Laboratory 10

SOURCE

Exercise 1

```
Colors = ['red', 'green', 'blue', 'black', 'white']
# 1.
print(Colors[0])
# 2.
Different_type = False
Previous_type = type(Colors[0])
for Color in Colors:
    if Previous_type != type(Color):
        Different_type = True
    Previous_type = type(Color)
str = 'different' if Different_type else 'same'
print(f'List Colors contains {str} types of variables')
# 3.
print(len(Colors))
# 4.
Random_color = 'pink'
print(f'Color {Random_color} is in the list: {Random_color in Colors}')
# 5.
Zeros = [0] * 100
print(Zeros)
# 6.
Numbers = []
for Number in range(0, len(Colors),1): Numbers.append(Number)
Numbers_and_Colors = Colors + Numbers
print(Numbers_and_Colors)
# 7.
Earth = 'earth'
Week_days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday']
Modified_list = Numbers_and_Colors
Modified_list[0] = Earth
Modified_list.append(Week_days)
print(Modified_list)
# 8.
Modified_list.append('mercury')
```

```
output:
red
List Colors contains same types of variables
5
Color pink is in the list: False
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
['red', 'green', 'blue', 'black', 'white', 0, 1, 2, 3, 4]
['earth', 'green', 'blue', 'black', 'white', 0, 1, 2, 3, 4, ['Monday', 'Tuesday',
'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']]
```

Exercise 2

```
# 1.
Tuple = ('red', 'green', 'red')
# 2.
print('first element:', Tuple[0],
      'length: ', len(Tuple),
      'red count: ', Tuple.count('red'))
# 3.
print('index of \'red\'', Tuple.index('red'))
# 4.
# Tuple.append('blue') there is no append function for tuple class
# tuples are immutable
```

```
output:
first element: red length:  3 red count:  2
index of 'red' 0
```

Exercise 3

```
L1 = [1, 2, 3, 5]
L2 = L1 # defining by reference not by copy like in c++
print(L2)
print(id(L1))
print(id(L2)) # both variable are refering to the same piece of memory
L2[2:] = [0, 0, 0, 0, 0, 0] # by mutating one variable changes are seen in both
variable
print(id(L2))
print(L1)
print(L2)

print('\n ---------------- ')
L1 = [1, 2, 3, 5] # overwriting variable with const assignes new id/adress in
memory
L2 = L1
print(id(L1))
print(id(L2))
L2 = [0, 0, 0, 0, 0] # overwriting variable that is refering to different variable
"breaks" reference and assignes new id
```

```
print(L1)
print(L2)
print(id(L2))
```

```
output:
[1, 2, 3, 5]
140014721460672
140014721460672
140014721460672
[1, 2, 0, 0, 0, 0, 0, 0]
[1, 2, 0, 0, 0, 0, 0, 0]

 ----------------
140014721794816
140014721794816
[1, 2, 3, 5]
[0, 0, 0, 0, 0]
140014721460672
```

Exercise 4

```
T1 = (1, 2, 3)
print('T2 = T1')
T2 = T1
print('addresses of the T1 and T2:')
print(id(T1))
print(id(T2), '\n')

L1 = [1, 2, 3, 5]
T1 = (1, 2, 3)
print('address of the L1:')
print(id(L1), '\n')
print('address of the T1:')
print(id(T1), '\n')
print('T2 = L1')
T2 = L1
print('address of the T2')
print(id(T2), '\n')
# because Python is dynamicly typed language so variable that were previously
diffrenet type can reference to different type variables
```

```
output:
T2 = T1
addresses of the T1 and T2:
140680639985088
140680639985088
```

```
address of the L1:
140680639976896

address of the T1:
140680639985088

T2 = L1
address of the T2
140680639976896
```

Exercise 5

```
n = 5
Dictionary = dict()
for i in range(0,n):
    Dictionary.update({i:i*i})
print(Dictionary)
for i in Dictionary:
    print(type(i))
```

```
output:
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
<class 'int'>
<class 'int'>
<class 'int'>
<class 'int'>
<class 'int'>
```

Exercise 6

```
Numbers = []
for Number in range(2000,3200):
    if Number%7==0 and Number%5!=0:
        Numbers.append(Number)
print(Numbers)
```

```
output:
[2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 2107,
2114, 2121, 2128, 2142, 2149, 2156, 2163, 2177, 2184, 2191, 2198, 2212, 2219,
2226, 2233, 2247, 2254, 2261, 2268, 2282, 2289, 2296, 2303, 2317, 2324, 2331,
2338, 2352, 2359, 2366, 2373, 2387, 2394, 2401, 2408, 2422, 2429, 2436, 2443,
2457, 2464, 2471, 2478, 2492, 2499, 2506, 2513, 2527, 2534, 2541, 2548, 2562,
2569, 2576, 2583, 2597, 2604, 2611, 2618, 2632, 2639, 2646, 2653, 2667, 2674,
```

```
2681, 2688, 2702, 2709, 2716, 2723, 2737, 2744, 2751, 2758, 2772, 2779, 2786,
2793, 2807, 2814, 2821, 2828, 2842, 2849, 2856, 2863, 2877, 2884, 2891, 2898,
2912, 2919, 2926, 2933, 2947, 2954, 2961, 2968, 2982, 2989, 2996, 3003, 3017,
3024, 3031, 3038, 3052, 3059, 3066, 3073, 3087, 3094, 3101, 3108, 3122, 3129,
3136, 3143, 3157, 3164, 3171, 3178, 3192, 3199]
```

Exercise 7

```
def myfun1(var_inside):
    var_inside = 2
    print("var_inside myfun1:" + str(var_inside))

def myfun2(list_inside):
    list_inside[2] = 0
    print("list_inside myfun2:" + str(list_inside))

def myfun3(list_inside):
    new_list = list_inside.copy()
    list_inside[3] = 0
    print("list_inside myfun3:" + str(list_inside))
    return new_list

var_outside = 1
myfun1(var_outside)
print("var_outside:" + str(var_outside))
# variable passed to function is copied to different internal variable for this
function and origianl variable is not mutated

listA_outside = [0,1,2,3,4,5,6,7]
listB_outside = listA_outside

print("listA_outside before:" + str(listA_outside))
myfun2(listA_outside)
print("listA_outside after:" + str(listA_outside))
print("listB_outside after:" + str(listB_outside))
# lists are passed to function by reference therefore original function was
mutated

listC_outside = myfun3(listA_outside)
print("listA_outside after:" + str(listA_outside))
print("listC_outside after:" + str(listC_outside))
# listC is not mutated because it is copy of listA and is not refering to listA
which was mutated
```

```
output:
var_inside myfun1:2
var_outside:1
listA_outside before:[0, 1, 2, 3, 4, 5, 6, 7]
list_inside myfun2:[0, 1, 0, 3, 4, 5, 6, 7]
```

```
listA_outside after:[0, 1, 0, 3, 4, 5, 6, 7]
listB_outside after:[0, 1, 0, 3, 4, 5, 6, 7]
list_inside myfun3:[0, 1, 0, 0, 4, 5, 6, 7]
listA_outside after:[0, 1, 0, 0, 4, 5, 6, 7]
listC_outside after:[0, 1, 0, 3, 4, 5, 6, 7]
```

Exercise 8.1

```
def convert_temperature(degrees, unit):
    degrees = int(degrees)
    if unit == 'F':
        temperature = (degrees -32) * 5/9
        converted_unit = 'C'
    else:
        temperature = 9/5 * degrees + 32
        converted_unit = 'F'
    print(str(degrees)+unit, '=', str(int(round(temperature,0)))+converted_unit)
    return (temperature, converted_unit)

def split_temperature(temperature):
    if temperature[-1] != 'C' and temperature[-1] != 'F':
        print('Wrong unit')
        return 0
    unit = temperature[-1]
    degrees = temperature[:-1]
    return (degrees, unit)


test_temperature1 = '21C'
test_temperature2 = '70F'
test_temperature3 = '2137F'

(degrees1, unit1) = split_temperature(test_temperature1)
(degrees2, unit2) = split_temperature(test_temperature2)
(degrees3, unit3) = split_temperature(test_temperature3)

convert_temperature(degrees1, unit1)
convert_temperature(degrees2, unit2)
convert_temperature(degrees3, unit3)
```

```
output:
21C = 70F
70F = 21C
2137F = 1169C
```

Exercise 8.2

```
# winter -
def current_season(day, month):
    if day > 31 or day < 1 or month > 12 or month < 1:
        print('you did something wrong')
        return 0
    if month == 3:
        if  day > 19: return 'spring'
        else: return 'winter'
    if month == 6:
        if day > 20: return 'summer'
        else: return 'spring'
    if month == 9:
        if day > 22: return 'autumn'
        else: return 'summer'
    if month == 12:
        if day > 21: return 'winter'
        else: return 'autumn'

test_days = [19, 20, 21, 22, 23]
test_months = [3, 6, 9, 12]
for month in test_months:
    for day in test_days:
        print(str(day)+'.'+str(month),current_season(day, month))
```

```
output:
19.3 winter
20.3 spring
21.3 spring
22.3 spring
23.3 spring
19.6 spring
20.6 spring
21.6 summer
22.6 summer
23.6 summer
19.9 summer
20.9 summer
21.9 summer
22.9 summer
23.9 autumn
19.12 autumn
20.12 autumn
21.12 autumn
22.12 winter
23.12 winter
```