Georgia CTAE

Introduction to Python (11.08300) (2021)

Adopted 2021

Use computational thinking and the Python programming language to solve problems and create programs for real world applications such as game development, data analysis, and the arts. PYTH-1

1. **Describe how computing principles are represented in the Python programming environment.** PYTH-1.1
2. **Analyze a program and describe the structures of Python code.** PYTH-1.2
3. **Describe the characteristics of input and output in a Python programming environment.** PYTH-1.3
4. **Describe the process of compiling and running a program in Python.** PYTH-1.4
5. **Create graphical organizers to model classes, attributes, methods and object relationships.** PYTH-1.5
6. **Describe advantages and disadvantages of information security when creating programs using Python.** PYTH-1.6

Develop techniques for debugging a program. PYTH-2

1. **Design, develop, debug and implement computer programs.** PYTH-2.1
2. **Use various debugging and testing methods to ensure program correctness.** PYTH-2.2
3. **Identify different types of errors that can occur including parameter mismatch and scope errors.** PYTH-2.3

Describe how procedural programming is implemented using the Python programming language. PYTH-3

1. **Describe how procedural programming is implemented using the Python programming language.** PYTH-3.1
2. **Describe the relationship between variables and values when programming in Python.** PYTH-3.2
3. **Implement scripts that use logical, relational, boolean, and mathematical operators.** PYTH-3.3
4. **Describe and use different types of logical operators in Python.** PYTH-3.4
5. **Use truth tables to simulate the results of Boolean operators.** PYTH-3.5

	6. Use arithmetic operators to modify variables in programs. PYTH-3.6
	7. Identify and assign values and different data types to a variable in a program. PYTH-3.7
	8. Convert variable values between different data types in Python. PYTH-3.8
	9. Describe cryptography in Python as the use of libraries that provide encryption to secure data and information (i.e., cryptographic standard libraries). PYTH-3.9
Demonstrate and identify the importance of commenting and documentation in coding. PYTH-4	1. Create single-line comments in programs that explain the purpose of each function PYTH-4.1
	2. Create multi-line comments that highlight the precondition and post condition state of each function PYTH-4.2
	3. Students will understand the importance of and make use of self-documenting code PYTH-4.3
Implement different types of control structures (conditionals, loops, functions) in programs. PYTH-5	1. Analyze how control structures are used in programs using Python. PYTH-5.1
	2. Identify and describe the structure of different types of conditional statements (if statements). PYTH-5.2
	3. Implement conditional statements with the use of mathematical, relational, and Boolean operators. PYTH-5.3
	4. Implement loops in programs using continue, break, and pass keywords. PYTH-5.4
	5. Describe the difference between for loops, while loops, for-each loops, infinite loops, and nested loops. PYTH-5.5
	6. Analyze the effect of a variable in a conditional statement and a loop. PYTH-5.6
	7. Describe how a function is implemented within a program. PYTH-5.7
	8. Implement different types of functions in a program including those with return statements and different parameters. PYTH-5.8
	9. Use external Python libraries to produce multi-media (visual or audio) outputs. PYTH-5.9
Analyze data structures in programs using Python. PYTH-6	1. Identify a list as an ordered series of data under one variable name and accessed with numeric indices. PYTH-6.1

	2. Identify a data structure as a collection of data values and types of data structures in the Python programming language (list, tuple, dictionary and set). PYTH-6.2
	3. Determine which data structures are most appropriate to model the program data (list, tuple, dictionary, or set). PYTH-6.3
	4. Explain the operations that can be applied to data structures using Python including lists and dictionaries as objects that can be changed, and strings and tuples as object that cannot be changed. PYTH-6.4
	5. Implement lists, sets, dictionaries, and tuples as function parameters, return values and internal variables within function bodies. PYTH-6.5
	6. Differentiate between methods and functions and analyze the effect of a method call on a program. PYTH-6.6
Construct and implement strings in programs using Python. PYTH-7	1. Identify strings as arrays of bytes representing Unicode characters. PYTH-7.1
	2. Define standard string methods and their use cases. PYTH-7.2
	3. Declare strings in programs using multiple declaration styles. PYTH-7.3
	4. Manipulate variables using string concatenation and slicing. PYTH-7.4
	5. Implement string search and methods to modify strings in programs. PYTH-7.5
Develop and implement objects in Python. PYTH-8	1. Explain the principles of object-oriented programming using the Python programming language as the use of data values in the form of fields and code in the form of procedures or methods. PYTH-8.1
	2. Describe the use of objects and recognize the difference between an object and an instance. PYTH-8.2
	3. Define objects in Python and use principles of object-oriented programming for declaring methods and combining classes. PYTH-8.3
Analyze algorithms and the implementation of algorithms in Python. PYTH-9	1. Identify an algorithm in Python as a sequence of instructions that transform data or generate conclusions based on the data. 9.2 Implement different algorithms in programs using Python including sorting, search and merge algorithms. PYTH-9.1
	3. Use procedures to reduce the complexity of a program. PYTH-9.3
	4. Identify an algorithm as being recursive. PYTH-9.4
	5. Implement recursion in search and sort algorithms. PYTH-9.5