00ex_introduction

January 18, 2025

1 1. (done) The MickeyMouse problem

a) Write a program that prints the numbers from 1 to 100. But for multiples of 3 print `Mickey` instead of the corresponding number and for the multiples of 5 print `Mouse`. For numbers which are multiples of both three and five print `MickeyMouse`

b) Put the result in a tuple and substitute `Mickey` with `Donald` and `Mouse` with `Duck`

[6]:
```
a = "Mickey"
b = "Mouse"
c = "Donald"
d = "Duck"

result = []

for i in range(1, 101):
    if i % 3 == 0 and i % 5 == 0:
        element = a + b
    elif i % 3 == 0:
        element = a
    elif i % 5 == 0:
        element = b
    else:
        element = i
    print(element, end='\n')
    result.append(element)

for i, x in enumerate(result):
    if x == a:
        result[i] = c
    elif x == b:
        result[i] = d
    elif x == a + b:
        result[i] = c + d

tuple(result)
print(result)
```

1

```
2
Mickey
4
Mouse
Mickey
7
8
Mickey
Mouse
11
Mickey
13
14
MickeyMouse
16
17
Mickey
19
Mouse
Mickey
22
23
Mickey
Mouse
26
Mickey
28
29
MickeyMouse
31
32
Mickey
34
Mouse
Mickey
37
38
Mickey
Mouse
41
Mickey
43
44
MickeyMouse
46
47
Mickey
49
```

```
Mouse
Mickey
52
53
Mickey
Mouse
56
Mickey
58
59
MickeyMouse
61
62
Mickey
64
Mouse
Mickey
67
68
Mickey
Mouse
71
Mickey
73
74
MickeyMouse
76
77
Mickey
79
Mouse
Mickey
82
83
Mickey
Mouse
86
Mickey
88
89
MickeyMouse
91
92
Mickey
94
Mouse
Mickey
97
```

```
98
Mickey
Mouse
[1, 2, 'Donald', 4, 'Duck', 'Donald', 7, 8, 'Donald', 'Duck', 11, 'Donald', 13,
14, 'DonaldDuck', 16, 17, 'Donald', 19, 'Duck', 'Donald', 22, 23, 'Donald',
'Duck', 26, 'Donald', 28, 29, 'DonaldDuck', 31, 32, 'Donald', 34, 'Duck',
'Donald', 37, 38, 'Donald', 'Duck', 41, 'Donald', 43, 44, 'DonaldDuck', 46, 47,
'Donald', 49, 'Duck', 'Donald', 52, 53, 'Donald', 'Duck', 56, 'Donald', 58, 59,
'DonaldDuck', 61, 62, 'Donald', 64, 'Duck', 'Donald', 67, 68, 'Donald', 'Duck',
71, 'Donald', 73, 74, 'DonaldDuck', 76, 77, 'Donald', 79, 'Duck', 'Donald', 82,
83, 'Donald', 'Duck', 86, 'Donald', 88, 89, 'DonaldDuck', 91, 92, 'Donald', 94,
'Duck', 'Donald', 97, 98, 'Donald', 'Duck']
```

2 2. (done) The swap function

Write a function that swap the values of two input variables x and y (whatever the type). Try to do that also without a temporary variable

[7]:
```python
def swap(a, b):
    return b, a

a = 1
b = 2
a, b = swap(a, b)
print(a, b)
```

```
2 1
```

3 3. (done) Computing the distance

Write a function that calculates and returns the euclidean distance between two points u and v, where u and v are both 2-tuples *(x,y)*. For example, if u=*(3,0)* and v=*(0,4)*, the function should return 5

[5]:
```python
import math #Basic library with math functions

def euc2(u:tuple, v:tuple):
    assert len(u) == len(v), "u and v must have the same length"
    assert isinstance(u, tuple) and isinstance(v, tuple), "u and v must be␣
 ↪tuple"
    sum = 0
    for i in range(len(u)):
        sum += (u[i] - v[i])**2
    return math.sqrt(sum)

u = (3, 0)
v = (0, 4)
```

```
dist = euc2(u, v)
dist
```

[5]: 5.0

4 4. (done) Counting letters

Write a program to calculate the number of times each character occurs in a given string s. Ignore differneces in capitalization

[11]:
```
s="Write a program that prints the numbers from 1 to 100. \
But for multiples of three print Mickey instead of the number and for the␣
 ↪multiples of five print Mouse. \
For numbers which are multiples of both three and five print MickeyMouse"

def count_characters(s:str) -> dict:
    assert isinstance(s, str), "s type must be str"
    # lowercase eveything
    s = s.lower()
    unique_characters = []
    for c in s:
    # find the unique keys
        if not c in unique_characters:
            unique_characters.append(c)
    dict = {}
    # populate the values
    for u in unique_characters:
        dict[u] = s.count(u)
    return dict


s = "Miriam"
dict = count_characters(s)
print(dict)
```

```
{'m': 2, 'i': 2, 'r': 1, 'a': 1}
```

5 5. (done) Isolating the unique

Write a function that determines and count the unique numbers in the list l

[19]:
```
l = [36, 45, 58, 3, 74, 96, 64, 45, 31, 10, 24, 19, 33, 86, 99, 18, 63, 70, 85,
 85, 63, 47, 56, 42, 70, 84, 88, 55, 20, 54, 8, 56, 51, 79, 81, 57, 37, 91,
 1, 84, 84, 36, 66, 9, 89, 50, 42, 91, 50, 95, 90, 98, 39, 16, 82, 31, 92, 41,
 45, 30, 66, 70, 34, 85, 94, 5, 3, 36, 72, 91, 84, 34, 87, 75, 53, 51, 20, 89,␣
 ↪51, 20]
```

```
def find_unique(l: list):
    dictionary = {}
    for element in l:
        if not element in list(dictionary.keys()):
            dictionary[element] = 1
        else:
            dictionary[element] += 1
    return dictionary

dict = find_unique(l)
print(dict)
```

```
{1: 1, 2: 2, 3: 1, 4: 1, 5: 1, 10: 1}
```

6 6. (done) Combination of functions

Write two functions - one that returns the square of a number, and one that returns the cube. Now write a third function that returns the number raised to the 6th power using the two previous functions.

[20]:
```
def square(x:float):
    assert isinstance(x, (float, int)), "x type must be float or int"
    return x**2

def cube(x: float):
    assert isinstance(x, (float, int)), "x type must be float or int"
    return x**3

def sixth_power(x: float):
    return cube(square(x))


y = sixth_power(2)
y
```

[20]:
```
64
```

7 7. (done) Cubes

Create a list of the cubes of x for x in *[0, 10]* using:

a) a for loop

b) a list comprehension

[21]:
```
# with for loop
cubes = []
for n in range(0, 11):
    cubes.append(n**3)
print(cubes)

# with list comprehension
cubes = [n**3 for n in range(0, 11)]
print(cubes)
```

```
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

8 8. (done) Nested list comprehension

A Pythagorean triple is an integer solution to the Pythagorean theorem $a^2 + b^2 = c^2$. The first Pythagorean triple is (3,4,5). Find and put in a tuple all unique Pythagorean triples for the positive integers a, b and c less than 100.

(3,4,5) (3,5,4) NO (3,4,6) (3,4,7) (3, 4, 50) (4, 4, 4) (4, 5, 4) (4, 6, 4)

if a <= b <= c then I am sure they are unique ....

[32]:
```
unique_triples = [] # list of tuples
for i in range(1, 100):
    for j in range(i, 100):
        for k in range(j, 100):
            if i**2 + j**2 == k**2:
                unique_triples.append((i, j, k))

print(unique_triples)
```

```
[(3, 4, 5), (5, 12, 13), (6, 8, 10), (7, 24, 25), (8, 15, 17), (9, 12, 15), (9,
40, 41), (10, 24, 26), (11, 60, 61), (12, 16, 20), (12, 35, 37), (13, 84, 85),
(14, 48, 50), (15, 20, 25), (15, 36, 39), (16, 30, 34), (16, 63, 65), (18, 24,
30), (18, 80, 82), (20, 21, 29), (20, 48, 52), (21, 28, 35), (21, 72, 75), (24,
32, 40), (24, 45, 51), (24, 70, 74), (25, 60, 65), (27, 36, 45), (28, 45, 53),
(30, 40, 50), (30, 72, 78), (32, 60, 68), (33, 44, 55), (33, 56, 65), (35, 84,
91), (36, 48, 60), (36, 77, 85), (39, 52, 65), (39, 80, 89), (40, 42, 58), (40,
75, 85), (42, 56, 70), (45, 60, 75), (48, 55, 73), (48, 64, 80), (51, 68, 85),
(54, 72, 90), (57, 76, 95), (60, 63, 87), (65, 72, 97)]
```

9 9. (done) Normalization

Write a function that takes a tuple of numbers and returns it with the entries normalized to one

[27]:
```
def normalize_tuple (t : tuple) -> tuple:
    sum = 0
    for element in t:
```

```
        assert isinstance(element, (float, int)), "t must be a tuple of numbers"
        sum += element
    t_list_normalized = [element/sum for element in t]
    return tuple(t_list_normalized)

t = (1, 0, 3)
r = normalize_tuple(t)
print(r)
type(r)
```

```
(0.25, 0.0, 0.75)
```

[27]:
```
tuple
```