|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Part** | **1** | **2** | **3** | **4** | **Total** |
| *maximum* | **25** points | **25** points | **25** points | **25** points | **100**G101010 pointsG |
| ***Your Score*** |  |  |  |  |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**The Fundamentals of Python Programming**

Reading Assignment: Thoroughly read Chapter 1 in the course textbook.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 1 Glossary Terms**

Define, in detail, each of these glossary terms from the realm of computer programming logic and design and computer topics, in general. If applicable, use examples to support your definitions. Consult your notes or course textbook(s) as references or by visiting Web sites such as: [**http://www.ask.com**](http://www.ask.com),[**http://www.bing.com**](http://www.bing.com), [**http://www.webopedia.com**](http://www.webopedia.com)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(a) Comparison Operators**

|  |
| --- |
| Python comparison operators include <, >, <=, >=, ==, and !=. The greater than / less than operators compare numerical values, such as `string.length >= otherString.length`, while the == and != look for strict equality / inequality of values between primitive types (strings, ints, floats, bools) and reference equality between non-primitive types (Lists, Dicts, Sets, Tuples), which differs from other programming languages such as JavaScript, wherein == looks for reference equality (i.e., the Objects assignments are pointing to the same place in memory.)   To test for reference equality between non primitive types in Python, one can use the “is” operator, e.g., `objectA is objectB` |

**(b) Control Statements**

|  |
| --- |
| Control statements are used in Python and most other programming languages to control the flow of statement execution. If-elif-else control statements test Boolean conditions (statements that make use of comparison operators, etc., that evaluate to True or False) and execute their respective code blocks. Looping statements (using `while` or `for … in`) similarly test for a certain condition and iterate over a code block until that condition is met. |

**(c) Data Types**

|  |
| --- |
| Python is a semantically safe language, which means that when an assignment is made and a data type is declared (either through type inference via literals or a type conversion function), that the compiler (or if your linter is properly set up, your IDE) will throw an appropriate TypeError message whenever you try to e.g. access a method that does not exist on the data type’s prototype class. This prevents any confusion when trying to, say, iterate over an non-iterable object or subscribe to a non-subscriptable object.   So it pays to know about the common data types and their methods in Python (strings, ints, bools, floats, dicts, lists, and tuples being the most common).   When building your own data types and methods via classes in OOP, however, it is wise to include exception statements and type hinting (new as of 3.5 <https://docs.python.org/3/library/typing.html>) to help other developers (and your future self) when working with your data type’s interface. |

**(d) Function Calls**

|  |
| --- |
| Python functions are “first class data objects” (p. 23) and can be assigned to variables, saved in data structures (e.g., as a method in a dictionary object), passed as an argument to another function (e.g., as a “callback” function argument in a higher order function), etc.  Calling a function can be done in the global name space (e.g., `myFunc(“someArgument”)`), called from an enclosed namespace via an object’s method (e.g., `myObj.myObjFunc(“myArg”)`), or called locally from a function defined within a function. Functions can mutate data within the modular namespace (e.g., `myMutatingFunc(myGlobalVariable)`), as well as internally, and Class methods / functions can mutate data on the instance specifically (via `self`, the usual way of doing things) or on the class itself (via the “@classmethod` decorator, which is an interesting way to define factory functions within a Class itself!) |

**(e) String Literals**

|  |
| --- |
| “This is a string literal”. When assigning a variable to a string literal (and other object literals, such as lists [], tuples (), and dicts {},), this tells Python that “this is a new instance of a string/list/dict/etc. data type”, which Python then uses to construct the new data type from its collection of built-in data types.   In some languages, for example, TypeScript or Objective-C, you have to declare a datatype, even universal primitives like strings, ints etc., upon variable assignment. But languages like Python and Javascript make it easy by inferring the data type from the shape of the object literal, and with custom data types, using the prototypical Class as a typology for a class instance.   String literals are a lot of fun to work with, especially if you use “f strings” which are available since 3.6. |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 2 True / False Exercises**

For each of these exercises, enter True or False in the spaces provided.

**FALSE** **(1)** The keyword **while** means the same thing as **While** in Python.

**TRUE (2)** In Python, **\_MyVar15** is a valid variable name.

**FALSE** **(3)** To make your program more secure, use obscure variable names such as **xz14dEEa** .

**TRUE.** **(4)** In Python, a variable containing the value 5 will evaluate as true if used with the **and** logical operator.

**FALSE, prints AFTER its arguments are printed** **(5)** The **print()** function outputs a newline by default before its arguments are printed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 3 Multiple Choice Exercises**

Select the correct response or responses.

**(1)** Which of the following is true about Python keywords?

(a) they can begin with a number, letter or hyphen

**(b)**  **they are case sensitive**

(c) they are written in uppercase

(d) they can be a maximum of 6 characters long

**(2)** Which statement is true about Python syntax?

(a) a code block must begin with a left brace

(b) each code statement must end with a semicolon

(c) white space is ignored

**(d) blocks of code are indicated by indentation**

**(3)** Which comparison operation returns True if variable x is not equal to variable y?

(a) x .ne. y (b) x neq y

(c) x <> y **(d) x != y**

**(4)** Assume x is equal to 5 and y is equal to 0 . Which comparison operation returns True?

(a) x == y or y (b) y < 5 and y

**(c) (x > y) and x**  (d) x > 0 and None

**(5)** What is the output of the following code?

**sum = 0**

**value = 3**

**while value <= 5 :**

**sum += value**

**value += 1**

**print (sum)**

(a) 15 (b) 10 **(c) 12** (d) 18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 4 Programming Exercises**

**(1)** **EOQ ( Economic Order Quantity )**

*from* math *import* sqrt

print("Welcome to the EOQ Calculator")

D = float(input("Please enter demand rate: "))

S = float(input("Please enter setup costs: "))

H = float(input("Please enter holding costs: "))

EOQ = sqrt((2\*D\*S)/H)

print(f"The EOQ is: {EOQ:0.2f}, you should order {round(EOQ)} units.")

**(2)** **( Sequential Program Control )**

X = 8

**(3) ( The Distance Between Two Points )**

Output is 3.61

**(4) ( Logical Operators )**

Both logical operators output the same result: True

**(5) ( The List Data Structure )**

The four events that are occurring when these statements are executed are:

1. An empty list is assigned to list1
2. The string “value” is appended to the end of the list (which is position 0 in this case)
3. The integer 10 is inserted at position 1
4. The element in the first position (“value”) is popped out from the list, leaving only the integer 10 in the list.