Python v2

Student Edition

Presented By

LearnKey®

Python v2
Project Workbook

First Edition

LearnKey creates signature multimedia courseware. LearnKey provides expert instruction for popular computer software, technical certifications, and application development with dynamic video-based courseware and effective learning management systems. For a complete list of courses, visit https://www.learnkey.com.

www.learnkey.com

LearnKey

Table of Contents

Introduction

Python v2

LK LearnKey

Best Practices Using LearnKey's Online Training

LearnKey offers video-based training solutions that are flexible enough to accommodate private students and educational facilities and organizations.

Our course content is presented by top experts in their respective fields and provides clear and comprehensive information. The full line of LearnKey products has been extensively reviewed to meet superior quality standards. Our course content has also been endorsed by organizations such as Certiport, CompTIA®, Cisco, and Microsoft. However, it is the testimonials given by countless satisfied customers that truly set us apart as leaders in the information training world.

LearnKey experts are highly qualified professionals who offer years of job and project experience in their subjects. Each expert has been certified at the highest level available for their field of expertise. This expertise provides the student with the knowledge necessary to obtain top-level certifications in their chosen field.

Our accomplished instructors have a rich understanding of the content they present. Effective teaching encompasses presenting the basic principles of a subject and understanding and appreciating organization, real-world application, and links to other related disciplines. Each instructor represents the collective wisdom of their field and within our industry.

Our Instructional Technology

Each course is independently created based on the manufacturer's standard objectives for which the course was developed.

We ensure that the subject matter is up-to-date and relevant. We examine the needs of each student and create training that is both interesting and effective. LearnKey training provides auditory, visual, and kinesthetic learning materials to fit diverse learning styles.

Course Training Model

The course training model allows students to undergo basic training, building upon primary knowledge and concepts to more advanced application and implementation. In this method, students will use the following toolset:

Pre-assessment: The pre-assessment is used to determine the student's prior knowledge of the subject matter. It will also identify a student's strengths and weaknesses, allowing them to focus on the specific subject matter they need to improve the most. Students should not necessarily expect a passing score on the pre-assessment as it is a test of prior knowledge.

Video training sessions: Each training course is divided into sessions or domains and lessons with topics and subtopics. LearnKey recommends incorporating all available external resources into your training, such as student workbooks, glossaries, course support files, and additional customized instructional material. These resources are located in the folder icon at the top of the page.

Exercise labs: Labs are interactive activities that simulate situations presented in the training videos. Step-by-step instructions and live demonstrations are provided.

Post-assessment: The post-assessment is used to determine the student's knowledge gained from interacting with the training. In taking the post-assessment, students should not consult the training or any other materials. A passing score is 80 percent or higher. If the individual does not pass the post-assessment the first time, LearnKey recommends incorporating external resources, such as the workbook and additional customized instructional material.

Workbook: The workbook has various activities, including fill-in-the-blank questions, short answer questions, practice exam questions, and group and individual projects that allow the student to study and apply concepts presented in the course videos.

Using This Workbook

This project workbook contains practice projects and exercises to reinforce the knowledge you have gained through the video portion of the **Python v2** course. The purpose of this workbook is twofold. First, get you further prepared to pass the IT Specialist – Python exam, and second, to teach you job-ready skills and increase your employability in the area of introductory-level knowledge of Python.

The projects within this workbook follow the order of the video portion of this course. To save your answers in this workbook, you must first download a copy to your computer. You will not be able to save your answers in the web version. You can complete the workbook exercises as you go through each section of the course, complete several at the end of each domain, or complete them after viewing the entire course. The key is to go through these projects to strengthen your knowledge in this subject.

Each project is based upon a specific video (or videos) in the course and specific test objectives. The materials you will need for this course include:

- LearnKey's **Python v2** courseware.
- Python software.
- Visual Studio Code software.
- The course project files. All applicable project files are located in the support area where you downloaded this workbook.

For Teachers

LearnKey is proud to provide extra support to instructors upon request. For your benefit as an instructor, we also provide an instructor support .zip file containing answer keys, completed versions of the workbook project files, and other teacher resources. This .zip file is available within your learning platform's admin portal.

Notes

- Extra teacher notes, when applicable, are in the Project Details box within each exercise.
- Exam objectives are aligned with the course objectives listed in each project, and project file names correspond with these numbers.
- The Finished folder in each domain has reference versions of each project. These can help you grade projects.
- Short answers may vary but should be similar to those provided in this workbook.
- Teachers may consider asking students to add their initials, student ID, or other personal identifiers at the end of each saved project.
- Refer to your course representatives for further support.

We value your feedback about our courses. If you have any questions, comments, or concerns, please let us know by visiting https://about.learnkey.com.

LearnKey

Skills Assessment

Instructions: Rate your skills on the following tasks from 1-5 (1 being needs improvement, 5 being excellent).

Skills	1	2	3	4	5
Evaluate expressions to identify the data types Python assigns to variables.	☐	☐	☐	☐	☐
Perform and analyze data and data type operations.	☐	☐	☐	☐	☐
Determine the sequence of execution based on operator precedence.	☐	☐	☐	☐	☐
Select operators to achieve the intended results.	☐	☐	☐	☐	☐
Construct and analyze code segments that use branching statements.	☐	☐	☐	☐	☐
Construct and analyze code segments that perform iteration.	☐	☐	☐	☐	☐
Construct and analyze code segments that perform file input and output operations.	☐	☐	☐	☐	☐
Construct and analyze code segments that perform console input and output operations.	☐	☐	☐	☐	☐
Document code segments.	☐	☐	☐	☐	☐
Construct and analyze code segments that include function definitions.	☐	☐	☐	☐	☐
Analyze, detect, and fix code segments that have errors.	☐	☐	☐	☐	☐
Analyze and construct code segments that handle exceptions.	☐	☐	☐	☐	☐
Perform unit testing.	☐	☐	☐	☐	☐
Perform basic file system and command-line operations by using built-in modules.	☐	☐	☐	☐	☐
Solve complex computing problems by using built-in modules.	☐	☐	☐	☐	☐
	☐	☐	☐	☐	☐
	☐	☐	☐	☐	☐
	☐	☐	☐	☐	☐
	☐	☐	☐	☐	☐

LearnKey

Python v2 Video Times

Domain 1	Video Time
Identify Data Types	00:18:43
Analyze Data Types and Operators	00:32:04
Sequence of Execution	00:18:05
Select Operators	00:24:50
Total Time	**01:33:42**

Domain 2	Video Time
Branching Statements	00:16:04
Iteration	00:20:06
Total Time	**00:36:10**

Domain 3	Video Time
File Input and Output	00:22:35
Console Input and Output	00:14:43
Total Time	**00:37:18**

Domain 4	Video Time
Document Code Segments	00:14:35
Function Definitions	00:13:10
Total Time	**00:27:45**

Domain 5	Video Time
Analyze, Detect, and Fix Errors	00:11:48
Exception Handling	00:14:58
Perform Unit Testing	00:11:38
Total Time	**00:38:24**

Domain 6	Video Time
System and Command-Line Operations	00:14:57
Solve Complex Problems	00:24:24
Total Time	**00:39:21**

Domain 1
Lesson 1

Python v2

LK LearnKey

Python Introduction

Python is a versatile programming language with a wide variety of uses, including basic apps, web development, data analysis, artificial intelligence, the Internet of Things, automation, and game development. This workbook helps users practice using Python and reinforces the concepts covered in the video portion of this course.

Purpose

Upon completing this project, you will better understand general facts about Python.

Steps for Completion

1. Label the following statements as true or false.
    a. _____ Python code uses curly brackets.
    b. _____ If Python code is not indented correctly, it will not work.
    c. _____ Python is an interpreted language, meaning one can write and run code in the Python engine.
    d. _____Classes cannot be created in Python, but objects can be.
    e. _____ Python is a dynamically typed language, meaning that data types do not need to be declared on variables before the variables themselves are declared.

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Identify Data Types
Subtopic: How to Study for This Exam; What Is Python?

Objectives covered
1 Operations Using Data Types and Operators
1.1 Evaluate expressions to identify the data types Python assigns to variables

LK LearnKey

Installing Python

Before one can practice using Python, they must have the proper tools installed on their device. This project is essential, as many other projects cannot be completed without Python and Visual Studio Code. The content displayed on the following websites may differ from this course because these tools are updated frequently.

Purpose

Upon completing this project, you will have installed Python and Visual Studio Code and ensured they work properly.

Steps for Completion

1. Download the latest version of Python at **python.org**
2. Download Visual Studio Code as your Integrated Development Environment app at **code.visualstudio.com**
    a. You can take the default installation once you have installed the Python engine.
3. Verify that Python is installed by opening a command prompt and running **py**
    a. If there are three arrows and Python information, Python is installed and ready for use.
4. Close the Python session within the command prompt.
5. Open Visual Studio Code.
6. Install the Python extension.
7. Restart Visual Studio Code if necessary.
8. Open the **Domain 1 Student** folder within Visual Studio to easily access files.
    a. You may open all six domain Student folders now, or when you begin projects for each domain.

Project Details

Project file
Domain 1 Student folder

Estimated completion time
10-15 minutes

Video reference
Domain 1
Topic: Identify Data Types
Subtopic: What Needs to Be Installed

Objectives covered
1 Operations Using Data Types and Operators
1.1 Evaluate expressions to identify the data types Python assigns to variables

Domain 1
Lesson 2

Python v2

LearnKey

Strings and Integers

Strings and integers are data types that can be added to variables within Python code. A variable is a container that stores a value, like a string of text, a number, or other data types. Strings of text go inside quotation marks and can be used in various ways. An integer is a numeric data type that represents a whole number. Variable names can contain letters, numbers, and underscores, but not symbols or spaces. Python's standard naming convention is all lowercase characters with underscores replacing spaces.

Purpose

Upon completing this project, you will better understand how to use strings and integers in Visual Studio. The files needed for this project should already be in the program.

Steps for Completion

1. Open the **111-str.py** file from your Domain 1 Student folder within Visual Studio.
2. Note the two variables with strings containing a first and last name.
3. Add a print statement on line 3 to make the code output a full name.
    a. The order of the names can be last, then first, or first, then last.
        i. If the names are last, then first, they must be separated by a comma.
4. Run the code to ensure it outputs a first and last name in your chosen order.
5. In Python, users can add single or double quotes around text strings but not _____ quotes.
6. Why will the following print statement cause an error message to appear when it is run?

    print ('Jason's turn')

    a. _____
    _____
7. Save the file as **111-str-completed**
8. Open the **112-numbers.py** file.
9. Numbers represented as _____ cannot be used in calculations.
10. Add a print statement on line 4 that prints the data type of the number_of_tries variable to verify that it is an integer.
11. Run the code to verify that the data type on line 1 is an integer.
12. Save the file as **112-numbers-completed**

Project Details

Project file
111-str.py
112-numbers.py

Estimated completion time
10-15 minutes

Video reference
Domain 1
Topic: Identify Data Types
Subtopic: str; int

Objectives covered
1 Operations Using Data Types and Operators
1.1 Evaluate expressions to identify the data types Python assigns to variables
1.1.1 str
1.1.2 int

LearnKey

Floats and Bools

Floats and bools are other data types used in Python code. While integers are whole numbers, floating numbers, or shorts, are numbers with decimals. The Boolean data type is either true or false or, in binary terms, 1 or 0.

Purpose

Upon completing this project, you will better understand how to use floats and bools in Python coding.

Steps for Completion

1. Open the **113-numbers.py** file from your Domain 1 Student folder.
2. Note that the multiplier variable on line 2 is a float data type.
3. Add a print statement on line 4 that calculates the sum of the number_of_tries and multiplier variables. Run the code.
4. Which data type did the calculation return?
    a. _____
5. Change the operator on line 4 from addition to division. Run the code.
6. Which data type did the calculation return?
    a. _____
7. Save the file as **113-numbers-completed**
8. Python code written for games with scores will likely use _____, while games involving money will probably need _____.
9. Open the **114-boolean.py** file.
10. Note the Boolean variable on line 1, char_life, which is set to True. Run the code.
11. Save the file as **114-boolean-completed**
12. Python is case-sensitive, and Boolean variables must be in _____ casing.

Project Details

Project file
113-numbers.py
114-boolean.py

Estimated completion time
10-15 minutes

Video reference
Domain 1
Topic: Identify Data Types
Subtopic: float; bool

Objectives covered
1 Operations Using Data Types and Operators
1.1 Evaluate expressions to identify the data types Python assigns to variables
1.1.3 float
1.1.4 bool

LK LearnKey

Review 1.1

Purpose

Upon completing this project, you will better understand data types and be able to identify data types for data better.

Steps for Completion

1. Open the **114-analyze.py** file from your Domain 1 Student folder.
2. Identify the data types for each of the five print statements:
    a. _____
    b. _____
    c. _____
    d. _____
    e. _____
3. Close the file and return to the videos.

Project Details

Project file
114-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Identify Data Types
Subtopic: bool; Review on 1.1

Objectives covered
1 Operations Using Data Types and Operators
1.1 Evaluate expressions to identify the data types Python assigns to variables

Domain 1
Lesson 3

Python v2

LearnKey

Data Type Conversion

Programmers must understand how to convert data from one type to another because sometimes data can be input or imported as an incorrect data type. Converting data types is necessary to make data usable in given situations.

Coding information to be aware of includes functions and methods. A function is a procedure that performs an action. Python has a variety of built-in functions, such as int for converting to integers and str for strings. Users can also define custom functions. A method is a specific type of function that belongs to an instance of a class or an object and can access or modify the instance or object to which it belongs.

Purpose

Upon completing this project, you will have experience converting data types within Python code.

Steps for Completion

1. Open the **121-conversion.py** file from your Domain 1 Student folder.
2. Note that the variable on line 2 stores a rating someone enters (which should be a whole number). Run the code.
3. Enter a float value between 1 and 5 within the terminal, and rerun the code.
4. What data type is stored in the input variable by default?
    a. _____
5. Convert the rating variable on line 3 to an integer. Run the code.
6. Enter a float value between 1 and 5 within the terminal, and rerun the code.
7. Convert the points value on line 4 to a string. Run the code.
8. Enter an integer value between 1 and 5 within the terminal, and rerun the code.
9. Enter a float value between 1 and 5 within the terminal, and rerun the code.
10. Convert the input on line 2 to a float. Run the code.
11. Enter an integer value between 1 and 5 within the terminal, and rerun the code.
12. Save the file as **121-conversion-completed**

Project Details

Project file
121-conversion.py

Estimated completion time
15 minutes

Video reference
Domain 1
Topic: Analyze Data Types and Operators
Subtopic: Data Type Conversion

Objectives covered
1 Operations Using Data Types and Operators
1.2 Perform and analyze data and data type operations
1.2.1 Data type conversion

Indexing

To best understand indexing, programmers need to understand lists and list management. A list is a set of multiple values assigned to a single variable. Lists have square brackets surrounding the values within the list. Indexing is the positioning of each element within a list. Most programming languages, including Python, are zero-based with lists and their values, meaning that the first value in the list is at index zero, the next is index one, and so forth.

Purpose

Upon completing this project, you will better understand how to navigate indexed data.

Steps for Completion

1. Open the **122-indexing.py** file from your Domain 1 Student folder.
2. Note that line 1 lists categories for users to choose and line 2 has a print statement to print the categories variable. Run the code.
3. Copy the code on line 2 and paste it on line 4, editing it so that only the first category, Early Bronze Age, prints. Run the code.
4. Change the category value on line 4 to 2. Run the code.
5. Which item in the list was returned?
    a. _____
6. Without counting all the list items, change the category value on line 4 so that the last category in the list returns. Run the code.
7. Save the file as **122-indexing-completed**

Project Details

Project file
122-indexing.py

Estimated completion time
10 minutes

Video reference
Domain 1
Topic: Analyze Data Types and Operators
Subtopic: Indexing

Objectives covered
1 Operations Using Data Types and Operators
1.2 Perform and analyze data and data type operations
1.2.2 Indexing

Domain 1
Lesson 4

Python v2

Slicing

Whereas indexing extracts a value from a list, slicing extracts characters from a list, word, or phrase. Knowing how to slice a value within Python code can save one from having to extract text elsewhere, saving time and potential errors within an app. Users can extract characters from a variable's beginning, middle, and end.

Purpose

Upon completing this project, you will better understand how to extract data from a variable.

Steps for Completion

1. Open the **123-slicing.py** file from your Domain 1 Student folder.
2. Note that line 1 consists of a variable called serial_number, a series of sixteen numbers.
3. Add a print function on line 2 to output the first four characters from the serial_number variable. Run the code.
4. Copy the code on line 2 and paste it on line 4, editing it so that only the last four characters of the variable print. Run the code.
5. Copy the code on line 4 and paste it on line 5, editing it so that the eight middle characters of the variable print. Run the code.
6. Save the file as **123-slicing-completed**

Project Details

Project file
123-slicing.py

Estimated completion time
10 minutes

Video reference
Domain 1
Topic: Analyze Data Types and Operators
Subtopic: Slicing

Objectives covered
1 Operations Using Data Types and Operators
1.2 Perform and analyze data and data type operations
1.2.3 Slicing

LearnKey

Data Structures

Data structures, or collection data types, consist of lists stored inside variables. Additional data types to understand when creating data structures include dictionaries, sets, and tuples. A dictionary stores data in key-value pairs. A set is a type of collection that stores unique values of data. A tuple is a list with immutable values. Programmers should know what each structure type does and the syntax used to build it. This knowledge can help make apps perform more smoothly.

Purpose

Upon completing this project, you will better understand data structures.

Steps for Completion

1. Strings are data structures as they are collections of _____.

2. Match each code example to its correct data type.

A. Dictionary	B. Set	C. Tuple

a. _____

```
regions = {"north","south","east"}
regions.add("west")
print(regions)
```

b. _____

```
char1_name=("Humphrey",'Cat')
```

c. _____

```
coins = {
    "gold":100,
    "silver":50,
    "bronze":25
}
print(coins)
```

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Analyze Data Types and Operators
Subtopic: Construct Data Structures

Objectives covered
1 Operations Using Data Types and Operators
1.2 Perform and analyze data and data type operations
1.2.4 Construct data structures

Domain 1
Lesson 5

Python v2

Lists and Their Operations

A list is a collection of items in a defined order. A list's primary purpose is to store a list of values in a single variable. Some list operations one can apply to a list include built-in methods. Methods are functions tied to a specific object and invoked using the period (.) notation. Particular methods include append, which adds an item to the end of a list, and insert, which inserts an item at a specified position.

Purpose

Upon completing this project, you will better understand lists and their operations.

Steps for Completion

1. When a list of values needs to be adjusted, one does not need to declare a _____ for every value.
2. In the following code example, lines 2 and 3 store information in separate variables. Why would one do this instead of using a list?

```
1  capitals=["Montgomery","Juneau","Phoenix"]
2  first_name="Star"
3  last_name="Metal"
4  print(capitals)
```

   a. _____
   _____
   _____

3. Open the **126-list_operations.py** file from your Domain 1 Student folder.
4. Use the append method on line 2 to add Sacramento to the capitals list. Run the code.
5. In programming, a period (.) is the accessor that allows one to use a _____ within an object.
6. Answer the questions about the following code:

   capitals.append()

   a. What is the object?
      i. _____
   b. What is the method?
      i. _____

7. Use the insert method on line 3 to add Little Rock to the capitals list before Sacramento and rerun the code.
8. Save the file as **126-list_operations-completed**

Project Details

Project file
126-list_operations.py

Estimated completion time
10 minutes

Video reference
Domain 1
Topic: Analyze Data Types and Operators
Subtopic: Lists; List Operations

Objectives covered
1 Operations Using Data Types and Operators
1.2 Perform and analyze data and data type operations
1.2.5 Lists
1.2.6 List operations

LearnKey

Review 1.2

Purpose

Upon completing this project, you will better understand lists as a whole and how changing one bit of code can affect multiple areas of code.

Steps for Completion

1. Open the **126-list_operations.py** file from your Domain 1 Student folder.
    a. You can use this as a live file to compliment what is in the video reference.
2. Determine why, when the code is run in the video, Little Rock shows twice.
    a. _____
3. What are two possible remedies to the problem described in the video?
    a. _____
    b. _____
4. Return to the videos for the answer.
5. Open the **126-analyze.py** file from your Domain 1 Student folder.
6. What will the print statement on line 2 print?
    a. _____
7. What type of data structure is on line 4?
    a. _____
8. How can the coins variable be adjusted to have a bronze value instead of a platinum value?
    a. _____

Project Details

Project file
126-list_operations.py
126-analyze.py

Estimated completion time
10 minutes

Video reference
Domain 1
Topic: Analyze Data Types and Operators
Subtopic: List Operations; Review on 1.2 Part 1; Review on 1.2 Part 2

Objectives covered
1 Operations Using Data Types and Operators
1.2 Perform and analyze data and data type operations

Domain 1
Lesson 6

Python v2

LearnKey

Assignment Operators

Operators help perform calculations and check for conditions within a block of code. From first to last, the six types of operators that can be applied are arithmetic, containment, comparison, identity, logical, and assignment. Assignments set values to variables.

Purpose

Upon completing this project, you will better understand assignment operators.

Steps for Completion

1. _____ supersede the six operators in the order of operations.
2. Review the code, then answer the question regarding variable assignments.
    a. What will happen when this code is run?

```
131-assignment.py > ...
1   x = 5
2   y = 3
3   print(x, y)
4   y = x
5   print(y)
```

i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Sequence of Execution
Subtopic: Assignment Order

Objectives covered
1 Operations Using Data Types and Operators
1.3 Determine the sequence of execution based on operator precedence
1.3.1 Assignment

Comparison Operators

A comparison operator compares two values. The comparisons include equal to, not equal to, less than, greater than, less than or equal to, and greater than or equal to. These comparisons can help users identify whether calculations in code are returning expected results.

Purpose

Upon completing this project, you will better understand comparison operators.

Steps for Completion

1. Comparisons return a(n) _____ value.
2. Comparisons happen _____ assignments in a sequence.
3. What is true with an expression if part of a comparison is false?
    a. _____
    _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Sequence of Execution
Subtopic: Comparison Order

Objectives covered
1 Operations Using Data Types and Operators
1.3 Determine the sequence of execution based on operator precedence
1.3.2 Comparison

LearnKey

Logical Operators

Logical operators use three keywords in Python: not, and, and or. These keywords have an order of operations that users must understand to implement logical operators and get the intended results.

Purpose

Upon completing this project, you will better understand logical operators.

Steps for Completion

1. List the order of the logical keywords in the order of operations from first to last.
    a. _____
    b. _____
    c. _____
2. Review the code, then answer the question regarding the logical keywords' order of operations.

```
1 unlock_level = True
2 target_score = 10000
3 score = 8000
4 if score < target_score or not unlock_level:
5     print("You still have some goals to make")
6 else:
7     print("Goals met")
```

    a. What will happen when this code is run?
        i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Sequence of Execution
Subtopic: Logical Order

Objectives covered
1 Operations Using Data Types and Operators
1.3 Determine the sequence of execution based on operator precedence
1.3.3 Logical

Domain 1
Lesson 7

Python v2

LearnKey

Arithmetic

Arithmetic has an order of operations when performing calculations. This order ensures that calculations are performed the same way every time to provide consistent results.

Purpose

Upon completing this project, you will better understand the order of operations.

Steps for Completion

1. List the order of operations from first to last.
    a. _____
    b. _____
    c. _____
    d. _____
    e. _____
    f. _____
2. Open the **134-arithmetic.py** file from your Domain 1 Student folder.
3. Run the code. What is the result?
    a. _____
4. Edit the code so that the base and bonus variables are added together before being multiplied by the rank variable.
5. Run the code. What is the result?
    a. _____
6. Save the file as **134-arithmetic-completed**

Project Details

Project file
134-arithmetic.py

Estimated completion time
10 minutes

Video reference
Domain 1
Topic: Sequence of Execution
Subtopic: Arithmetic Order

Objectives covered
1 Operations Using Data Types and Operators
1.3 Determine the sequence of execution based on operator precedence
1.3.4 Arithmetic

Identity Operator

Two variables can appear equal, but one can use the identity operator to determine if they are truly the same. The identity operator determines if two variables share the same memory location.

Purpose

Upon completing this project, you will better understand the identity operator.

Steps for Completion

1. The _____ keyword is used to identify whether two variables share the same memory space.
2. In the order of operations, the identity operation happens before logical and assignment operations but after arithmetic and _____operations.
3. If two variables share the same memory space, they will have _____ values.

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Sequence of Execution
Subtopic: Identity Order

Objectives covered
1 Operations Using Data Types and Operators
1.3 Determine the sequence of execution based on operator precedence
1.3.5 Identity (is)

LearnKey

Containment Operator

Containment is often used for lists, sets, tuples, and dictionaries. The containment operator checks whether a value is contained within a list of values.

Purpose

Upon completing this project, you will better understand the containment operator.

Steps for Completion

1. Containment operations happen _____ comparison, logical, and identity operations.
2. Review the code, then answer the question about the containment operator.

```
136-containment.py > ...
1  large_islands = ["Hokkaido", "Honshu", "Shikoku", "Kyushu"]
2  island1 = "Hokkaido"
3  print(island1 in large_islands)
```

   a. What result will the code on line 3 output?

      i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Sequence of Execution
Subtopic: Containment Order

Objectives covered
1 Operations Using Data Types and Operators
1.3 Determine the sequence of execution based on operator precedence
1.3.6 Containment (in)

LearnKey

Review 1.3

Purpose

Upon completing this project, you will better understand the order of operations across categories of operators and within a category of operators.

Steps for Completion

1. What is the order for the six types of operators?
    a. _____
    b. _____
    c. _____
    d. _____
    e. _____
    f. _____
2. What supersedes all operators in the order of operations?
    a. _____
3. What is the order of preference for logical operators?
    a. _____
    b. _____
    c. _____
4. What is the order of operations for arithmetic operators?
    a. _____
    b. _____
    c. _____
    d. _____
    e. _____
    f. _____

Project Details

Project file
N/A

Estimated completion time
15 minutes

Video reference
Domain 1
Topic: Sequence of Execution
Subtopic: Containment Order; Review on 1.3

Objectives covered
1 Operations Using Data Types and Operators
1.3 Determine the sequence of execution based on operator precedence

Domain 1
Lesson 8

Python v2

LearnKey

Using Assignment Operators

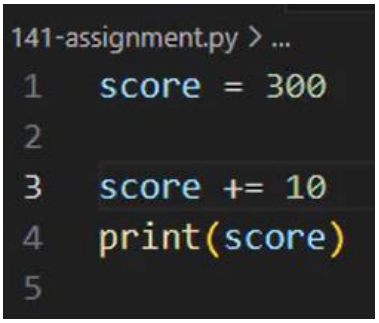
Assignments, which give a value to a variable, can be simple or compound. Many operations can be performed using assignments, including addition, subtraction, multiplication, exponents, and division. The modulus and floor operators are related to the division operator.

Purpose

Upon completing this project, you will better understand how to use assignments.

Steps for Completion

1. A(n) _____ assignment attaches a value to a variable using an equal sign.
2. The modulus is the _____ of two numbers divided by each other.
3. A(n) _____ assignment changes the value of a variable using an arithmetic operator.
4. Floor division returns the _____ of two numbers divided by each other.
5. Review the code, then answer the question about the assignment operator in the code.
    a. What result will the code on lines 3 and 4 produce?



    i. _____
6. What symbols are used to determine the modulus of two numbers?
    a. _____
7. What symbols are used for floor division?
    a. _____
8. What symbols are used to calculate an exponent to a number?
    a. _____

Project Details

Project file
N/A

Estimated completion time
10 minutes

Video reference
Domain 1
Topic: Select Operators
Subtopic: Assignment

Objectives covered
1 Operations Using Data Types and Operators
1.4 Select operators to achieve the intended results
1.4.1 Assignment

LearnKey

Using Comparison Operators

Comparison operators are used to evaluate the similarities between two variables. Understanding how to use the different comparison operators available helps ensure that one can understand how two variables relate to one another.

Purpose

Upon completing this project, you will better understand how to use comparison operators.

Steps for Completion

1. Review the code, then answer the questions regarding the comparison operators in the code.

    a. What result will the code on line 5 output?

```
142-comparison.py > ...
1  player1_score = 300
2  player2_score = 400
3  player3_score = 300
4
5  print(player1_score == player2_score)
6  print(player1_score < player3_score)
7  print(player1_score > player2_score)
```

        i. _____

    b. What result will the code on line 6 output?

        i. _____

    c. What result will the code on line 7 output?

        i. _____

```
142-comparison.py > ...
1  player1_score = 300
2  player2_score = 400
3  player3_score = 300
4
5  print(player1_score != player2_score)
6  print(player1_score <= player3_score)
7  print(player1_score > player2_score)
```

    d. What result will the code on line 5 output?

        i. _____

    e. What result will the code on line 6 output?

        i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Select Operators
Subtopic: Comparison

Objectives covered
1 Operations Using Data Types and Operators
1.4 Select operators to achieve the intended results
1.4.2 Comparison

LearnKey

Using Logical Operators

Logical operators involve using the keywords and, or, and not to compare two or more conditions. These keywords have their own order of operations. Developers must understand the order of operations for logical operators to know why statements containing logical operators produce the results they output.

Purpose

Upon completing this project, you will better understand how to use logical operators.

Steps for Completion

1. Review the code, then answer the questions regarding the logical operators in the code.

143-logical.py > ...

```
1  gases = ('Hydrogen','Helium','Nitrogen','Oxygen')
2  number_of_gases = 11
3  number_of_liquids = 2
4
5  print('Sodium' in gases and number_of_liquids < number_of_gases)
6
7  print('Hydrogen' in gases or 'Helium' in gases and number_of_liquids == 4)
```

   a. What result will the code on line 5 output? Why?

      i. _____

   b. What result will the code on line 7 output? Why?

      i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Select Operators
Subtopic: Logical

Objectives covered
1 Operations Using Data Types and Operators
1.4 Select operators to achieve the intended results
1.4.3 Logical

Domain 1
Lesson 9

Python v2

LearnKey

Using Arithmetic Operators

Arithmetic operators are an important part of performing calculations. Knowing what operators are available and the order in which they are performed can help developers write code efficiently and produce intended results, especially when testing code.

Purpose

Upon completing this project, you will better understand how to use arithmetic operators.

Steps for Completion

1. The quotient of two integers can be an integer or a(n) _____.
2. A(n) _____ operator makes a positive value negative and a negative value positive.
3. Review the code, then answer the questions regarding the arithmetic operators in the code.

```
144-arithmetic.py > ...
 7
 8  print (x * y)
 9
10  print (x / y)
11
12  print (x // y)
13
14  print (x % y)
15
16  print (x ** y)
17
18  #print( x + 2 * y)
19
```

a. What result will the code on line 12 output?

   i. _____

b. What result will the code on line 14 output?

   i. _____

c. What result will the code on line 16 output?

   i. _____

d. What result will the code on line 18 output?

   i. _____

Project Details

Project file
N/A

Estimated completion time
10 minutes

Video reference
Domain 1
Topic: Select Operators
Subtopic: Arithmetic

Objectives covered
1 Operations Using Data Types and Operators
1.4 Select operators to achieve the intended results
1.4.4 Arithmetic

LearnKey

Using the Identity Operator

The identity operator, is, is used to see if two variables are taking up the same memory space. This operator is especially useful when speed is an important factor.

Purpose

Upon completing this project, you will better understand how to use the identity operator.

Steps for Completion

1. Review the code, then answer the questions about the identity operators in the code.

    a. Why will the code on line 5 return True?

```
145-identity.py > ...
1  team1 = {"color":"red", "rank":1}
2  team2 = team1
3  team3 = {"color":"red", "rank":1}
4
5  print(team1 is team2)
6  print(team1 is team3)
7
```

    i. _____

    b. Why will the code on line 6 return False?

    i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Select Operators
Subtopic: Identity

Objectives covered
1 Operations Using Data Types and Operators
1.4 Select operators to achieve the intended results
1.4.5 Identity (is)

LearnKey

Using the Containment Operator

Containment uses the in keyword to determine whether a value is in a list of values. The containment operator can be especially useful when a variable in code contains a long list of values.

Purpose

Upon completing this project, you will better understand how to use the containment operator.

Steps for Completion

1. Review the code, then answer the questions regarding the containment operators in the code.

```
146-containment.py > ...
1  ancient_civilizations = ['Mesopotamia','Egypt','Indus Valley','China']
2
3  print('Egypt' in ancient_civilizations)
4  print('Japan' in ancient_civilizations)
5  print('China' not in ancient_civilizations)
```

   a. What result will the code on line 3 output?

      i. _____

   b. What result will the code on line 4 output?

      i. _____

   c. What result will the code on line 5 output?

      i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Select Operators
Subtopic: Containment

Objectives covered
1 Operations Using Data Types and Operators
1.4 Select operators to achieve the intended results
1.4.6 Containment (in)

Review 1.4

Purpose

Upon completing this project, you will better understand how assignment operators work in Python.

Steps for Completion

1. Open the **146-analyze.py** file from your Domain 1 Student folder.
2. Answer the questions the code has on the following lines:
    a. Line 4: _____
    b. Line 6: _____
    c. Line 9: _____
    d. Line 11: _____

Project Details

Project file
146-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 1
Topic: Select Operators
Subtopic: Containment; Review on 1.4

Objectives covered
1 Operations Using Data Types and Operators
1.4 Select operators to achieve the intended results

Domain 2
Lesson 1

Python v2

LearnKey

Branching Statements: if

Conditions are important parts of decisions and loops that control the flow of a program. An if statement performs an action if a condition is true. For example, an if statement might print a particular message if a person's score in a game is over 10,000.

Purpose

Upon completing this project, you will better understand the purpose of if statements and how to use them in code.

Steps for Completion

1. Review the code, then answer the questions about the if statement.

```
1  score = 12000
2
3  if score > 10000:
4      print("You have reached level 2")
5  print("Thank you for playing")
```

   a. What will be the output of running this code?

      i. _____

   b.

      i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Branching Statements
Subtopic: if

Objectives covered
2 Flow Control with Decisions and Loops
2.1 Construct and analyze code segments that use branching statements
2.1.1 if

LearnKey

Branching Statements: elif

An elif (else if) statement is used when a code segment needs to check for multiple possibilities for conditions and then perform an action based on which condition in a set of conditions is true.

Purpose

Upon completing this project, you will better understand the purpose of elif statements and how to use them to build code.

Steps for Completion

1. Review the code, then answer the questions about the if and elif statements.

```
1  score = 8000
2
3  if score > 10000:
4      print("You have reached level 2")
5  elif score > 5000:
6      print("You have reached level 1")
7
```

   a. What will the output of running this code be?

      i. _____

   b.

      ii. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Branching Statements
Subtopic: elif

Objectives covered
2 Flow Control with Decisions and Loops
2.1 Construct and analyze code segments that use branching statements
2.1.2 elif

LearnKey

Branching Statements: else

The else condition comes into play when neither the if statement nor any of the elif statements are true within a branch of code. For example, a game might need to print a particular statement if none of its defined score goals have been reached.

Purpose

Upon completing this project, you will better understand the purpose of else statements and how to use them within code.

Steps for Completion

1. Review the code, then answer the questions about the branching statements.

```
score = 3000

if score > 10000:
    print("You have reached level 2")
elif score > 5000:
    print("You have reached level 1")
else:
    print("You need to score higher to reach a level")

print("Thank you for playing")
```

   a. What will the output of running this code be?

      i. _____

   b.

      i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Branching Statements
Subtopic: else

Objectives covered
2 Flow Control with Decisions and Loops
2.1 Construct and analyze code segments that use branching statements
2.1.3 else

LearnKey

Nested/Compound Conditions

Often, more complicated statements are needed to create more complex code. For example, in a nested if statement, one if statement goes inside of another and is used if a condition requires checking another. On the other hand, a compound conditional statement contains multiple conditions within the same statement. Programmers might use this type of statement when one of two conditions must be true for a specific code to run.

Purpose

Upon completing this project, you will better understand nested and compound conditions and how to use them within code.

Steps for Completion

1. Open the **214-nested.py** file from your Domain 2 Student folder.
2. What type of statement is on line 10 of the code in this file?
    a. _____
3.
    a. _____
4.
    a. _____

Project Details

Project file
214-nested.py

Estimated completion time
5 minutes

Video reference
Domain 2
- **Topic**: Branching Statements
  - **Subtopic**: Nested and Compound Conditions

Objectives covered
2 Flow Control with Decisions and Loops
- **2.1** Construct and analyze code segments that use branching statements
  - **2.1.4** Nested and compound conditional expressions

LearnKey

Review 2.1

Purpose

Upon completing this project, you will be better equipped to work through logic in code using decisions and loops and be able to determine why a block of code is not producing the desired result.

Steps for Completion

1. Open the **214-analyze.py** file from your Domain 2 Student folder.
2. The anticipation for this code is that it returns the print statement on line 7, as a player with a gold coin who has reached 10000 points should be moved to level 3. Right now, it is returning the print statement on line 13. What two problems with the code are causing this issue?

   a. _____

   b. _____
3. Return to the video to compare your answers to the video's answers.

Project Details

Project file
214-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Branching Statements
Subtopic: Nested and Compound Conditions; Review 2.1

Objectives covered
2 Flow Control with Decisions and Loops
2.1 Construct and analyze code segments that use branching statements

Domain 2
Lesson 2

Python v2

LearnKey

Iteration: while

Iterations, also known as loops, perform an action repeatedly when run. A while loop runs while conditions are true or false. For example, a while loop might run until the score in a game reaches 100 points.

Purpose

Upon completing this project, you will better understand the purpose of while loops and how to use them in code.

Steps for Completion

1. Open the **221-while.py** file from your Domain 2 Student folder.
2. What would happen if the score increase on line 2 was changed to 0?
   a. _____
3. Edit the code so that the output will print a message for having 100 points. Run the code to check your work.
4. Save the file as **221-while-completed**

Project Details

Project file
221-while.py

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Iteration
Subtopic: while

Objectives covered
2 Flow Control with Decisions and Loops
2.2 Construct and analyze code segments that perform iteration
2.2.1 while

LearnKey

Iteration: for

A for loop runs for a predetermined number of iterations based on the length of a collection. As an example, a for loop might iterate over a list or a number of positions.

Purpose

Upon completing this project, you will better understand the purpose of for loops and how to use them in code.

Steps for Completion

1. Open the **222-for.py** file from your Domain 2 Student folder.
2. Run the code. What is the starting number for the positions in the output?
    a. _____
3. Edit the range so that the numbers for the positions in the output will print as positions 1 through 10. Run the code to check your work.
3. Add a comma and the number **2** at the end of the range and run the code. How does this change the output?
    a. _____
    _____
4. Save the file as **222-for-completed**

Project Details

Project file
222-for.py

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Iteration
Subtopic: for

Objectives covered
2 Flow Control with Decisions and Loops
2.2 Construct and analyze code segments that perform iteration
2.2.2 for

LearnKey

Iteration: break and continue

The break keyword is used to exit a loop once a desired point is reached. The continue keyword skips an iteration of a loop.

Purpose

Upon completing this project, you will better understand the purpose of the break and continue keywords and how to use them in code.

Steps for Completion

1. Review the code, then answer the question about the break keyword.

```
1  coins = ('Bronze','Silver','Platinum','Gold')
2  for coin in coins:
3      print ('You possess a', coin, 'coin.')
4      if coin == 'Platinum':
5          print('Congratulations! You move to the next level!')
6          break
```

   a. What is the result of the break keyword in this code?

      i. _____

2. Open the **224-continue.py** file from your Domain 2 Student folder.
3. Edit the code so that the "You possess a Platinum coin" statement does not print as part of the output if the coin is platinum.
4. Save the file as **224-continue-completed**

Project Details

Project file
224-continue.py

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Iteration
Subtopic: break; continue

Objectives covered
2 Flow Control with Decisions and Loops
2.2 Construct and analyze code segments that perform iteration
2.2.3 break
2.2.4 continue

Domain 2
Lesson 3

Python v2

Iteration: pass

The pass keyword allows one to build parts of code while waiting for information. For example, a message as part of a loop may be unknown, but the loop still needs to be tested. The pass keyword can make the code-writing process more efficient by allowing developers to work on other parts of the code, even if the code currently in the file is unfinished.

Purpose

Upon completing this project, you will better understand the purpose of the pass keyword.

Steps for Completion

1. The pass keyword functions as a(n) _____ for code that will be added later.
2. Open the **225-pass.py** file from your Domain 2 Student folder.
3. Edit the code so a placeholder replaces the print statement on line 5.
4. Save the file as **225-pass-completed**

Project Details

Project file
225-pass.py

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Iteration
Subtopic: pass

Objectives covered
2 Flow Control with Decisions and Loops
2.2 Construct and analyze code segments that perform iteration
2.2.5 pass

LK LearnKey

Nested Loops

While loops are used for conditional situations, and for loops are run a set number of times. Developers may sometimes want one loop inside another, known as a nested loop. A nested loop allows developers to run a process inside of another process.

Purpose

Upon completing this project, you will better understand how to use nested loops.

Steps for Completion

1. Open the **226-nested.py** file from your Domain 2 Student folder.
2. Edit the code so the while statement runs on the third life and the first two lives.
3. Edit the code so the positions shown when the code runs are 1 through 10.
4. Save the file as **226-nested-completed**

Project Details

Project file
226-nested.py

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Iteration
Subtopic: Nested Loops

Objectives covered
2 Flow Control with Decisions and Loops
2.2 Construct and analyze code segments that perform iteration
2.2.6 Nested loops

Loops with Compound Conditions

A compound conditional statement is a statement with two or more conditions. Using a compound conditional operator saves time as it prevents one from having to run multiple if statements to get the same results from code.

Purpose

Upon completing this project, you will better understand how to use compound conditional statements.

Steps for Completion

1. Open the **227-compound.py** file from your Domain 2 Student folder.
2. Edit the code so the if statement on line 5 is a compound conditional statement requiring the scepter variable to be True.
3. Save the file as **227-compound-completed**

Project Details

Project file
227-compound.py

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Iteration
Subtopic: Loops with Compound Conditions

Objectives covered
2 Flow Control with Decisions and Loops
2.2 Construct and analyze code segments that perform iteration
2.2.7 Loops that include compound conditional expressions

LK LearnKey

Review 2.2

Purpose

Upon completing this project, you will be able to analyze code with loops better and deduce the number of times a loop will run.

Steps for Completion

1. Open the **227-analyze.py** file from your Domain 2 Student folder.
2. How many print statements will run with the code in its current state?
   a. _____
3. Change the continue statement to a break statement.
4. How many print statements will run with the code in its new state?
   a. _____
5. Return to the video to get the answers to the questions in this review exercise.

Project Details

Project file
227-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 2
Topic: Iteration
Subtopic: Loops with Compound Conditions; Review 2.2

Objectives covered
2 Flow Control with Decisions and Loops
2.2 Construct and analyze code segments that perform iteration

Domain 3
Lesson 1

Python v2

LearnKey

open and close

Knowing how to open a file in Python is important as it allows one to perform various operations on the file. Equally important is understanding how to close files and objects when they are no longer needed. Closing objects frees up memory space, prevents accidental altering of a file, and allows safe modification of files by other programs.

Purpose

Upon completing this project, you will become more familiar with opening and closing an object in Python.

Steps for Completion

1. Review the code, then answer the questions regarding opening and closing a file.

```
311-open.py > ...
1  message = open('311-message.txt','w')
2  message.write('Testing file for player configuration')
3  message.close()
4
5  message_test = open('311-message.txt','r')
6  print('311-message is open')
7
```

   a. On line 5, what does the 'r' indicate?

      i. _____

   b. What will be the output of running this code?

      i. _____

   c. What line of code would you write to close this file?

      i. _____

   d. What are two purposes that parentheses serve in programming languages?

      i. _____

Project Details

Project file
N/A

Estimated completion time
5-10 minutes

Video reference
Domain 3
Topic: File Input and Output
Subtopic: open; close

Objectives covered
3 Input and Output Operations
3.1 Construct and analyze code segments that perform file input and output operations
3.1.1 open
3.1.2 close

read

When developing apps, storing information in files and reading information from files is common. For a file to be read, it must be opened in read mode and its contents saved to a variable.

Purpose

Upon completing this project, you will become more familiar with the read operation.

Steps for Completion

1. Open the **313-read.py** file from your Domain 3 Student folder.
2. On line 7, add code to read the file's contents.
3. Add a statement on the following line to print what has been read.
4. Save the file as **313-read-completed**
5. What is the r+ mode in Python?

   a. _____
   _____

6. When reading a file, why is it necessary to disseminate the file's contents through a print statement?

   a. _____
   _____

Project Details

Project file
313-read.py

Estimated completion time
5 minutes

Video reference
Domain 3
Topic: File Input and Output
Subtopic: read

Objectives covered
3 Input and Output Operations
3.1 Construct and analyze code segments that perform file input and output operations
3.1.3 read

LearnKey

write

The write command in Python allows users to write data to a file and have that information ready for later use within an app or the next time an app runs. Knowing what happens with a programming language when trying to write to a file that exists or does not exist is key to managing files.

Purpose

Upon completing this project, you will better understand the write operation in Python.

Steps for Completion

1. Review the code, then answer the questions regarding writing to a file.

```
314-write.py > ...
1  message = open('314-message.txt','w')
2  message.write('Testing file for player
   configuration')
3  message.write('Testing file for player
   score')
4  message.close()
5
```

   a. What do lines 2 and 3 do?

      i. _____

   b. What should be written at the end of line 2 to create a new line?

      i. _____

   c. What does the 'w' on line 1 indicate?

      i. _____

2. Label the following statements as true or false.

   a. _____ Python requires code to account for a file that does not exist.

   b. _____ The r+ and w+ modes of opening a file are the same.

Project Details

Project file
N/A

Estimated completion time
10 minutes

Video reference
Domain 3
Topic: File Input and Output
Subtopic: write

Objectives covered
3 Input and Output Operations
3.1 Construct and analyze code segments that perform file input and output operations
3.1.4 write

append

While the write command writes to a new file or overwrites an existing one, the append command adds text to the end of an existing file without changing the original content. The append command allows users to add new data to a file without rewriting the entire file each time.

Purpose

Upon completing this project, you will become more familiar with the append command.

Steps for Completion

1. Open the **315-append.py** file from your Domain 3 Student folder.
2. On line 3, add the code needed to indicate that a new line should be formed after the current line of text is written to the file.
3. Copy the open statement on line 1 to line 6.
4. Change the mode on line 6 from write to append.
5. On line 7, add a write statement to add the player name, **Spencer**, to the text file.
6. On line 8, add a statement to close the file.
7. Run the code and verify that the **315-message.txt** document was created.
8. Save the file as **315-append-completed**

Project Details

Project file
315-append.py

Estimated completion time
5-10 minutes

Video reference
Domain 3
Topic: File Input and Output
Subtopic: append

Objectives covered
3 Input and Output Operations
3.1 Construct and analyze code segments that perform file input and output operations
3.1.5 append

Domain 3
Lesson 2

Python v2

LearnKey

Check Existence

Before working on or creating a new file, it's important to check if the file already exists. Checking for a file's existence is especially useful when deciding whether to open it in write or append mode. Users can save time and effort while handling file operations by knowing whether a file exists.

Purpose

Upon completing this project, you will become more familiar with checking for the existence of a file.

Steps for Completion

1. Review the code, then answer the questions about checking for a file's existence.

316-check_existence.py > ...

```
1  import os
2
3  if not os.path.exists('316-message.txt'):
4      message = open('316-message.txt','w')
5      message.write('Testing file for player configuration\n')
6      message.write('Testing file for player score')
7      print("Configuration file made")
8      message.close()
9  else:
10     message_test = open('316-message.txt','r')
11     content = message_test.read()
12     print(content)
13     message_test.close()
```

   a. What does the if statement on line 3 do?

      i. _____

   b. If 316-message.txt does not exist, what do lines 4 through 8 do?

      i. _____

   c. When would lines 10 through 13 run? What would they do?

      i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 3
Topic: File Input and Output
Subtopic: Check Existence

Objectives covered
3 Input and Output Operations
3.1 Construct and analyze code segments that perform file input and output operations
3.1.6 Check existence

LearnKey

delete

Files no longer required should be deleted so that data within them does not pose a security risk. Some potential challenges to the delete operation in Python include the file being open elsewhere, the script not having delete permissions, and corruption or disk errors.

Purpose

Upon completing this project, you will become more familiar with the delete command.

Steps for Completion

1. Review the code, then answer the questions regarding the delete operation.

```
317-delete.py > ...
1  import os
2  message_file = '317-message.txt'
3
4  if os.path.exists(message_file):
5      os.remove(message_file)
6      print("Message file removed")
7
8  else:
9      print("There was no message file to remove")
10
```

   a. What do lines 4 through 6 do?

      i. _____

   b. What does the else statement on lines 8 and 9 do?

      i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 3
Topic: File Input and Output
Subtopic: delete

Objectives covered
3 Input and Output Operations
3.1 Construct and analyze code segments that perform file input and output operations
3.1.7 delete

with Statement

Using a with statement can help reduce the amount of code needed when working with files in Python. This statement opens a file, performs an action on it, and then closes the file without telling Python to do so. By ensuring that a file is properly closed, the memory space it uses can be recovered, which can help improve device performance and reduce the risk of data corruption.

Purpose

In this project, you will practice using a with statement.

Steps for Completion

1. Open the **318-with.py** from your Domain 3 Student folder.
2. Beginning on line 6, use a with statement to rewrite the code from lines 1 through 4.
3. Comment out the code on lines 1 through 4.
4. On line 9, add a print statement that will return **file created**
5. Save the file as **318-with-completed**

Project Details

Project file
318-with.py

Estimated completion time
5 minutes

Video reference
Domain 3
Topic: File Input and Output
Subtopic: with Statement

Objectives covered
3 Input and Output Operations
3.1 Construct and analyze code segments that perform file input and output operations
3.1.8 with statement

Review 3.1

Purpose

Upon completing this project, you will better understand how file input and output work within Python.

Steps for Completion

1. Open the **318-analyze.py** file from your Domain 3 Student folder.
2. Will the code work as written now?

   a. _____

3. If the code will not work, what change needs to be made for the code to work?

   a. _____
   _____

4. Return to the video to see the answer to the question and the change that is needed to allow the code to work as intended.

Project Details

Project file
318-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 3
Topic: File Input and Output
Subtopic: with Statement; Review 3.1

Objectives covered
3 Input and Output Operations
3.1 Construct and analyze code segments that perform file input and output operations

Domain 3
Lesson 3

Python v2

LK LearnKey

Read Input from Console

When developing applications, user input often needs to be captured and stored in a variable for later use. Properly capturing and storing user input ensures that an application will perform the desired actions and provide the expected output.

Purpose

Upon completing this project, you will become more familiar with reading input from console.

Steps for Completion

1. Open the **321-input.py** file from your Domain 3 Student folder.
2. Run the code.
3. Enter your first name into the console.
4. What is the response to entering your name?
    a. _____
5. Adjust the code so there is a space after "Please enter your first name" and a comma after "Welcome to the game."
6. Run the code and enter your first name to verify your changes.
7. Save the file as **321-input-completed**

Project Details

Project file
321-input.py

Estimated completion time
5-10 minutes

Video reference
Domain 3
Topic: Console Input and Output
Subtopic: Read Input from Console

Objectives covered
3 Input and Output Operations
3.2 Construct and analyze code segments that perform console input and output operations
3.2.1 Read input from console

LearnKey

Print Formatted Text

Python provides various methods to control text display, which is especially important when displaying a customized message. Although some methods may be more complicated than others, they all enable users to format text to make the information display more smoothly.

Purpose

Upon completing this project, you will better understand printing formatted text.

Steps for Completion

1. Review the code, then answer the questions regarding printing formatted text.

```
322-format.py > ...
1  score = 2000
2  level = 3
3  player1 = "Tristan"
4
5  print("Player1:", player1, "Score:", score, "Level:", level)
6  print("{} has {} points and has reached level {}".format(player1, score, level))
7
```

   a. On line 6, which method is being used to format the text?

      i. _____

```
322-format.py > ...
1  score = 2000
2  level = 3
3  player1 = "Tristan"
4
5  print("Player1:", player1, "Score:", score, "Level:", level)
6  print("{2} has {0} points and has reached level {1}".format(score, level, player1))
7  print(f"{player1} has {score} points and has reached level {level}")
8
```

   b. On line 6, what are the numbers in curly brackets, and what will they do?

      i. _____
      _____
      _____

   c. On line 7, which formatting method is being used?

      i. _____

2. Which method is the preferred way of formatting strings beginning with Python 3.6?

   a. _____

3. What does the Modulo operator mimic the functionality of in Python?

   a. _____

Project Details

Project file
N/A

Estimated completion time
5-10 minutes

Video reference
Domain 3
Topic: Console Input and Output
Subtopic: Print Formatted Text

Objectives covered
3 Input and Output Operations
3.2 Construct and analyze code segments that perform console input and output operations
3.2.2 Print formatted text (string.format() method, f-String method)

LearnKey

Use Command-Line Arguments

In some situations, a command-line argument can be used to run an app, such as when testing small blocks of code or completing files without being in a development environment. Using command-line arguments allows an app to be run multiple ways without changing the code within the app.

Purpose

Upon completing this project, you will become more familiar with command-line arguments.

Steps for Completion

1. Review the code, then answer the questions regarding command-line arguments.

```
323-command.py > ...
1  message = open('323-message.txt','w')
2  message.write('Testing file for player configuration\n')
3  message.write('Testing file for player score\n')
4  message.close()
5  print("Message file complete")
6
```

   a. What does this code do?

      i. _____

2. Open a command prompt, navigate to your Student folder, and run the code from the command line that will run the **323-command.py** file.
3. Take a screenshot of the result and save the file as **323-command-completed.png**
4. Label the following statements as true or false.
   a. _____ The f-string feature is available on all versions of Python.
   b. _____ One may encounter problems running code from a command line if Python is not set up as an environment variable on their system.

Project Details

Project file
323-command.py

Estimated completion time
5 minutes

Video reference
Domain 3
Topic: Console Input and Output
Subtopic: Use Command-Line Arguments

Objectives covered
3 Input and Output Operations
3.2 Construct and analyze code segments that perform console input and output operations
3.2.3 Use command-line arguments

LK LearnKey

Review 3.2

Purpose

Upon completing this project, you will have created a file from the start, encompassing pieces of what has been covered in the first three domains of this course.

Steps for Completion

1. Create a Python file that will output the following:

   You can get a Rock at level 1.

   You can get a Pogo Stick at level 1.

   You can get a Wand at level 2.

   You can get a Pogo Stick at level 2.

   You can get a Wand at level 3.

   You can get a Rock at level 3.

   You can get a Pogo Stick at level 3.

2. Save the file as **323-items-completed** to your Domain 3 Student folder.

Project Details

Project file
N/A

Estimated completion time
15 minutes

Video reference
Domain 3
Topic: Console Input and Output
Subtopic: Use Command-Line Arguments, First Half Review

Objectives covered
3 Input and Output Operations
3.2 Construct and analyze code segments that perform console input and output operations

Domain 4
Lesson 1

Python v2

LearnKey

Use Indentation

Python uses indentation as part of its code, specifically in any block of code that runs as part of a loop or conditional statement. Indentation is a unique feature of Python, as other programming languages use brackets. Using indentation makes code more readable and may affect code logic.

Purpose

Upon completing this project, you will become more familiar with indentation.

Steps for Completion

1. Review the code, then answer the questions regarding indentation.

411-indentation.py > ...

```
1  game_state = True
2  game_lives = 1
3  while game_lives <= 3:
4      for i in range(1,11):
5          print("You have reached position", i, "in game life", game_lives)
6      if game_state == True:
7          game_lives +=1
8  print("Thank you for playing.")
```

   a. How many times will the print statement on line 8 run? Why?

      i. _____

   b. How can you change line 8 to make the print statement print with every while loop iteration?

      i. _____

2. What is the recommended number of spaces for indentation?

   a. _____

3. Label the following statement as true or false.

   a. _____ Most Python code editors will immediately provide a tooltip telling users when their indentation is inconsistent.

4. Why is it important to use consistent indentation in Python code?

   a. _____

5. Why does Python 3 not allow users to mix tabs and spaces in a single block?

   a. _____

Project Details

Project file
N/A

Estimated completion time
10 minutes

Video reference
Domain 4
Topic: Document Code Segments
Subtopic: Use Indentation

Objectives covered
4 Code Documentation and Structure
4.1 Document code segments
4.1.1 Use indentation

LearnKey

Whitespace

Whitespace refers to the sequence of spaces, tabs, or line breaks between the characters in programming code. The use of whitespace in programming languages is often flexible; however, each language has its own set of best practices for utilizing it effectively. In Python, whitespace has a more significant role than indentation and syntax. It also distinguishes between different code blocks or functions that perform separate tasks. Python's style guide, PEP 8 – Style Guide for Python Code | peps.python.org, provides rules to follow when writing Python code.

Purpose

Upon completing this project, you will better understand whitespace.

Steps for Completion

1. Review the code, then answer the questions regarding whitespace.

412-white_space.py > ...

```
1  game_state = True
2  game_lives = 1
3  while game_lives <= 3:
4      for i in range(1,11):
5          print(f "You have reached position {i} in game life {game_lives})
6      if game_state == True:
7          game_lives +=1
8  print("Thank you for playing.")
```

   a. Where might you add a space on lines 1 and 2 to make them more consistent and readable?

      i. _____

   b. How can you fix the unterminated string error on line 5?

      i. _____

2. According to Python's style guide, PEP 8, how many blank lines should precede function and class definitions?

   a. _____

3. How many blank lines should separate code blocks within a function that carries out different tasks?

   a. _____

Project Details

Project file
N/A

Estimated completion time
5-10 minutes

Video reference
Domain 4
Topic: Document Code Segments
Subtopic: Whitespace

Objectives covered
4 Code Documentation and Structure
4.1 Document code segments
4.1.2 Whitespace

LK LearnKey

Comments

A comment is text that follows a number sign and is either on a line by itself or at the end of a line of code. Python will ignore this text during execution, allowing users to annotate code for better understanding. Comments are marked by a number sign (#) and can either be on a line by themselves or at the end of a line of code. During execution, Python ignores comments, making them helpful in adding explanations or notes to the code without affecting its functionality.

Purpose

In this project, you will practice adding comments to code.

Steps for Completion

1. Open the **413-comments.py** file from your Domain 4 Student folder.
2. Add a comment to line 4 that reads, **#Prints positions achieved during lives**
3. Add a comment to the end of line 6 that reads, **#ranges are zero-based**
4. Add a number sign before the word, print, on line 10.
5. What does adding a number sign do to line 10?
    a. _____
6. Save the file as **413-comments-completed**
7. Label the following statement as true or false.
    a. _____ Each line of code can have more than one inline comment.

Project Details

Project file
413-comments.py

Estimated completion time
5-10 minutes

Video reference
Domain 4
Topic: Document Code Segments
Subtopic: Comments

Objectives covered
4 Code Documentation and Structure
4.1 Document code segments
4.1.3 Comments

LearnKey

Documentation Strings

Documentation strings are also known as docstrings, document functions, classes, modules, and, at times, objects. Docstrings play a critical role in explaining the purpose of an application and the components used within it. Providing documentation helps one easily understand the purpose and code of a program, which makes managing enhancements and troubleshooting much more straightforward. Ideally, docstrings should follow PEP 257 – Docstring Conventions | peps.python.org, Python Enhancement Proposal, which provides docstring writing conventions.

Purpose

In this project, you will practice adding a documentation string to code.

Steps for Completion

1. Open the **414-docstrings.py** file from your Domain 4 Student folder.
2. Add four blank lines to the beginning of the code.
3. On line 1, add a docstring that reads, **"""This code shows items a player can obtain on each level in a game. Notice that the rock is not attainable on level 2."""**
4. Add a print statement on line 15 that reads, **print(_doc_)** and run the code.
5. What does adding the print statement on line 15 do?
    a. _____
6. Save the file as **414-docstrings-completed**
7. What are the two differences between comments and docstrings?
    a. _____
    _____

Project Details

Project file
414-docstrings.py

Estimated completion time
5-10 minutes

Video reference
Domain 4
Topic: Document Code Segments
Subtopic: Documentation Strings

Objectives covered
4 Code Documentation and Structure
4.1 Document code segments
4.1.4 Documentation strings

Use Pydoc for Documentation

Pydoc is a Python documentation generation tool that extracts and organizes information from modules, classes, and functions. Pydoc makes it easier for people to understand modules without running countless searches on the internet for the same information.

Purpose

In this project, you will practice using pydoc to generate documentation.

Steps for Completion

1. Open a command line command prompt.
2. Run a command to get documentation on the built-in datetime module.
3. What does the first line under CLASSES read?
    a. _____
4. Use a keyboard shortcut to end the file.
5. Run a command to send the datetime module documentation to a text file.
6. Take a screenshot of the results.
7. Save the file as **415-pydoc-completed.png** to your Domain 4 Student folder.

Project Details

Project file
Datetime.txt

Estimated completion time
5-10 minutes

Video reference
Domain 4
Topic: Document Code Segments
Subtopic: Use Pydoc for Documentation

Objectives covered
4 Code Documentation and Structure
4.1 Document code segments
4.1.5 Generate documentation by using pydoc

LK LearnKey

Review 4.1

Purpose

Upon completing this project, you will better understand how to add documentation to code files created in Python.

Steps for Completion

1. Open the **415-review.py** file from your Domain 4 Student folder.
2. Add text near the top of the file explaining the purpose of the file.
3. Convert the text you have added to a documentation string.
4. Add a line of code that prints the documentation.
5. Compare your results to the answer in the next video.

Project Details

Project file
415-review.py

Estimated completion time
5 minutes

Video reference
Domain 4
Topic: Document Code Segments
Subtopic: Use Pydoc for Documentation; Review 4.1

Objectives covered
4 Code Documentation and Structure
4.1 Document code segments

Domain 4
Lesson 2

Python v2

LearnKey

Call Signatures

A function is a block of code that can be called multiple times within an app and is best used when a process needs to be repeated. By using functions, developers can avoid repeating the same code repeatedly, making their programs more efficient and easier to manage.

Purpose

Upon completing this project, you will become more familiar with call signatures.

Steps for Completion

1. Review the code, then answer the questions regarding functions and call statements.

```
421-signatures.py > ...
1  def total_score(score, multiplier):
2      return score * multiplier
3
4  print(total_score(3000,1.5))
5  print(total_score(4000,2))
```

   a. On line 4, which number represents the score and which represents the multiplier?

      i. _____

   b. What will be the output from running this code?

      i. _____

   c. What type of statements in this file call the function?

      i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 4
Topic: Function Definitions
Subtopic: Call Signatures

Objectives covered
4 Code Documentation and Structure
4.2 Construct and analyze code segments that include function definitions
4.2.1 Call signatures

LearnKey

Default Values

When writing code in Python, it is possible to set default values for function arguments. These values are utilized when the caller does not provide a value for them. Default values can save time and prevent errors when calling functions repeatedly with the same values.

Purpose

Upon completing this project, you will better understand default values.

Steps for Completion

1. Open the **422-default.py** file from your Domain 4 Student folder.
2. Make **1.5** the default value for the multiplier argument.
3. On line 4, remove the multiplier from the function call.
4. Save the file as **422-default-completed**
5. Review the code, then answer the question regarding the new print statement.

```
422-default.py > ...
1  def total_score(score, multiplier=1.5):
2      return score * multiplier
3
4  print(total_score(3000,))
5  print(total_score(4000,2))
6  print(total_score(multiplier=3, score=15000))
```

   a. What will be the output of running this code?

      i. _____

Project Details

Project file
422-default.py

Estimated completion time
5-10 minutes

Video reference
Domain 4
Topic: Function Definitions
Subtopic: Default Values

Objectives covered
4 Code Documentation and Structure
4.2 Construct and analyze code segments that include function definitions
4.2.2 Default values

LearnKey

Use return

In Python, functions may either return a value or not, depending on the task. When they return a value, it can be saved in a variable and used later for further calculations or other purposes. Thus, functions save time by eliminating the need to write the same code multiple times and provide a convenient way to organize and reuse code.

Purpose

Upon completing this project, you will become more familiar with using the return keyword.

Steps for Completion

1. Review the code, then answer the questions regarding the functions.

```
423-return.py > ...
1  def total_score(score, multiplier):
2      return score * multiplier
3
4  def welcome():
5      print("Welcome to the next level")
6
7  score1 = total_score(3000,2)
8  score2 = total_score(2000,1.5)
9
10 welcome()
11 print(f"Your two scores are {score1} and {score2}.")
12
```

   a. Which function would warrant a value stored for future use, the total score or the welcome message function?

      i. _____

   b. Why does this code need the return statement on line 2?

      i. _____

2. What is a void function?

   a. _____

Project Details

Project file
N/A

Estimated completion time
5-10 minutes

Video reference
Domain 4
Topic: Function Definitions
Subtopic: return

Objectives covered
4 Code Documentation and Structure
4.2 Construct and analyze code segments that include function definitions
4.2.3 return

LearnKey

Use def

When defining a function in Python, the keyword function is not used, as in many other programming languages. Defining a function increases the efficiency of code and helps prevent errors.

Purpose

In this project, you will practice defining a function using the def keyword.

Steps for Completion

1. Open the **424-def.py** file from your Domain 4 Student folder.
2. On line 2, define a function named **calculate_score** and add two arguments in parentheses: **score** and **multiplier**
3. On line 3 add a return statement that returns the **score** times the **multiplier**
4. On line 5, use a print statement to call the **calculate_score** function, making the score **3500** and the multiplier **2**
5. Run the code. What is the output of running this code?
    a. _____
6. Copy the print statement on line 5 to line 6, changing the score to **5500** and the multiplier to **1.6**
7. Run the code. What is the output of running this code?
    a. _____
8. Save the file as **424-def-completed**

Project Details

Project file
424-def.py

Estimated completion time
10 minutes

Video reference
Domain 4
Topic: Function Definitions
Subtopic: def

Objectives covered
4 Code Documentation and Structure
4.2 Construct and analyze code segments that include function definitions
4.2.4 def

LearnKey

Use pass in Functions

The pass keyword acts as a temporary placeholder when testing a function. It is useful when all the necessary information to complete a function's calculation is unavailable, as it allows users to ensure the remaining code works as expected.

Purpose

Upon completing this project, you will become more familiar with using pass in a function.

Steps for Completion

1. Open the **425-pass.py** file from your Domain 4 Student folder.
2. Run the code.
3. Enter the amount **35000** for the first input statement.
4. Enter the amount **1500** for the second input statement.
5. What is the total?
    a. _____
6. What has the pass statement in this code block allowed you to do?
    a. _____
    _____

Project Details

Project file
425-pass.py

Estimated completion time
5 minutes

Video reference
Domain 4
Topic: Function Definitions
Subtopic: Use pass in Functions

Objectives covered
4 Code Documentation and Structure
4.2 Construct and analyze code segments that include function definitions
4.2.5 pass

LearnKey

Review 4.2

Purpose

Upon completing this project, you will better understand how to define and work with functions.

Steps for Completion

1. Open the **425-analyze.py** file from your Domain 4 Student folder.
2. When calling the calculate_total function, must a tax rate be specified?

   a. _____
3. What will be the amounts of the two subtotals when the code runs?

   a. _____
4. Return to the video portion of the course to compare your answers to the ones in the videos.

Project Details

Project file
425-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 4
Topic: Function Definitions
Subtopic: Use pass in Functions; Review 4.2

Objectives covered
4 Code Documentation and Structure
4.2 Construct and analyze code segments that include function definitions

Domain 5
Lesson 1

Python v2

Syntax Errors

Syntax errors in code are similar to punctuation and spelling errors in writing. In programming, syntax errors cause code not to run. Syntax errors prevent code from being compiled and executed. Python's compiler checks for syntax correctness before running any code.

Purpose

Upon completing this project, you will better understand how to recognize syntax errors.

Steps for Completion

1. Open the **511-syntax.py** file from your Domain 5 Student folder.
2. Run the code as many times as needed to help identify the syntax errors within the code.
3. Edit the code to remove the three syntax errors that are present.
4. Save the file as **511-syntax-completed**

Project Details

Project file
511-syntax.py

Estimated completion time
5 minutes

Video reference
Domain 5
Topic: Analyze, Detect, and Fix Errors
Subtopic: Syntax Errors

Objectives covered
5 Troubleshooting and Error Handling
5.1 Analyze, detect, and fix code segments that have errors
5.1.1 Syntax errors

Logic Errors

With a logic error, there is no syntax problem with the code, and the code does not cause a crash, but it does not perform the way one expects it to when the code is run. Logic errors can be difficult to identify because they do not return any error messages.

Purpose

Upon completing this project, you will better understand how to recognize logic errors.

Steps for Completion

1. Open the **512-logic.py** file from your Domain 5 Student folder.
2. Run the code. Does the result match the expected result in the comment on line 5?

   a. _____
3. Edit the code to remove the logic error that is present.
4. Save the file as **512-logic-completed**

Project Details

Project file
512-logic.py

Estimated completion time
5 minutes

Video reference
Domain 5
Topic: Analyze, Detect, and Fix Errors
Subtopic: Logic Errors

Objectives covered
5 Troubleshooting and Error Handling
5.1 Analyze, detect, and fix code segments that have errors
5.1.2 Logic errors

Runtime Errors

Runtime errors occur when one attempts to run code. They occur when one tries to do something with an application that cannot be done as the application's code is being run. Runtime errors can be a result of logic errors.

Purpose

Upon completing this project, you will better understand how to recognize runtime errors.

Steps for Completion

1. Open the **513-runtime.py** file from your Domain 5 Student folder.
2. Run the code to identify the runtime error present in the code.
3. Edit the code to remove the runtime error.
4. Save the file as **513-runtime-completed**

Project Details

Project file
513-runtime.py

Estimated completion time
5 minutes

Video reference
Domain 5
Topic: Analyze, Detect, and Fix Errors
Subtopic: Runtime Errors

Objectives covered
5 Troubleshooting and Error Handling
5.1 Analyze, detect, and fix code segments that have errors
5.1.3 Runtime errors

LearnKey

Review 5.1

Purpose

Upon completing this project, you will better understand how to identify, find, and fix errors in code.

Steps for Completion

1. Open the **513-analyze.py** file from your Domain 5 Student folder.
2. Run the code. Where is there an error, and what type of error is it?
   a. _____
3. What is the error?
   a. _____
4. Fix the error and rerun the code. Are there any other errors?
   a. _____
5. Identify the new error.
   a. _____
   _____
6. Fix the error and rerun the code. Are there any other errors?
   a. _____
7. Return to the videos to compare your answers with those in the videos.

Project Details

Project file
513-analyze.py

Estimated completion time
10 minutes

Video reference
Domain 5
Topic: Analyze, Detect, and Fix Errors
Subtopic: Runtime Errors; Review 5.1

Objectives covered
5 Troubleshooting and Error Handling
5.1 Analyze, detect, and fix code segments that have errors.

Domain 5
Lesson 2

Python v2

LearnKey

Exception Handling: try, except, else, and finally

Many programming languages, Python included, have keywords that can be used to attempt to run code, control the error message displayed if the code would otherwise cause a crash, and then run certain logic depending on whether the original intent to run code worked. Controlling errors and their messages is known as exception handling. Four common exception-handling keywords often used together are try, except, else, and finally.

Purpose

Upon completing this project, you will better understand how to use the try, except, else, and finally keywords to manage exception handling.

Steps for Completion

1. Review the code, then answer the questions regarding the try, except, else, and finally statements in the code.

524-finally.py > ...

```
1  figuratives = ['Simile','Metaphor','Personification','Hyperbole','Allusion']
2  try:
3      figurative_input = int(input('Enter a number from 1-5 to get an example '))
4      figurative = figuratives[figurative_input-1]
5  except:
6      print('You did not enter a figurative. Try again')
7  else:
8      print(f'You chose the {figurative} figurative and will get an example soon.')
9  finally:
10     print('Thank you for playing.')
```

   a. What is the purpose of the try keyword on line 2?

      i. _____

   b. What is the purpose of the except keyword on line 5?

      i. _____

   c. What is the purpose of the else keyword on line 7?

      i. _____

   d. What is the purpose of the finally keyword on line 9?

      i. _____

Project Details

Project file
N/A

Estimated completion time
10 minutes

Video reference
Domain 5
Topic: Exception Handling
Subtopic: try; except; else in Exception Handling; finally

Objectives covered
5 Troubleshooting and Error Handling
5.2 Analyze and construct code segments that handle exceptions
5.2.1 try
5.2.2 except
5.2.3 else
5.2.4 finally

LearnKey

Exception Handling: raise

The raise keyword is another keyword commonly used for exception handling. Raising errors allows a developer to control the type of error to raise and log it if necessary.

Purpose

Upon completing this project, you will better understand how to use the raise keyword to manage exception handling.

Steps for Completion

1. Review the code, then answer the questions regarding the raise keyword in the code.

```
525-raise.py > [e] temperature
1  temperature = float(input("What temperature is absolute zero "))
2  if temperature > 0:
3      raise ValueError("The temperature has to be negative")
4  else:
5      print("The temperature is negative, yes, -459.67 F to be exact.")
```

   a. What will happen if a negative number is entered for the temperature variable?

      i. _____
      _____.

   b. What will happen if a positive number is entered for the temperature variable?

      i. _____
      _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 5
Topic: Exception Handling
Subtopic: raise

Objectives covered
5 Troubleshooting and Error Handling
5.2 Analyze and construct code segments that handle exceptions
5.2.5 raise

LK LearnKey

Review 5.2

Purpose

Upon completing this project, you will better understand the logic flow of exception handling, specifically try-except-else-finally blocks of code.

Steps for Completion

1. Open the **525-analyze.py** file from your Domain 5 Student folder.
2. Which blocks of code will run as the code is currently written?
   a. _____
3. Change the games variable to **2**
4. Which blocks of code will run after the change in the games variable?
   a. _____

Project Details

Project file
525-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 5
Topic: Exception Handling
Subtopic: raise; Review 5.2

Objectives covered
5 Troubleshooting and Error Handling
5.2 Analyze and construct code segments that handle exceptions

Domain 5
Lesson 3

Python v2

LearnKey

Assert Methods

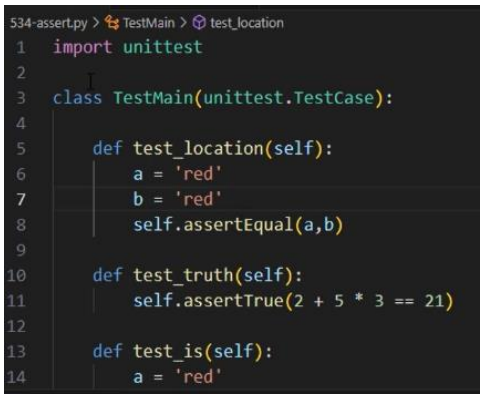
Unit testing is done on a small unit of code to check its functionality. One way of performing unit testing is to use assert methods. Common assert methods include assertEqual, assertTrue, assertIs, assertIn, and assertIsInstance.

Purpose

Upon completing this project, you will better understand assert methods.

Steps for Completion

1. Review the code, then answer the questions regarding the assert methods present in the code.

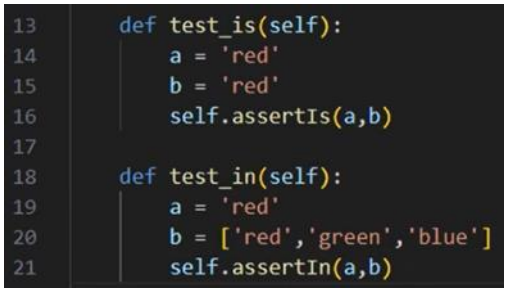


   a. What action does the code on line 8 perform?

      i. _____

   b. What action does the code on line 11 perform?

      i. _____



   c. What action does the code on line 16 perform?

      i. _____

   d. What action does the code on line 21 perform?

      i. _____

Project Details

Project file
N/A

Estimated completion time
10 minutes

Video reference
Domain 5
Topic: Perform Unit Testing
Subtopic: Unittest; Assert Methods

Objectives covered
5 Troubleshooting and Error Handling
5.3 Perform unit testing
5.3.1 Unittest
5.3.4 Assert methods (assertIsInstance, assertEqual, assertTrue, assertIs, assertIn)

Functions and Methods

One of the most common types of unit tests is testing a function to see if it performs as it should. Because classes are often reused in code, it is important that a class's methods be tested for accuracy as well.

Purpose

Upon completing this project, you will better understand the importance of unit testing on functions and methods.

Steps for Completion

1. To test a function or method correctly, one must know what _____ are expected.
2. All code should be tested _____ an application is released to production.
3. _____ are functions that belong to classes.
4. Classes are frameworks for _____.

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 5
Topic: Perform Unit Testing
Subtopic: Functions; Methods

Objectives covered
5 Troubleshooting and Error Handling
5.3 Perform unit testing
5.3.2 Functions
5.3.3 Methods

LK LearnKey

Review 5.3

Purpose

Upon completing this project, you will better understand which assert test to use in a given situation and predict whether a test will pass or fail.

Steps for Completion

1. Open the **534-analyze.py** file from your Domain 5 Student folder.
2. Replace the comment on line 8 with the assert test needed to test whether variable a is contained within variable b.
3. Which test is needed for this testing situation?
   a. _____
4. Will the test pass in this situation?
   a. _____
5. Return to the videos to compare your answers with those found in the videos.

Project Details

Project file
534-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 5
Topic: Perform Unit Testing
Subtopic: Assert Methods; Review 5.3

Objectives covered
5 Troubleshooting and Error Handling
5.3 Perform unit testing

Domain 6
Lesson 1

Python v2

io

Programming languages come with pre-built code libraries that focus on specific functions. In Python, these libraries are called modules and must be imported before they can be used. The io module enables users to store and access data from memory, eliminating the need to work with physical files.

Purpose

Upon completing this project, you will become more familiar with the io module.

Steps for Completion

1. Review the code, then answer the questions regarding the io module.

```
611-io.py > ...
1  import io
2
3  game_stream = io.StringIO()
4
5  game_stream.write("The game has started.\n")
6  game_stream.write("Here is your first question. \n")
7
8  game_stream.seek(1)
9  print(game_stream.read())
```

   a. What does the seek function on line 8 of this code do?

      i. _____

   b. What will be the output of running this code?

      i. _____

2. What is the advantage of using the io module instead of physical files?

   a. _____

3. Label the following statements as true or false.

   a. _____ Bytes IO is a built-in class that only allows in-memory text data storage.

   b. _____ String IO is a built-in io class that allows users to input text strings into memory and retrieve them later.

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 6
Topic: System and Command-Line Operations
Subtopic: io

Objectives covered
6 Operations Using Modules and Tools
6.1 Perform basic file system and command-line operations by using built-in modules
6.1.1 io

OS

Python's os module enables programmers to perform various operating system-based actions. The module offers versatile functions allowing users to rename files, access and manipulate environment variables, and execute shell commands from Python code. By offering the ability to manipulate files within a Python application, the os module eliminates the need to use another source to modify those files, thus simplifying working with them.

Purpose

Upon completing this project, you will better understand the os module.

Steps for Completion

1. Review the code, then answer the questions regarding the os module.

```
612-os.py > ...
1  import os
2
3  print("Your current directory is:", os.getcwd())
4
5  for text_file in os.listdir():
6      if text_file.endswith('.txt'):
7          print(text_file)
```

   a. What does line 3 do?

      i. _____

   b. What does os.listdir do in line 5?

      i. _____

2. Open the **612-os.py** file from your Domain 6 Student folder.
3. Run the code and observe the list of files within the directory.
4. Add code on line 9 to rename the **601-message.txt** file **OLD601-message.txt**
5. Cut the for loop and paste it after the rename code.
6. Run the code to ensure the 601-message.txt file is renamed.
7. Save the file as **612-os-completed**

Project Details

Project file
612-os.py

Estimated completion time
5-10 minutes

Video reference
Domain 6
Topic: System and Command-Line Operations
Subtopic: os

Objectives covered
6 Operations Using Modules and Tools
6.1 Perform basic file system and command-line operations by using built-in modules
6.1.2 os

LK LearnKey

os.path

The os.path sub-module provides methods to check for the existence of a file or directory in a specific location. The ability to do this is beneficial when an application depends on particular directory structures and needs to ensure they exist.

Purpose

In this project, you will practice using the os.path submodule.

Steps for Completion

1. Open the **613-ospath.py** file from your Domain 6 Student folder.
2. Change line 1 to limit this import to only the os.path submodule.
3. On line 15, as part of the else block of code, write **file_path os.path.join('613-test' , '613-test.txt')**
4. On line 16, add code that checks if the path on line 15 is a legitimate file path.
5. On line 17, add a print statement to confirm the file exists.
6. Run the code. Does the file exist? How do you know?
    a. _____
7. Save the file as **613-ospath-completed**

Project Details

Project file
613-ospath.py

Estimated completion time
5-10 minutes

Video reference
Domain 6
Topic: System and Command-Line Operations
Subtopic: os.path

Objectives covered
6 Operations Using Modules and Tools
6.1 Perform basic file system and command-line operations by using built-in modules
6.1.3 os.path

LearnKey

sys

The sys module provides access to some of the system-specific functionalities in Python. It allows users to import other modules, manage files, and handle command-line arguments. It also offers access to system-based commands like changing file paths and terminating an application.

Purpose

Upon completing this project, you will better understand the sys module.

Steps for Completion

1. Review the code, then answer the questions regarding the sys module.

```
614-sys.py > ...
1  import sys
2
3  game_lives = 1
4  while game_lives < 3:
5      print("Game in progress")
6      game_lives +=1
7  sys.exit()
```

a. What does line 7 of this code do?

i. _____

```
614-command.py > ...
1  import sys
2
3  filename = sys.argv[1]
4  print (f" {filename} has been specified.")
5
6  with open (filename, 'r') as file:
7      content = file.read()
8      print(content)
```

b. What is sys.argv[1] on line 3?

i. _____

c. Where do you run this file to ensure the sys.argv command works?

i. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 6
Topic: System and Command-Line Operations
Subtopic: sys

Objectives covered
6 Operations Using Modules and Tools
6.1 Perform basic file system and command-line operations by using built-in modules
6.1.4 sys (importing modules, opening, reading and writing files, command-line arguments)

LK LearnKey

Review 6.1

Purpose

Upon completing this project, you will become more familiar with identifying and importing Python libraries to code, and using methods from these built-in libraries.

Steps for Completion

1. Open the **614-analyze.py** file from your Domain 6 Student folder.
2. Complete the import statement on line 1 to import the module that works with input and output streams.
3. Complete the code on line 3 to use the method needed to enable streaming text to memory.
4. Return to the videos to compare your answers to the answers in the videos.

Project Details

Project file
614-analyze.py

Estimated completion time
5 minutes

Video reference
Domain 6
Topic: System and Command-Line Operations
Subtopic: sys; Review 6.1

Objectives covered
6 Operations using Modules and Tools
6.1 Perform basic file system and command-line operations by using built-in modules

Domain 6
Lesson 2

Python v2

LearnKey

Math

Python's Math module is a comprehensive computational category that provides various math functions such as fabs, ceil, floor, trunc, fmod, frexp, and nan. All these functions are readily available in Python's standard library, eliminating the need for additional installations.

Purpose

Upon completing this project, you will become more familiar with the math submodule.

Steps for Completion

1. Match the math function to its description.

A. fabs	B. ceil	C. floor
D. fmod	E. frexp	

   a. _____ Returns the next integer up after a decimal number.

   b. _____ Returns a number's absolute distance from zero.

   c. _____ Returns an integer without a decimal number.

   d. _____ Returns the modulus, also known as a remainder, from dividing two numbers in floating decimal format.

   e. _____ Returns the next integer up after a decimal number.

2. Which math function is similar to ceil and floor?

   a. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 6
Topic: Solve Complex Problems
Subtopic: Math

Objectives covered
6 Operations Using Modules and Tools
6.2 Solve complex computing problems by using built-in modules
6.2.1 Math (fabs, ceil, floor, trunc, fmod, frexp, nan, isnan, sqrt, isqrt, pow, pi)

isnan, sqrt, isqrt, and pi

The Math module helps solve complex math problems related to statistics, geometry, trigonometry, and others without writing much code. These math functions can help with gaming apps, as many have math or statistics-based functionality. Some additional math functions include nan, isnan, sqrt, isqrt, and pi.

Purpose

Upon completing this project, you will become more familiar with the isnan, sqrt, isqrt, and pi math functions.

Steps for Completion

1. What does the isnan function do?
   a. _____
   _____
   _____
2. Label the following statement as true or false.
   a. _____ nan is a result, not a function.
3. What does isqrt return?
   a. _____
4. What will this print statement return if y = 3 and x = 4?

   print(math.pow(y,x)

   a. _____
5. What will this print statement return if x = 3?

   print(math.pi * math.pow(x, 2))

   a. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 6
Topic: Solve Complex Problems
Subtopic: isnan, sqrt, isqrt, and pi

Objectives covered
6 Operations Using Modules and Tools
6.2 Solve complex computing problems by using built-in modules
6.2.1 Math (fabs, ceil, floor, trunc, fmod, frexp, nan, isnan, sqrt, isqrt, pow, pi)

LearnKey

datetime

Every programming language, including Python, offers several ways to format dates and times. Some apps use fixed dates, while others require the current date, such as when a game logs the current date and time as the starting point for a game based on timing.

Purpose

In this project, you will practice working with the datetime module.

Steps for Completion

1. Open the **622-datetime.py** file from your Domain 6 Student folder.
2. On line 2, delete the comment and add code to show the current date and time.
3. On line 3, delete the comment and add code to change the current date and time format to mm-dd-yy.
4. Run the code. What is the result?

    a. _____

    _____

5. On line 3, replace the comment with the code required to show the current date and time format as mm-dd-yyyy hh:mm.
6. On line 4, add a print statement to return the number of the weekday.
7. On line 5, add a print statement to return the name of the weekday.
8. Save the file as **622-datetime-completed**

Project Details

Project file
622-datetime.py

Estimated completion time
5-10 minutes

Video reference
Domain 6
Topic: Solve Complex Problems
Subtopic: datetime

Objectives covered
6 Operations Using Modules and Tools
6.2 Solve complex computing problems by using built-in modules
6.2.2 datetime(now, strftime, weekday)

Domain 6
Lesson 3

Python v2

LearnKey

Random with Numbers

The random module offers multiple methods to generate random numbers or values from a given list of values. In gaming, random numbers play a significant role as many games need different events to happen each time someone plays those games to keep them fun and interesting.

Purpose

Upon completing this project, you will become more familiar with using the random module with numbers.

Steps for Completion

1. Review the code, then answer the questions regarding the random module.

```
623-random.py > ...
1  from random import randrange
2  for i in range(10):
3      print(randrange(5,20))
```

   a. What will running this code return?

      i. _____

   b. What can be added to the randrange method on line 3 to return odd numbers only?

      i. _____

   c. How can you make this return a list of even numbers?

      i. _____

```
623-random.py > ...
1  from random import randrange
2  for i in range(10):
3      print(randint(1,20))
```

   d. What will happen if you run this code?

      i. _____

   e. How can you fix this code?

      i. _____

2. What does the random method return?

   a. _____

Project Details

Project file
N/A

Estimated completion time
10 minutes

Video reference
Domain 6
Topic: Solve Complex Problems
Subtopic: Random with Numbers

Objectives covered
6 Operations Using Modules and Tools
6.2 Solve complex computing problems by using built-in modules
6.2.3 random (randrange, randint, random, shuffle, choice, sample)

LearnKey

Random with Lists

The random module is not just for use with numbers; it can also be used with words and phrases. Several random modules are available to use with lists, such as shuffle, choice, and sample.

Purpose

Upon completing this project, you will become more familiar with using the random module with lists.

Steps for Completion

1. Match the random module with its description.

A. shuffle	B. choice	C. sample

   a. _____ Returns multiple random values from a list.

   b. _____ Rearranges the order of a list.

   c. _____ Chooses a value, at random, from a list of values.

2. Which random module should you use to ensure you get two unique values returned from a list?

   a. _____

Project Details

Project file
N/A

Estimated completion time
5 minutes

Video reference
Domain 6
Topic: Solve Complex Problems
Subtopic: Random with Lists

Objectives covered
6 Operations Using Modules and Tools
6.2 Solve complex computing problems by using built-in modules
6.2.3 random (randrange, randint, random, shuffle, choice, sample)

Review 6.2

Purpose

Upon completing this project, you will better understand how to build a small app that calculates using random numbers.

Steps for Completion

1. Build a Python app that does the following:
    a. Captures two whole numbers via inputs from a user
    b. Performs a calculation of those two numbers. The calculation itself is random, with a random operator of a plus sign, minus sign, or multiplication sign.
    c. The results should print in this format: 3 + 5 = 8
2. Save the file as **623-final** to your Domain 6 Student folder.

Project Details

Project file
N/A

Estimated completion time
15 minutes

Video reference
Domain 6
Topic: Solve Complex Problems
Subtopic: Random with Lists; Final Review

Objectives covered
6 Operations using Modules and Tools
6.2 Solve complex problems by using built-in modules

Appendix

Python v2

LK LearnKey

Glossary

Domain 1

Term	Definition
AND	A logical keyword where both sides of the comparison must be true in order for the statement to be true.
Append	An act that adds items to the end of a list or writes text to the end of a file.
Arithmetic Operator	An operator used to perform basic mathematical operations.
Assignment	An event that takes place when a variable is set to be equal to a string, number, Boolean operator, list, or date.
Boolean Variable	A type of variable with two possible values: true or false. A Boolean variable is often used to determine a course of action within a program.
Comparison Operator	An operator that is used inside Python to compare arguments. Comparison operators include less than, less than or equal to, greater than, or greater than or equal to.
Containment Operator	An operator that determines if a value is contained in a set of values.
Dictionary	A list of data that stores values in key-value pairs.
Float Variable	A type of number variable that uses a decimal as part of its storage.
Identity Operator	An operator in Python that is used to determine if two variables share the same memory space.
Immutable	In the context of variables, a type of variable that cannot be changed without redefining the variable.
Index	The position in which an item is in a list or a character is within a set of characters. The first index value is always 0.
Insert	In the context of lists, the act of adding an item to a specified spot within the list.
Integer Variable	A type of variable that stores whole numbers.
List	A variable that stores a collection of items in a defined order.
Logical Operator	An operator that compares two conditions using keywords and, or, or not.
NOT	The logical keyword that takes a condition set by the user and checks for its opposite.
OR	A logical keyword where only one of the two possibilities needs to be true in an argument for the argument to be true.
Remove	In the context of lists, the act of deleting an item from a list.
Set	A list of data that stores unique values of data.
Slicing	A process that extracts characters from a list, word, or phrase.
String Variable	A type of variable in Python that is used to store items and information.
Tuple	A list of data with immutable values.
Variable	A container that stores information that can be used elsewhere in code.

LK LearnKey

Domain 2

Term	Definition
Break Statement	A statement used to exit a loop of code.
Compound Conditional	A type of statement that has two or more conditions separated by logical operators.
Continue Statement	A statement used to skip running the rest of an iteration within a loop.
Elif Statement	A conditional statement that checks to see if a condition is true based on an if statement's condition not being true.
Else Statement	A statement used in conjunction with if and elif statements. An else statement provides an action in case no if or elif conditions are true.
For Loop	A loop that runs a block of code a set number of times.
If Statement	A statement used to check to see if a condition is true, and, if it is, causes a specific action to be performed.
Nested If	An if statement that is placed inside of another if statement.
Nested Loop	A loop that is placed inside of another loop.
Pass Keyword	A keyword that is used as a placeholder for code to be filled in later. The Pass keyword is often used as part of a loop.
While Loop	A loop that runs a block of code so long as a given condition is true.

Domain 3

Term	Definition
Append Function	In the context of file management, a built-in function that adds content to an existing file.
Close Function	In the context of file management, a built-in function that closes a file and frees the memory space the file occupies.
Command-Line Argument	A value used to control how a block of code runs via the command line.
Exists Function	In the context of file management, a built-in function that checks to see if a file or folder exists.
Open Function	A built-in function that opens a file, usually for reading purposes.
Read Function	In the context of file management, a built-in function that reads the contents of a file or stream.
Remove Function	In the context of file management, a built-in function that deletes a file from a folder.
string.format	A method used to format strings of text and place variables in precise positions within text.
With Statement	A statement block that supports file operations and automatically closes a file when the block of code is complete.
Write Function	In the context of file management, a built-in function that writes content to a new file or overwrites an existing file.

LK LearnKey

Domain 4

Term	Definition
Call Signature	A line of code that calls a function and, when applicable, provides needed arguments for said function.
Comment	A symbol used to get an interpreter to ignore a line of code. Comments are often used to explain code.
Def Keyword	A keyword used in Python to begin and define a function.
Default Value	A value or item in a function that is set to be constant but can be overridden.
Docstring	Also known as a documentation string, a block of code, surrounded by triple quotation marks, that is used to explain what code is doing.
Indentation	The formatting of code in which a block of code knows which statements are part of that block, such as actions within an if statement.
Pydoc	A feature in Python that acts as a library module. It automatically generates documentation on Python modules or keywords.
Return Keyword	A keyword which defines the end of a function and holds a value to be used outside of the function.
White Space	Empty lines of code found in Python. It can be utilized to make code easier to read.

Domain 5

Term	Definition
Assert Test	A type of unit test that compares two values and returns a True or False based on the result of the comparison.
assertEqual	An assert test that determines whether two values are equal to each other.
assertIn	An assert test that determines whether a value is contained in a set of values.
assertIs	An assert test that determines whether two variables share the same memory space.
assertIsInstance	An assert test that determines whether a variable is an instance of a class.
assertTrue	An assert test that determines whether a statement is true.
Class	A framework for a group of objects and methods that help define those objects.
Else	In the context of exception handling, the code block that runs if a Try block succeeds.
Except	In the context of exception handling, the code block that runs if a Try block fails.
Finally Keyword	A block of code that runs regardless of whether a try block was successful.
Function	A block of code that can be called upon multiple times within a program.
Logic Error	A type of error in Python that occurs when an argument, calculation, or flow is set incorrectly.
Method	A function that belongs to a class.
Object	A tangible asset used as a building block for an app.
Raise	A type of keyword that is used to display an error message when an exception occurs.
Runtime Error	An error type that only occurs when an app runs and often results in the app crashing.
Syntax Error	An error in programming code that is caused either by a typo or by trying to use an item incorrectly.
Try	A block of code that attempts to run without any errors being generated.
Unit Test	A test that is done on a small unit of code to see if the code works as intended.

Term	Definition
ceil	A math function that returns the next integer up from a decimal number.
choice	A random function that returns one value from a list of values.
Datetime Module	A module that provides functionality for dates, times, and date and time formats.
fabs	A math function that measures the absolute distance from zero and is presented as a floating decimal.
floor	A math function that returns an integer portion of a number.
fmod	A math function that returns the modulus for two numbers and is presented as a floating decimal.
frexp	A math function that returns the mantissa and exponent of a number.
Io Module	A module that provides functionality for file management.
isnan	A math function that checks to see if a value is a number and returns False if it is a number and True if it is not a number.
isqrt	A math function that returns the integer of a square root of a number.
Math Module	A module that provides a means to multiple types of calculations.
nan	An expression that signifies a value is not a number.
now	A date function that returns the current date and time.
Os Module	The operating system (os) module allows the user to perform operating system tasks, such as create a folder.
Os.path	A piece of the os module, the os.path is used to help users to find files and folders on specific paths.
pi	A math function that returns pi (3.14159).
pow	A math function that returns one number raised to the power of another number.
randint	A random function that returns an integer between a starting value and an ending value.
random	A function that returns a random number between 0 and 1.
Random Library	A library that contains many random number and list-related functions.
randrange	A random function that returns a number between a starting value and one less than the ending value.
sample	A random function that returns a set number of values from a list of values.
shuffle	A random function that reorders a list of values randomly.
sqrt	A math function that returns the square root of a number.
strftime	A date function that formats a date and time as a string, with symbols used to manipulate the format of the string.
Sys Module	A module used to import built-in system functions that can be used in Python.
trunc	A math function that returns the next highest positive integer or the next lowest negative integer for a number.
weekday	A date function that returns the weekday number for a date, with 1 being Monday.

LK LearnKey

Objectives

Python v2 Objectives		
Domain 1 Operations Using Data Types and Operators	**Domain 2 Flow Control with Decisions and Loops**	**Domain 3 Input and Output Operations**
1.1 Evaluate expressions to identify the data types Python assigns to variables 1.1.1 str 1.1.2 int 1.1.3 float 1.1.4 bool	2.1 Construct and analyze code segments that use branching statements 2.1.1 if 2.1.2 elif 2.1.3 else 2.1.4 Nested and compound conditional expressions	3.1 Construct and analyze code segments that perform file input and output operations 3.1.1 open 3.1.2 close 3.1.3 read 3.1.4 write 3.1.5 append 3.1.6 Check existence 3.1.7 delete 3.1.8 with statement
1.2 Perform and analyze data and data type operations 1.2.1 Data type conversion 1.2.2 Indexing 1.2.3 Slicing 1.2.4 Construct data structures 1.2.5 Lists 1.2.6 List operations	2.2 Construct and analyze code segments that perform iteration 2.2.1 while 2.2.2 for 2.2.3 break 2.2.4 continue 2.2.5 pass 2.2.6 Nested loops 2.2.7 Loops that include compound conditional expressions	3.2 Construct and analyze code segments that perform console input and output operations 3.2.1 Read input from console 3.2.2 Print formatted text (string.format() method, f-string method) 3.2.3 Use command-line arguments
1.3 Determine the sequence of execution based on operator precedence 1.3.1 Assignment 1.3.2 Comparison 1.3.3 Logical 1.3.4 Arithmetic 1.3.5 Identity (is) 1.3.6 Containment (in)		
1.4 Select operators to achieve the intended results 1.4.1 Assignment 1.4.2 Comparison 1.4.3 Logical 1.4.4 Arithmetic 1.4.5 Identity (is) 1.4.6 Containment (in)		

Domain 4 Code Documentation and Structure	Domain 5 Troubleshooting and Error Handling	Domain 6 Operations Using Modules and Tools
4.1 Document code segments 4.1.1 Use indentation 4.1.2 Whitespace 4.1.3 Comments 4.1.4 Documentation strings 4.1.5 Generate documentation by using pydoc	5.1 Analyze, detect, and fix code segments that have errors 5.1.1 Syntax errors 5.1.2 Logic errors 5.1.3 Runtime errors	6.1 Perform basic file system and command-line operations by using built-in modules 6.1.1 io 6.1.2 os 6.1.3 os.path 6.1.4 sys (importing modules, opening, reading and writing files, command-line arguments)
4.2 Construct and analyze code segments that include function definitions 4.2.1 Call signatures 4.2.2 Default values 4.2.3 return 4.2.4 def 4.2.5 pass	5.2 Analyze and construct code segments that handle exceptions 5.2.1 try 5.2.2 except 5.2.3 else 5.2.4 finally 5.2.5 raise	6.2 Solve complex computing problems by using built-in modules 6.2.1 Math (fabs, ceil, floor, trunc, fmod, frexp, nan, isnan, sqrt, isqrt, pow, pi) 6.2.2 datetime (now, strftime, weekday) 6.2.3 random (randrange, randint, random, shuffle, choice, sample)
	5.3 Perform unit testing 5.3.1 Unittest 5.3.2 Functions 5.3.3 Methods 5.3.4 Assert methods (assertIsInstance, assertEqual, assertTrue, assertIs, assertIn)	

Lesson Plan

Approximately 34.5 hours of videos,
application questions, and projects

Python v2

LK LearnKey

Domain 1 Lesson Plan

Domain 1 - Operations Using Data Types and Operators [approximately 9.5 hours of videos, application questions, and projects]				
Lesson	**Lesson Topic and Subtopics**	**Objectives**	**Practical Application**	**Workbook Projects and Files**
Pre-Assessment Assessment time - 00:30:00	Operations Using Data Types and Operators: Pre-Assessment			
Lesson 1 Video time - 00:08:29 Practical application time - 00:00:00 Workbook time - 00:20:00	**Identify Data Types Part 1** How to Study for This Exam What Is Python? What Needs to Be Installed	1.1 Evaluate expressions to identify the data types Python assigns to variables	N/A	Python Introduction – pg. 7 N/A Installing Python – pg. 8 Domain 1 Student folder
Lesson 2 Video time - 00:10:14 Practical application time - 00:24:00 Workbook time - 00:35:00	**Identify Data Types Part 2** str int float bool Review on 1.1	1.1.1 str 1.1.2 int 1.1.3 float 1.1.4 bool	D1Q1-D1Q6 (6 questions)	Strings and Integers – pg. 10 111-str.py 112-numbers.py Floats and Bools – pg. 11 113-numbers.py 114-boolean.py Review 1.1 – pg. 12 114-analyze.py
Lesson 3 Video time - 00:09:33 Practical application time - 00:16:00 Workbook time - 00:25:00	**Analyze Data Types and Operators Part 1** Data Type Conversion Indexing	1.2 Perform and analyze data and data type operations 1.2.1 Data type conversion 1.2.2 Indexing	D1Q7-D1Q10 (4 questions)	Data Type Conversion – pg. 14 121-conversion.py Indexing – pg. 15 122-indexing.py
Lesson 4 Video time - 00:8:44 Practical application time - 00:16:00 Workbook time - 00:15:00	**Analyze Data Types and Operators Part 2** Slicing Construct Data Structures	1.2.3 Slicing 1.2.4 Construct data structures	D1Q11-D1Q14 (4 questions)	Slicing – pg. 17 123-slicing.py Data Structures – pg. 18 N/A
Lesson 5 Video time - 00:13:47 Practical application time - 00:20:00	**Analyze Data Types and Operators Part 3** Lists List Operations Review on 1.2 Part 1 Review on 1.2 Part 2	1.2.5 Lists 1.2.6 List operations	D1Q15-D1Q19 (5 questions)	Lists and Their Operations – pg. 20 126-list_operations.py Review 1.2 – pg. 21 126-list_operations.py 126-analyze.py

Domain 1 - Operations Using Data Types and Operators [approximately 9.5 hours of videos, application questions, and projects]				
Lesson	**Lesson Topic and Subtopics**	**Objectives**	**Practical Application**	**Workbook Projects and Files**
Workbook time - 00:20:00				
Lesson 6 Video time - 00:07:55 Practical application time - 00:24:00 Workbook time - 00:15:00	**Sequence of Execution Part 1** Assignment Order Comparison Order Logical Order	1.3 Determine the sequence of execution based on operator precedence 1.3.1 Assignment 1.3.2 Comparison 1.3.3 Logical	D1Q20-D1Q25 (6 questions)	Assignment Operators – pg. 23 N/A Comparison Operators – pg. 24 N/A Logical Operators – pg. 25 N/A
Lesson 7 Video time - 00:10:10 Practical application time - 00:20:00 Workbook time - 00:35:00	**Sequence of Execution Part 2** Arithmetic Order Identity Order Containment Order Review on 1.3	1.3.4 Arithmetic 1.3.5 Identity (is) 1.3.6 Containment (in)	D1Q26-D1Q30 (5 questions)	Arithmetic – pg. 27 134-arithmetic.py Identity Operator – pg. 28 N/A Containment Operator – pg. 29 N/A Review 1.3 – pg. 30 N/A
Lesson 8 Video time - 00:12:35 Practical application time - 00:24:00 Workbook time - 00:20:00	**Select Operators Part 1** Assignment Comparison Logical	1.4 Select operators to achieve the intended results 1.4.1 Assignment 1.4.2 Comparison 1.4.3 Logical	D1Q31-D1Q36 (6 questions)	Using Assignment Operators – pg. 32 N/A Using Comparison Operators – pg. 33 N/A Using Logical Operators – pg. 34 N/A
Lesson 9 Video time - 00:12:15 Practical application time - 00:24:00 Workbook time - 00:25:00	**Select Operators Part 2** Arithmetic Identity Containment Review on 1.4	1.4.4 Arithmetic 1.4.5 Identity (is) 1.4.6 Containment (in)	D1Q37-D1Q42 (6 questions)	Using Arithmetic Operators – pg. 36 N/A Using the Identity Operator – pg. 37 N/A Using the Containment Operator – pg. 38 N/A Review 1.4 – pg. 39 146-analyze.py
Post-Assessment Assessment time - 01:00:00	Operations Using Data Types and Operators: Post-Assessment			

LK LearnKey

Domain 2 Lesson Plan

Domain 2 - Flow Control with Decisions and Loops [approximately 4.5 hours of videos, application questions, and projects]				
Lesson	**Lesson Topic and Subtopics**	**Objectives**	**Practical Application**	**Workbook Projects and Files**
Pre-Assessment Assessment time - 00:30:00	Flow Control with Decisions and Loops: Pre-Assessment			
Lesson 1 Video time - 00:16:04 Practical application time - 00:36:00 Workbook time - 00:25:00	**Branching Statements** if elif else Nested and Compound Conditions Review 2.1	2.1 Construct and analyze code segments that use branching statements 2.1.1 if 2.1.2 elif 2.1.3 else 2.1.4 Nested and compound conditional expressions	D2Q1-D2Q9 (9 questions)	Branching Statements: if – pg. 41 N/A Branching Statements: elif – pg. 42 N/A Branching Statements: else – pg. 43 N/A Nested/Compound Conditions – pg. 44 214-nested.py Review 2.1 – pg. 45 214-analyze.py
Lesson 2 Video time - 00:11:45 Practical application time - 00:36:00 Workbook time - 00:15:00	**Iteration Part 1** while for break continue	2.2 Construct and analyze code segments that perform iteration 2.2.1 while 2.2.2 for 2.2.3 break 2.2.4 continue	D2Q10-D2Q18 (9 questions)	Iteration: while – pg. 47 221-while.py Iteration: for – pg. 48 222-for.py Iteration: break and continue – pg. 49 224-continue.py
Lesson 3 Video time - 00:08:21 Practical application time - 00:16:00 Workbook time - 00:20:00	**Iteration Part 2** pass Nested Loops Loops with Compound Conditions Review 2.2	2.2.5 pass 2.2.6 Nested loops 2.2.7 Loops that include compound conditional expressions	D2Q19-D2Q22 (4 questions)	Iteration: pass – pg. 51 225-pass.py Nested Loops – pg. 52 226-nested.py Loops with Compound Conditions – pg. 53 227-compound.py Review 2.2 – pg. 54 227-analyze.py
Post-Assessment Assessment time - 01:00:00	Flow Control with Decisions and Loops: Post-Assessment			

LearnKey

Domain 3 Lesson Plan

Domain 3 - Input and Output Operations [approximately 5 hours of videos, application questions, and projects]				
Lesson	**Lesson Topic and Subtopics**	**Objectives**	**Practical Application**	**Workbook Projects and Files**
Pre-Assessment Assessment time - 00:30:00	Input and Output Operations: Pre-Assessment			
Lesson 1 Video time - 00:13:17 Practical application time - 00:40:00 Workbook time - 00:35:00	**File Input and Output Part 1** open close read write append	3.1 Construct and analyze code segments that perform file input and output operations 3.1.1 open 3.1.2 close 3.1.3 read 3.1.4 write 3.1.5 append	D3Q1-D3Q10 (10 questions)	open and close – pg. 56 N/A read – pg. 57 313-read.py write – pg. 58 N/A append – pg. 59 315-append.py
Lesson 2 Video time - 00:09:18 Practical application time - 00:24:00 Workbook time - 00:20:00	**File Input and Output Part 2** Check Existence delete with Statement Review 3.1	3.1.6 Check existence 3.1.7 delete 3.1.8 with statement	D3Q11-D3Q16 (6 questions)	Check Existence – pg. 61 N/A delete – pg. 62 N/A with Statement – pg. 63 318-with.py Review 3.1 – pg. 64 318-analyze.py
Lesson 3 Video time - 00:14:43 Practical application time - 00:24:00 Workbook time - 00:35:00	**Console Input and Output** Read Input from Console Print Formatted Text Use Command-Line Arguments First Half Review	3.2 Construct and analyze code segments that perform console input and output operations 3.2.1 Read input from console 3.2.2 Print formatted text (string.format() method, f-String method) 3.2.3 Use command-line arguments	D3Q17-D3Q22 (6 questions)	Read Input from Console – pg. 66 321-input.py Print Formatted Text – pg. 67 N/A Use Command-Line Arguments – pg. 68 323-command.py Review 3.2 – pg. 69 N/A
Post-Assessment Assessment time - 01:00:00	Input and Output Operations: Post-Assessment			

LK LearnKey

Domain 4 Lesson Plan

Domain 4 - Code Documentation and Structure [approximately 5 hours of videos, application questions, and projects]				
Lesson	**Lesson Topic and Subtopics**	**Objectives**	**Practical Application**	**Workbook Projects and Files**
Pre-Assessment Assessment time - 00:30:00	Code Documentation and Structure: Pre-Assessment			
Lesson 1 Video time - 00:14:35 Practical application time - 00:40:00 Workbook time - 00:55:00	**Document Code Segments** Use Indentation Whitespace Comments Documentation Strings Use Pydoc for Documentation Review 4.1	4.1 Document code segments 4.1.1 Use indentation 4.1.2 Whitespace 4.1.3 Comments 4.1.4 Documentation strings 4.1.5 Generate documentation by using pydoc	D4Q1-D4Q10 (10 questions)	Use Indentation – pg. 71 N/A Whitespace – pg. 72 N/A Comments – pg. 73 413-comments.py Documentation Strings – pg. 74 414-docstrings.py Use Pydoc for Documentation – pg. 75 Datetime.txt Review 4.1 – pg. 76 415-review.py
Lesson 2 Video time - 00:13:10 Practical application time - 00:40:00 Workbook time - 00:45:00	**Function Definitions** Call Signatures Default Values return def Use pass in Functions Review 4.2	4.2 Construct and analyze code segments that include function definitions 4.2.1 Call signatures 4.2.2 Default values 4.2.3 return 4.2.4 def 4.2.5 pass	D4Q11-D4Q20 (10 questions)	Call Signatures – pg. 78 N/A Default Values – pg. 79 422-default.py Use return – pg. 80 N/A Use def – pg. 81 424-def.py Use pass in Functions – pg. 82 425-pass.py Review 4.2 – pg. 83 425-analyze.py
Post-Assessment Assessment time - 01:00:00	Code Documentation and Structure: Post-Assessment			

LK LearnKey

Domain 5 Lesson Plan

Domain 5 - Troubleshooting and Error Handling [approximately 5 hours of videos, application questions, and projects]				
Lesson	**Lesson Topic and Subtopics**	**Objectives**	**Practical Application**	**Workbook Projects and Files**
Pre-Assessment Assessment time - 00:30:00	Troubleshooting and Error Handling: Pre-Assessment			
Lesson 1 Video time - 00:11:48 Practical application time - 00:28:00 Workbook time - 00:25:00	**Analyze, Detect, and Fix Errors** Syntax Errors Logic Errors Runtime Errors Review 5.1	5.1 Analyze, detect, and fix code segments that have errors 5.1.1 Syntax errors 5.1.2 Logic errors 5.1.3 Runtime errors	D5Q1-D5Q7 (7 questions)	Syntax Errors – pg. 85 511-syntax.py Logic Errors – pg. 86 512-logic.py Runtime Errors – pg. 87 513-runtime.py Review 5.1 – pg. 88 513-analyze.py
Lesson 2 Video time - 00:14:58 Practical application time - 00:40:00 Workbook time - 00:20:00	**Exception Handling** try except else in Exception Handling finally raise Review 5.2	5.2 Analyze and construct code segments that handle exceptions 5.2.1 try 5.2.2 except 5.2.3 else 5.2.4 finally 5.2.5 raise	D5Q8-D5Q17 (10 questions)	Exception Handling: try, except, else, and finally – pg. 90 N/A Exception Handling: raise – pg. 91 N/A Review 5.2 – pg. 92 525-analyze.py
Lesson 3 Video time - 00:11:38 Practical application time - 00:36:00 Workbook time - 00:20:00	**Perform Unit Testing** Unittest Functions Methods Assert Methods Review 5.3	5.3 Perform unit testing 5.3.1 Unittest 5.3.2 Functions 5.3.3 Methods 5.3.4 Assert methods (assertIsInstance, assertEqual, assertTrue, assertIs, assertIn)	D5Q18-D5Q26 (9 questions)	Assert Methods – pg. 94 N/A Functions and Methods – pg. 95 N/A Review 5.3 – pg. 96 534-analyze.py
Post-Assessment Assessment time - 01:00:00	Troubleshooting and Error Handling: Post-Assessment			

LK LearnKey

Domain 6 Lesson Plan

Domain 6 - Operations Using Modules and Tools [approximately 5.5 hours of videos, application questions, and projects]				
Lesson	**Lesson Topic and Subtopics**	**Objectives**	**Practical Application**	**Workbook Projects and Files**
Pre-Assessment Assessment time - 00:30:00	Operations Using Modules and Tools: Pre-Assessment			
Lesson 1 Video time - 00:14:57 Practical application time - 00:36:00 Workbook time - 00:35:00	**System and Command-Line Operations** io os os.path sys Review 6.1	6.1 Perform basic file system and command-line operations by using built-in modules 6.1.1 io 6.1.2 os 6.1.3 os.path 6.1.4 sys (importing modules, opening, reading and writing files, command-line arguments)	D6Q1-D6Q9 (9 questions)	io – pg. 98 N/A os – pg. 99 612-os.py os.path – pg. 100 613-ospath.py sys – pg. 101 N/A Review 6.1 – pg. 102 614-analyze.py
Lesson 2 Video time - 00:13:47 Practical application time - 00:56:00 Workbook time - 00:20:00	**Solve Complex Problems Part 1** Math isnan, sqrt, isqrt, and pi datetime	6.2 Solve complex computing problems by using built-in modules 6.2.1 Math (fabs, ceil, floor, trunc, fmod, frexp, nan, isnan, sqrt, isqrt, pow, pi) 6.2.2 datetime(now, strftime, weekday)	D6Q10-D6Q23 (14 questions)	Math – pg. 104 N/A isnan, sqrt, isqrt, and pi – pg. 105 N/A datetime– pg. 106 622-datetime.py
Lesson 3 Video time - 00:10:37 Practical application time - 00:24:00 Workbook time - 00:30:00	**Solve Complex Problems Part 2** Random with Numbers Random with Lists Final Review	6.2.3 random (randrange, randint, random, shuffle, choice, sample)	D6Q24-D6Q29 (6 questions)	Random with Numbers – pg. 108 N/A Random with Lists – pg. 109 N/A Review 6.2 – pg. 110 N/A
Post-Assessment Assessment time - 01:00:00	Operations Using Modules and Tools: Post-Assessment			