Revised 8/2023

NOVA COLLEGE-WIDE COURSE CONTENT SUMMARY
ITP 150 – PHYTHON PROGRAMMING (4 CR.)

Course Description

Entails instruction in fundamentals of object-oriented programming using Python. Emphasizes program construction, algorithm development, coding, debugging, and documentation of Python applications. Lecture 4 hours per week.

General Course Purpose

This course provides a comprehensive foundation sufficient for a student to write Python programs in order to meet the minimum programming goals of students who plan to transfer and students who take the course for employment purposes.

Course Prerequisites/Corequisites

None.

Course Objectives

Upon completing the course, the student will be able to:

- Design, develop, code, test, and debug Python programs.
- Use the Python shell for initial exploration of language constructs.
- Use a Python Integrated Development Environment (IDE).
- Use common features of the Python core language and the Python Standard Library.
- Use data types and flow control statements that are the building blocks of all programming.
- Use their foundation knowledge of object-oriented coding techniques to create classes that are applied appropriately in a Python program as a solution for a specific problem statement.

Major Topics to be Included

- The Python Shell and a Python Integrated Development Environment (IDE)
- Python data types and corresponding operators, functions, and methods
- Input and output methods and techniques
- Strings and corresponding operators, functions, and methods
- If Statements
- Loops
- Functions
- Lists and Tuples
- Classes, Objects, and Methods
- Inheritance

Student Learning Outcomes

- Python built-in data types and corresponding operators. Write Python statements that:
    - Utilize the built-in date types: int, float, bool, and str
    - Assign values to variables using expressions containing mathematical and Boolean operators
    - Use arithmetic expressions. Note order of precedence rules
    - Use the using the arithmetic operators: addition, subtraction, multiplication, division, modulus, and/or quotient

- Use one of the augmented assignment operators, such as: +=, -=, *=, and /=

- Input and Output methods and techniques. Write Python statements that:
  - Produce Console Output. Write Python statements that:
    - Call print() to write output statements to the console.
    - Call print() with multiple arguments.
  - Console Input. Write Python statements that:
    - Call input() to read input data from the console. Include output of all primitive types (int, float, and str), and call proper conversion functions as needed.
    - Validate input of data from the keyboard.
- Strings
  - Write Python statements that utilize the str data type
  - Call the following methods on a str: upper(), lower()
  - Call the len() function on a str and other data types
- The Python Shell, Python programs, and an Integrated Development Environment (IDE)
  - Be able to write, check syntax, and run Python programs
  - Be able to write Python programs combining the statements described in this document
- If Statements. Write Python statements that:
  - Use one or more relational operators >, <, >=, <=, ==, !=
  - Use one or more Boolean operators: and, or, not
  - Use an if/elif/else statement (multi-level, many branches)
- Loops. Write Python statements that:
  - Contain a while loop
  - Contain a for loop that uses the range function
  - Contain a for loop that iterates over an iterable object
- Functions.
  - Write statements that call functions and pass arguments.
  - Write various functions in one program.
  - Write functions that accept parameters.
  - Write return statements.
  - Create local variables.
  - Optional: Discuss difference when passing mutable vs. immutable objects
- Lists. Write Python statements that:
  - Create, append, and index from a list.
  - Optional: Pass a list to a function or method. Show effect of mutable list.
- Tuples.
  - Be able to differentiate between lists and tuples and explain the differences and uses of each
- Dictionaries
  - Write Python statements that:

▪ Create, add, retrieve from a dictionary

o Distinguish between various data types of keys and values

- Functions, Methods, and Modules
  - Define the relationship between a class (type) and method.
  - Differentiate between functions, methods, and modules.
  - Write Python statements that call built-in functions
  - Write Python statements that call methods and understand the concept of the object
- Import and use modules from the Python Standard Library

o Required: math module, random module

o Optional: sys, json, sqlite3, csv, other PSL modules or third-party modules.

- Exceptions
  - Describe the concept of inheritance such that the exception inheritance hierarchy can be understood.
  - Write a try / except block to catch an exception
  - Use various exception types, such as TypeError, IndexError, and exceptions that are raised with File I/O
- Other Python language features
  - Optional: Use of the pass statement
  - Optional: Call the type() method to determine the type
- File I/O. Write Python statements that:
  - Open a text file for reading. Read data from a text file. Close the text file.

- Open a text file for writing or append. Write data to a text file. Close the text file.
- Work with text files, such as .txt, .csv, .json or other
- Handle exceptions as required for file input and output

- Programs
  - Be able to write Python programs using the results of an appropriate design methodology as a starting point
  - Be able to write Python programs combining the statements described in this document
  - Be able to write, correct syntax errors, and run Python programs

Required Time Allocation per Topic

To standardize the core topics of ITP 150 so that a course taught at one campus is equivalent to the same course taught at another campus, the following student contact hours per topic are required. Each syllabus should be created to adhere as closely as possible to these allocations. Of course, the topics cannot be followed sequentially. Many topics are taught best as an integrated whole, often revisiting the topic several times, each time at a higher level. The topics listed should comprise 60 contact hours of instruction for a 4-credit class excluding the final exam regardless of the format of instruction. The final exam time is not included in the timetable.

Topic	Hours	Percent
Structured Program Design	4	6.7%
Python built-in data types & corresponding operators	6	10%
Input and Output methods and techniques	5	8.3%
Strings	4	6.7%
If Statements and related operators	6	10%
Loops	7	11.7%
Functions	7	11.7%
Lists, Tuples, Dictionaries	8	13.3%
Classes, Methods, Modules	5	8.3%
Exceptions	3	5%
Files	5	8.3%
Total	60	100%