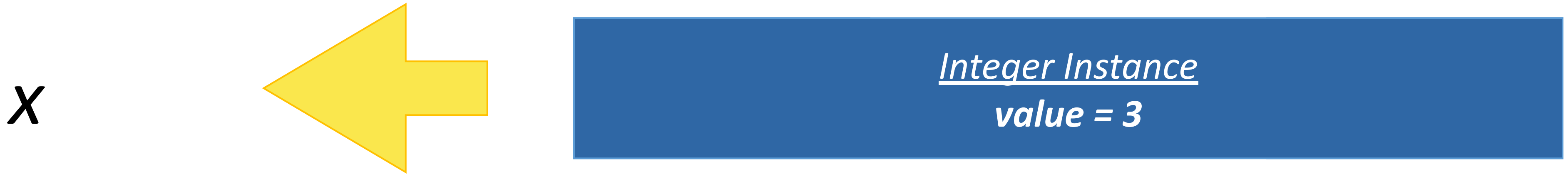
Expression evaluation

Expressions are evaluated by manipulating objects:

$$x \quad = \quad 1 \quad + \quad 2$$

Expression evaluation

Expressions are evaluated by manipulating objects:



- Python ensures *x* is a valid variable name
- Sees numbers without quotation marks or decimal places and identifies the corresponding object
- Performs the operation (addition)
- Assigns the result object to the variable

Expression evaluation

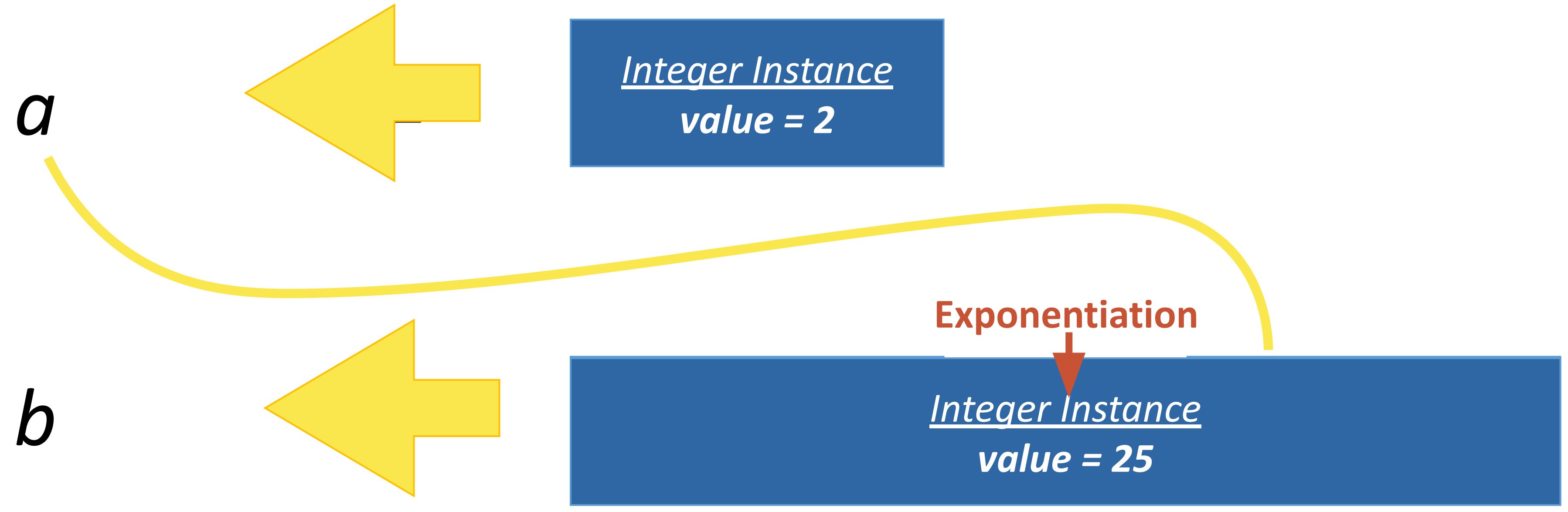
Expressions are evaluated by manipulating objects:

$$a = 2$$

$$b = 5 ** a$$

Expression evaluation

Expressions are evaluated by manipulating objects:



Expression evaluation

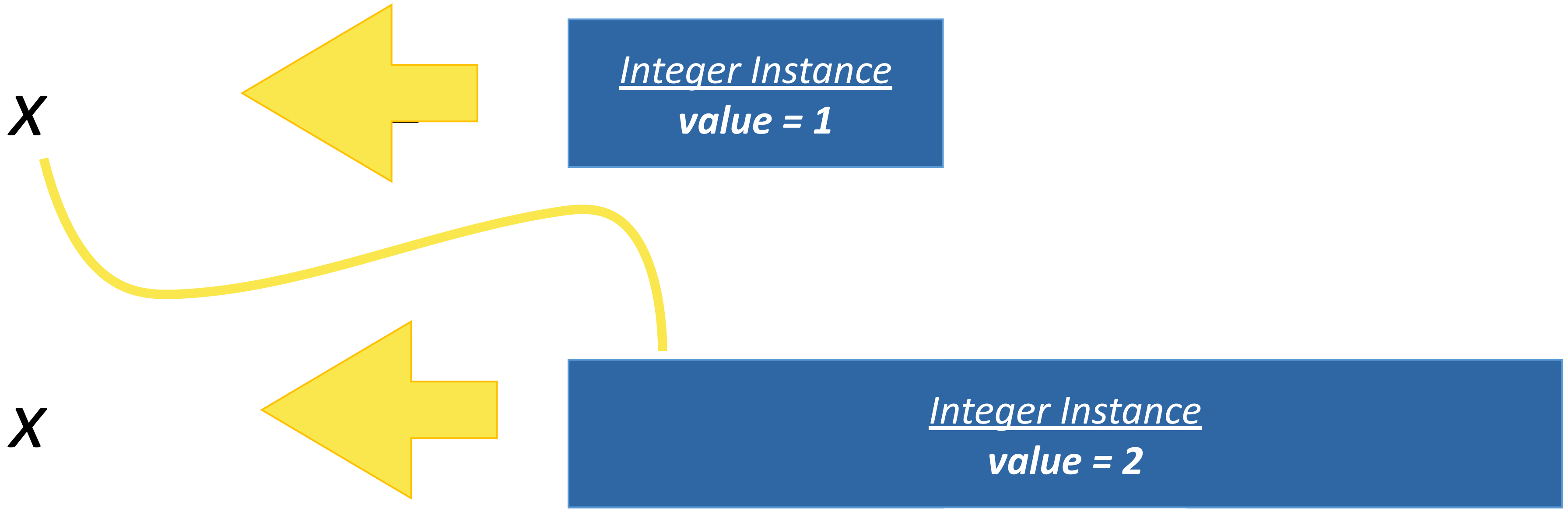
Expressions are evaluated by manipulating objects:

$$x = 1$$

$$x = x + 1$$

Expression evaluation

Expressions are evaluated by manipulating objects:


x
Integer Instance
value = 1
x
Integer Instance
value = 2