Assignment2

October 11, 2024

0.0.1 CS2101 - Programming for Science and Finance

Prof. Götz Pfeiffer School of Mathematical and Statistical Sciences University of Galway

---

1 Computer Lab 2

- Robert Davidson
- Student ID: 21428974
- Email

1.1 1. Lists.

Lists are an important compound data structure in Python. Being dynamic and mutable, they can be applied in many different ways for different purposes.

1. Using a `for` loop, write a function `multiple_list` with two arguments, `c` and `numbers`, where `c` is a number and `numbers` is a list of numbers, which returns the list of `c`-multiples of the numbers in the list `numbers`. That is, when `numbers` is `[1,2,3,4]`, and `c` is `5`, the function should return the list `[5, 10, 15, 20]`.

[97]:
```
def multiple_list(c :int, numbers : list[int]) -> list[int]:
    multiples = []
    for num in numbers:
        multiples.append(num * c)

    return multiples
```

[98]:
```
numbers = [1,2,3,4]
c = 5

multiple_list(c, numbers)  == [5, 10, 15, 20]
```

[98]:
```
True
```

2. Using a `for` loop, write a function `sum_lists` which takes two arguments `numbersA` and `numbersB`, both lists of numbers, and returns the single list of all sums of corresponding

numbers in the lists `numbersA` and `numbersB`. That is, if `numbersA` is `[1,2,3]` and `numbersB` is `[1,0,-1]`, the function should return the list `[2,2,2]`.

[99]:
```
def sum_lists(numbersA : list[int], numbersB : list[int]) -> list[int]:
    sums = []

    for i in range(len(numbersA)):
        sums.append(numbersA[i] + numbersB[i])

    return sums
```

[100]:
```
numbersA = [1,2,3]
numbersB = [1,0,-1]

sum_lists(numbersA, numbersB) == [2, 2, 2]
```

[100]:
```
True
```

3. Modify your `sum_list` function in such a way that it prints an error message in case the two given argument lists do not have the same length.

[101]:
```
def sum_lists_v2(numbersA : list[int], numbersB: list[int]) -> list[int]:
    if not len(numbersA) == len(numbersB):
        print("Lists must be the same length")
    else:
        sums = []

        for i in range(len(numbersA)):
            sums.append(numbersA[i] + numbersB[i])

        return sums

numbersA = [1,2,3]
numbersB = [1,0,-1, 1]
sum_lists_v2(numbersA, numbersB)
```

```
Lists must be the same length
```

The first few digits of π are like so:

[102]:
```
pi = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9, 3, 2, 3, 8, 4, 6, 2, 6, 4,␣
↪3, 3,
      8, 3, 2, 7, 9, 5, 0, 2, 8, 8, 4, 1, 9, 7, 1, 6, 9, 3, 9, 9, 3, 7, 5, 1, 0]
```

4. Consult the python documentation and find the **list method** that allows you to determine the **position** of the first occurrance of the digit 9 in the above list. What is the position of the first 9 in this list? What is the position of the second 9 in this list?

[103]:
```
firstNineIndex = pi.index(9)
secondNineIndex = pi.index(9, firstNineIndex + 1)

print(f"{firstNineIndex = }")
print(f"{secondNineIndex = }")
```

```
firstNineIndex = 5
secondNineIndex = 12
```

5. Consult the python documentation and find out what `enumerate` does to a list. What does `list(enumerate(pi))` return? Using `enumerate`, write a python program that determines **all** the positions of the digit 9 in this list.

[104]:
```
def findAllPositions(d : int, numbers : list[int]) -> list[int]:
    for index, number in enumerate(pi):
        if number == d:
            print(f"{index = }")
```

[117]:
```
findAllPositions(9, pi)
```

```
index = 5
index = 12
index = 14
index = 30
index = 38
index = 42
index = 44
index = 45
```

6. Consult the python documentation and find the **list method** that allows you to determine the **number** of occurrances of the digit 9 in the above sequence `pi`. How often does 9 appear in the sequence `pi`?

[105]:
```
pi.count(9)
```

[105]: 8

1.2 2. List Comprehension

Often, the purpose of a `for` loop is to construct a new list from one or more old lists. Many such tasks can be concisely expressed through list comprehension, where the elements of the new list are constructed by a **formula**, rather than a loop. For instance

```
[ x**2 for x in range(10) ]
```

is the list of squares of the numbers $0, \ldots, 9$.

1. Using **list comprehension**, write a function `multiple_list1` with two arguments, `c` and `numbers`, where `c` is a number and `numbers` is a list of numbers, which returns the list of

c-multiples of the numbers in the list `numbers`. That is, when `numbers` is `[1,2,3,4]`, and `c` is `5`, the function should return the list `[5, 10, 15, 20]`.

[106]:
```
def multiple_list1(c :int, numbers: list[int]) -> list[int]:
    return [num * c for num in numbers]
```

[107]:
```
numbers = [1,2,3,4]
c = 5

multiple_list1(c, numbers) == multiple_list(c, numbers)  == [5, 10, 15, 20]
```

[107]:
```
True
```

2. Using **list comprehension**, write a function `sum_lists1` which takes two arguments `numbersA` and `numbersB`, both lists of numbers, and returns the single list of all sums of corresponding numbers in the lists `numbersA` and `numbersB`. That is, if `numbersA` is `[1,2,3]` and `numbersB` is `[1,0,-1]`, the function should return the list `[2,2,2]`.

[108]:
```
def sum_lists1(numbersA : list[int], numbersB : list[int]) -> list[int]:
    return [numbersA[i] + numbersB[i] for i in range(len(numbersA))]
```

[109]:
```
numbersA = [1,2,3]
numbersB = [1,0,-1]

sum_lists1(numbersA, numbersB) == sum_lists(numbersA, numbersB) == [2, 2, 2]
```

[109]:
```
True
```

3. Write a python program that determines, for each digit d in $\{0, 1, \ldots, 9\}$, the number of occurrances of d in the list `pi` above.

[119]:
```
def findOccurence(pi : list[int]) -> int:
    for d in range(10):
        print(f"Occurences of {d}  = {pi.count(d)}")
```

[120]:
```
findOccurence(pi)
```

```
Occurences of 0  = 2
Occurences of 1  = 5
Occurences of 2  = 5
Occurences of 3  = 9
Occurences of 4  = 4
Occurences of 5  = 5
Occurences of 6  = 4
Occurences of 7  = 4
Occurences of 8  = 5
Occurences of 9  = 8
```

4. Write a python function `standardBasisElement` that takes two arguments, `n` and `i`, and computes and returns a list of length `n` with all entries 0, except for the entry at position i,

which should be 1. That is, `standardBasisElement(5, 3)` should return the list `[0, 0, 0, 1, 0]`.

[123]:
```
def standardBasisElement(n : int, i : int) -> list[int]:
    return [1 if j == i else 0 for j in range(n)]
```

[126]:
```
standardBasisElement(5,3) == [0, 0, 0, 1, 0]
```

[126]:
```
True
```

1.3 3. Strings

In Python, strings are like lists, except that they are immutable and homogeneous.

Suppose that `text` is the following string.

[112]:
```
text = "Programming for Science and Finance"
```

1. Consult the Python documentation and find the **string method** that converts a string into ALL CAPITALS. What is the all capitals version of `text`?

[128]:
```
capitalText = text.upper()
capitalText
```

[128]:
```
'PROGRAMMING FOR SCIENCE AND FINANCE'
```

The builtin function `ord` returns the ASCII code of a single character like so:

[114]:
```
ord('A')
```

[114]:
```
65
```

The inverse function of `ord` is `chr`. It converts an ASCII code into the corresponding character:

[115]:
```
chr(65)
```

[115]:
```
'A'
```

2. Using **list comprehension**, construct the list (or string) of all 26 uppercase letters of the alphabet. Then use this list to determine, for each letter `l`, the number of occurrances of `l` in the all capitals version of text.

[132]:
```
alphabetUpper = [chr(65 +i) for i in range(26)]
alphabetUpper
```

[132]:
```
['A',
 'B',
 'C',
 'D',
 'E',
```

```
 'F',
 'G',
 'H',
 'I',
 'J',
 'K',
 'L',
 'M',
 'N',
 'O',
 'P',
 'Q',
 'R',
 'S',
 'T',
 'U',
 'V',
 'W',
 'X',
 'Y',
 'Z']
```

[134]:
```python
def charOccurence(alphabet : list[str], upperText : str) -> list[int]:
    for letter in alphabet:
        print(f"{letter} = {upperText.count(letter)}")

charOccurence(alphabetUpper, capitalText)
```

```
A = 3
B = 0
C = 3
D = 1
E = 3
F = 2
G = 2
H = 0
I = 3
J = 0
K = 0
L = 0
M = 2
N = 5
O = 2
P = 1
Q = 0
R = 3
S = 1
T = 0
```

```
U = 0
V = 0
W = 0
X = 0
Y = 0
Z = 0
```