Assignment1

October 25, 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 1

1.1 1. Warmup.

1. The code in the following cell computes the 10 th power of 2. Determine its exact value. Then modify the code so that it represents the 1000 th power of 2 and compute that value.

[1]:
```
2**1000
```

[1]:
```
10715086071862673209484250490600018105614048117055336074437503883703510511249361
22493198378815695855812759467291755314682518714528569231404359845775746985748039
34567774824230985421074605062371141877954182153046474983581941267398767559165543
94607706291457119647768654216766042983165262438683720566806937 6
```

2. The code in the following cell computes the sum of `"1234"` and `"5432"`. Determine that value. Then modify the code so that it represents the sum of the **numbers** 1234 and 5432 and compute that value.

[2]:
```
1234 + 5423
```

[2]:
```
6657
```

1.2 2. One Liners

1. Write a `Python` program that determines the number of **full** weeks left until the end of the year, if there are 91 days remaining.

[3]:
```
fullWeeks = 91 // 7
fullWeeks
```

[3]:
```
13
```

2. Write a `Python` program that allows you to determine whether 200001079 is a multiple of 5323.

[4]:
```
isMultiple = 200001079 % 5323 == 0
isMultiple
```

[4]:
```
True
```

3. Write a `Python` program that computes the interest on an investment of EUR 89235.41 at an interest rate of 4.5 percent.

[5]:
```
interest = 89235.41 * 0.045 # principal * rate (decimal)
interest
```

[5]:
```
4015.59345
```

4. Write a `Python` program that determines the future value of a sum of EUR 189235.41 invested for two years at an interest rate of 4.4 percent per year, with interest compounded every three months.

$$FV = P \times \left(1 + \frac{r}{n}\right)^{n \times t}$$

- With:
- FV : Future value
- P : Initial Investment (principal amount)
- r : Annual interest rate
- n : Num. times interest rate is compounded per year
- t : Num. years

[6]:
```
future_value = 189235.41 * (1 + 0.044 / 4) ** (4 * 2)
future_value
```

[6]:
```
206543.55615739748
```

1.3 3. Short Functions

1. Write a short `python` function that computes the sum of the cubes of the numbers from 1 to n.

[7]:
```
def sum_of_cubes(n : int) -> int: ## Change function name 'sum_of_squares ->␣
↪sum_of_cubes'
    return (n * (n+1) / 2)**2
```

2. The sum of the first 10 cubes is 3025. Verify that this is the value your function returns.

[8]:
```
sum_of_cubes(10) == 3025
```

[8]:
```
True
```

3. Write a `python` function `mpg2lp100km` that converts miles per gallon into litres per 100 kilometres.

- $L/100\,\text{km} = \dfrac{\text{Litres in 100km}}{\text{Miles per gallon}}$
- $1\,\text{mile} = 1.609344\,\text{km}$
- $\text{Litres in 100km} = \dfrac{100\,\text{km}}{1.609344\,\text{km/mile}}$
- $4.54609\,\text{liters/gallon}$ (Imperial)
- $L/100\,\text{km} = \dfrac{100\,\text{km}}{1.609344\,\text{km/mile}} \times \dfrac{4.54609\,\text{litres}}{\text{miles per gallon (MPG)}}$

$$L/100\,\text{km} = \frac{282.481}{\text{MPG}}$$

[9]:
```
def mpg2lp100km(mpg : int) -> int:
    return (282.481) / mpg
```

4. It is known that 35 miles per gallon are roughly 8.1 litres per 100 km. Verify that this is what your function computes.

[10]:
```
litresPer100km =  mpg2lp100km(35)
roundedLitresPer100km = round(litresPer100km, 1)
roundedLitresPer100km == 8.1
```

[10]:
```
True
```

1.4 4. More Functions.

1. In a certain country, a single employee pays 20% tax on an annual income of up to EUR 42,000, and 40% on every euro earned in excess of that. Write a Python function `tax` that takes the annual income in euro as argument, and computes and returns the amount (in euro) of tax to be paid on that.

[11]:
```
def tax(income : int) -> int:
    lower_threshold = 42_000
    lower_rate = 0.2
    higher_rate = 0.4

    if income <= lower_threshold:
        return income * lower_rate
    return lower_threshold * lower_rate + (income - lower_threshold) *␣
↪higher_rate
```

2. How much tax is to be paid on an annual income of EUR 90,000?

[12]:
```
tax(90_000)
```

[12]:
```
27600.0
```

3. Let's say an **anagram** of a word is **any** rearrangement of its letters (regardless of whether this gives a word from the dictionary or not). So the rearrangements of the word `"lab"` are

```
[ "abl", "alb", "bal", "bla", "lab", "lba" ]
```

Write a python function `anagrams`, which given a (short) word finds **all** its anagrams and returns the list of them.

[13]:
```
def anagrams(word : str, allowDupes : bool = False ) -> list:
    # Function to generate permutations of the given word
    def generatePermutations(currentAnagram, remainingChars):
        # Base case: if no characters are remaining, add the current anagram to␣
 ↪the result list
        if len(remainingChars) == 0:
            ## Allow duplicate anagrams if allowDupes is True, or if the␣
 ↪current anagram is not already in the result list
            if allowDupes or currentAnagram not in result:
                result.append(currentAnagram)
        else:
            # Recursive case: iterate over the remaining characters
            for i in range(len(remainingChars)):
                # Generate new permutations by choosing each character in turn
                # and recursively calling the function with the remaining␣
 ↪characters
                generatePermutations(currentAnagram + remainingChars[i],␣
 ↪remainingChars[:i] + remainingChars[i+1:])

    result = []  # Initialize an empty list to store the anagrams
    generatePermutations("", word)  # Start the recursion with an empty current␣
 ↪anagram and the full word
    return result  # Return the list of generated anagrams

sorted(anagrams("lab")) == [ "abl", "alb", "bal", "bla", "lab", "lba" ] ##␣
 ↪Check if the result matches the expected anagrams
```

[13]:
```
True
```

4. What are the anagrams of the word `"food"`?

[14]:
```
anagrams("food", allowDupes=False)
```

[14]:
```
['food',
 'fodo',
 'fdoo',
 'ofod',
```

```
'ofdo',
'oofd',
'oodf',
'odfo',
'odof',
'dfoo',
'dofo',
'doof']
```

1.5 5. Euclid's Algorithm

- Modify this code, by adding a suitable `print` statement, so that the function when called prints out its step-by-step progress, as you would do yourself. That is, `gcd(60, 24)` should give a report like

```
def gcd(a : int, b : int) -> int:
    if b == 0:
        return a
    else:
        return gcd(b, a % b)
```

```
gcd(60, 24) =
gcd(24, 12) =
gcd(12, 0) =
12
```

[15]:
```
def gcd(a, b):

    print(f"gcd({a}, {b}) = ")

    if b == 0:
        print(a)
    else:
        return gcd(b, a % b)
```

[16]:
```
gcd(60, 24)
```

```
gcd(60, 24) =
gcd(24, 12) =
gcd(12, 0) =
12
```
