7001 Python Programming Exercises

The following are coding exercises to test your understanding of basic Python functionality that you will encounter in the MS program. Most of the coding questions involve writing a function to get the job done and passing parameters to the function. Be aware that Python is NOT a strongly typed language which means the type of parameters are not checked by the compiler.

However, Python 3.5 and above have a way to document s and this information may be used by tools to help when you incorrectly pass an integer when your code expects a string.

In these exercises, we'll use the following format to specify what your function should look like, what data types are expected to be passed in and what kind of return value is expected.

The function below takes and returns a string and is annotated as follows:

```
def greeting(name: str) -> str:
    return 'Hello ' + name
```

In the function `greeting`, the argument `name` is expected to be of type `str` and the return type `str`.

Note that Python compilers will not report type errors (like Java, and C++ do). These type annotations are for documentation purposes and are used to document your intent.

The types used here are: int, float, bool, List, Tuple, Dict, None, Any (expect any type) , Number (int or float)

If you were asked to take this course as part of your admission, you should plan to submit the results of your coding explorations. For each assignment, capture the code and the output and submit as a PDF file. When you prepare your submissions, use COURIER font which is the standard for code examples. Courier font provides uniform spacing is makes it easy to read. Happy coding!

Assignment 3: Python Data Structures: Lists, Dictionaries, Tuples

1. Write a Python function to concatenate following dictionaries and return a new one. Display the new dictionary outside the function

Sample Dictionary :
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

function: def merge_dicts (d1: Dict, d2: Dict, d3: Dict) -> Dict:

2. Write a Python function to check whether a given key already exists in a dictionary.

function: def isKey (d1: Dict, key: Any) -> Boolean:

3. Write a Python function to get the maximum and minimum value in a dictionary. Return max and min as a tuple. Ignore any values that are not integers.

function: def min_max (d1: Dict) -> Tuple:

IN: {'foo':9, 'bar': 'xyz', 'baz':8, 'boz':4}

OUT: (9, 4)

4. Write a Python function to replace dictionary values with their average. Display the dict outside the function.

IN: {'foo':10, 'bar': 'xyz', 'baz':20, 'boz':30}

OUT: {'foo':20, 'bar': 'xyz', 'baz':20, 'boz':20}

function: def avg (d1: Dict) -> None:

5. Write a Python function to convert string values of a given dictionary, into integer/float datatypes. Return and display the new dictionary.

function: def str_to_num (d1: Dict) -> Dict:

IN: {'x': '10', 'y': '20', 'z': '30'}, {'p': '40', 'q': '50', 'r': '60'}
OUT: {'x': 10, 'y': 20, 'z': 30}, {'p': 40, 'q': 50, 'r': 60}

IN: {'x': '10.12', 'y': '20.23', 'z': '30'}, {'p': '40.00', 'q': '50.19', 'r': '60'}
OUT: {'x': 10.12, 'y': 20.23, 'z': 30.0}, {'p': 40.0, 'q': 50.19, 'r': 60}

6. Write a Python function to count the elements in a list until an element is a tuple.

function: def count1 (p1: List) -> int:

IN: [2, 'foo', 3.4737, (4,5), 'zzz']
OUT: 3

7. Write a Python function to compute the sum of all the elements of each tuple stored inside a list of tuples.

function: def tup1 (p1: List) -> List:

IN: [(1, 2), (2, 3), (3, 4)]

OUT: [3, 5, 7]
Original list of tuples:

SETS

8. Write a Python function to check if a given value is present in a set or not.

function: def set1 (s1: Set, val: ANY) -> bool:

IN: (3,9,11,1,'fiji'), 1
OUT: True

Assignment 4: args and kwargs

1. The following adds two numbers and returns the result.

```
def add(a,b):
        return a+b
```

Use *args as parameter to return the sum of all numbers passed to the function.

function: def add_nums(*args) ->Number:

Beware of parameters that are not int or float

IN: add_nums(3,4,5, 'foo', 6.2)

OUT: 18.2

2. Use **kwargs to print the names and the values of all parameters passed in.

function foo2 (a, **kwargs) -> NoReturn :

```
IN: foo2(99, c=3, d=8)
Prints:
a is 99
c is 3
d is 8
```

Assignment 5: Objects and Exceptions

1. Write a Python class named Rectangle. Pass in length and width to the constructor.

Write a method which will compute the area of a rectangle.

2. Write a Python class named Circle constructed by a radius.

Write two methods which will compute the area and the perimeter of a circle.

Exceptions

3. Write a function to divide two integers, n1 and n2 as n1/n2

function: divide (n1: int, n2:int) -> Any:

If n2 is passed as zero, catch the exception, display "Can't divide by Zero" and return None as the value of the function.

Do not use if statements – use Exceptions!