Thursday Nov 14, 2024

In-class Handout

COSC 101C Intro to Computing I

Prof. Forrest Davis

Name:

Discuss and complete the following questions with the person nearest you. You **may** be asked to share your thoughts with the class.

1. Write a function called **hundreds_chart** that produces the following output

```
  1   2   3   4   5   6   7   8   9  10
 11  12  13  14  15  16  17  18  19  20
 21  22  23  24  25  26  27  28  29  30
 31  32  33  34  35  36  37  38  39  40
 41  42  43  44  45  46  47  48  49  50
 51  52  53  54  55  56  57  58  59  60
 61  62  63  64  65  66  67  68  69  70
 71  72  73  74  75  76  77  78  79  80
 81  82  83  84  85  86  87  88  89  90
 91  92  93  94  95  96  97  98  99 100
```

You can use the following helper function to help you align all of the numbers:

```
def print_num_right_align(num:int) -> None:
    num = str(num)
    pad = 4 - len(num)
    print((" " * pad) + num, end="")

def hundreds_chart() -> None:
    """Displays a hundreds chart"""
    for row in range(0,10):
        for col in range(1,11):
            num = row * 10 + col
            print_num_right_align(num)
        print("")

hundreds_chart()
```

2. Write a function called **multiplication_table** that takes a positive integer and outputs a multiplication chart from 1 through that number. For example multiplication_table(5) should produce:

```
     |   1   2   3   4   5
-----+--------------------
   1 |   1   2   3   4   5
   2 |   2   4   6   8  10
   3 |   3   6   9  12  15
   4 |   4   8  12  16  20
   5 |   5  10  15  20  25
```

Again, use the print_num_right_align helper function to help you align all of the numbers.

```
def multiplication_table(upper: int) -> None:
    """Displays a multiplication table"""
    # Print column headers
    print("     |", end="")
    for col in range(1,upper+1):
        print_num_right_align(col)
```

```
    print("")

    # Print column header line
    print("-----+" + ("-" * (upper * 4)))

    # Print rows
    for row in range(1,upper+1):
        # Print row number
        print_num_right_align(row)
        print(" |", end="")

        # Print multiplication results
        for col in range(1,upper+1):
            num = row * col
            print_num_right_align(num)
        print("")

multiplication_table(5)
```

3. Write a program that asks users about their pets. You should create a data structure that encodes the information you've gotten from users. Below is an example of asking users for the information.

```
What's your name?  forrest
What type of animal is your pet?  cat
What is your cat's name?  fig
Do you have more pets [y/n]?  y
What type of animal is your pet?  cat
What is your cat's name?  bella
Do you have more pets [y/n]?  n
Are there more users [y/n]?  y
What's your name?  irene
What type of animal is your pet?  rabbit
What is your rabbit's name?  rabbit
Do you have more pets [y/n]?  n
Are there more users [y/n]?  n
```

```
def addUserPets(data: dict) -> None:
    """ Asks a user for their pets' information and adds to it to data """
    name = input("What's your name? ")
    data[name] = {'types': [], 'names': []}
    addPet = True
    while addPet:
        petType = input("What type of animal is your pet? ")
        petName = input(f"What is your {petType}'s name? ")
        data[name]['types'].append(petType)
        data[name]['names'].append(petName)
        if input("Do you have more pets [y/n]? ").lower() == 'n':
            addPet = False

def main() -> None:
    data = {}
    moreUsers = True
    while moreUsers:
        addUserPets(data)
        if input("Are there more users [y/n]? ").lower() == 'n':
            moreUsers = False
main()
```

4. What is the output of the following code snippet?

```
import random
def func(aDict: dict) -> None:
    for i in range(15):
        aDict[i] = random.randint(0, 15)

def bar(aDict: dict) -> None:
    for i in range(10, 24, 2):
        aDict[i] = random.randint(0, 15)

def main() -> None:
    d = {}
    func(d)
    bar(d)
    print(len(d))
main()
```

19

5. **Challenge**: Write a function that returns a list of all the ngrams of any size in a string. A ngram is a grouping of n words based on their occurrence in a sentence. For example, imagine you have the sentence *the cat is out of the bag*, for the size 1 your function would return:

   ['the', 'cat', 'is', 'out', 'of', 'the', 'bag']

For size 3, your function would return:

```
['the cat is', 'cat is out', 'is out of', 'out of the', 'of the bag']
```

```
def get_ngrams(sentence: str, size: int) -> list:
    """ Returns size-grams in a sentence

    Parameters:
        sentence (str): a sentence
        size (int): desired n in ngrams
    Returns:
        ngrams (list): A list of ngrams

    >>> sentence = "the cat is out of the bag"
    >>> get_ngrams(sentence, 1)
    ['the', 'cat', 'is', 'out', 'of', 'the', 'bag']
    >>> get_ngrams(sentence, 3)
    ['the cat is', 'cat is out', 'is out of', 'out of the', 'of the bag']
    """
    ngrams = []
    index = 0
    sentence = sentence.split(' ')
    while index <= (len(sentence)-size):
        ngram = " ".join(sentence[index:index+size])
        ngrams.append(ngram)
        index += 1
    return ngrams

def main() -> None:
    # We can use doctest to test our code with the examples in our docstring !
    # In this, that is checking that get_ngrams('the cat is out of the bag', 1)
```

```
    # returns ['the', 'cat', 'is', 'out', 'of', 'the', 'bag'] along with the
    # size = 3 example
    import doctest
    doctest.testmod()
main()
```