10/10

Group_08_Exercice_00

November 24, 2020

1 Exercice 00

1.1 1 Numbers

1.1.1 1.a. What is the *type* of the result of the expression 3 + 1.5 + 4? (without typing code)

type float

1.1.2 1.b. How do you get it with code? (method?)

[3]:
```
# get the type of the result from 1.a
type(3+1.5+4)
```

[3]:
```
float
```

1.1.3 1.c. Ask the user for an iput and then save to input to an integer called "user_in" and then print the value multiplied by 5.

[21]:
```
# value multiplied by 5
user_in = int(input("please enter an integer"))
print(user_in * 5)
```

```
please enter an integer 34262
```

```
171310
```

1.1.4 1.d. Ask the user for an iput and then save to input to an integer called "square_root_value" and calculate the square_root of the number from the user

[117]:
```
# Square root

square_root_value = int(input("Please enter a number"))
print(square_root_value**0.5)
```

```
Please enter a number 36
```

```
6.0
```

1.1.5 1.e. Ask the user for an iput and then save to input to an integer called "square_value" and calculate the square of the number from the user

[28]:
```
# Square
square_value = int(input("enter a value"))
print(square_value * square_value)
```

```
enter a value 20
```

```
400
```

1.2 2 Strings

1.2.1 2.a. Given the string 'hello' give an index command that returns 'e'. Enter your code in the cell below:

[29]:
```
greeting = 'hello'
# Print out 'e' using indexing

greeting[1]
```

[29]:
```
'e'
```

1.2.2 2.b. Given the string 'hello' give an index command that returns 'hell'. Enter your code in the cell below:

[51]:
```
greeting = 'hello'
# Print out 'hell' using indexing

greeting[0:4]
```

[51]:
```
'hell'
```

1.2.3 2.c Given the string 'hello', create a new string variable called 'greeting_rest' from it to and save 'llo' in the new variable

[56]:
```
greeting = 'hello'
# Save the part 'llo' in a new variable called 'greeting_rest' using indexing

greeting_rest =  greeting[-3:]

greeting_rest
```

[56]:
```
'llo'
```

1.2.4 2.d. Ask the user for his or her name and then save the input to a variable named "user_name". Then print "Hello, *user_name* !"

[57]:
```
user_name = input("Enter your Name")

print(f"Hello, {user_name} !")
```

```
Enter your Name Krishna

Hello, Krishna !
```

1.2.5 2.e. Ask the user for his or her 'first_name', 'last_name' and 'age' and print the reust in a multi-line string like:

'Hello, first_name last_name.

You are age years old. '

[63]:
```
# hint: 3 inputs => 3 variables

first_name = input("Enter your first name")
last_name = input("Enter your last name")
age = int(input("Enter your age"))

print(f"'Hello, {first_name} {last_name}. \n\nYou are {age} years old.'")
```

```
Enter your first name Goutham Krishna
Enter your last name Munaga
Enter your age 27

'Hello, Goutham Krishna Munaga.

You are 27 years old.'
```

1.3 3. List

1.3.1 3.a Create a list with 4 elements "45,25,56" in two differents way and save it to a variable called 'my_list'

[77]:
```
# my_list =

my_list = [45,25,56,36] # first method to create desired list
print(my_list)

my_list = [45,25,56, "Moin"] #An example to create a list with multiple␣
 ↪datatypes
print(my_list)

my_list = [] # Another way to create desired list, append methdod could also be␣
 ↪used to create desired list
```

```
my_list = my_list + [45, 25]
my_list = my_list + [56,36]

print(my_list)

my_list = [45] #  append methdod could also be used to create a desired list
my_list.append(25)
my_list.append(56)
my_list.append(36)

print(my_list)
```

```
[45, 25, 56, 36]
[45, 25, 56, 'Moin']
[45, 25, 56, 36]
[45, 25, 56, 36]
```

1.3.2 3.b. From 'my_list' change the first value (index 0) to 0.

[78]:
```
# index 0 must be 0

my_list[0] = 0

print(my_list)
```

```
[0, 25, 56, 36]
```

1.3.3 3.c. Save the sum of all number in the list to a variable called 'sum_of_my_list'

[81]:
```
# sum of 0,25,56

sum_of_my_list = sum(my_list)

print(sum_of_my_list)
```

```
117
```

1.3.4 3.d. sort the list bellow:

[87]:
```
list1 = [4,5,6,3,6,7,2,9]

list1.sort()
print(list1)
```

```
[2, 3, 4, 5, 6, 6, 7, 9]
```

1.3.5 3.e. Get the last 3 elements of the list using indexing and save it to a variable called 'list2'. Then make again the sum of 'list2' and insert the result to 'list2'

[92]:
```
# hint: you might use 3 differents variables

list2 = list1[-3:]

print(list2)

list2 + [sum(list2)]

print(list2 + [sum(list2)])
```

```
[6, 7, 9]
[6, 7, 9, 22]
```

1.3.6 3.f. swap list elements

Swap the first and last elements from the list `one_to_five`

[97]:
```
# create list
one_to_five = [5,2,3,4,1]
first = one_to_five[0]
one_to_five[0] = one_to_five[-1]
one_to_five[-1] = first

print(one_to_five)
```

```
[1, 2, 3, 4, 5]
```

1.4 4. Dictionaries

Using keys and indexing, grap the word *Bremerhaven* from the following dictionaries:

[98]:
```
name = {'university':'Bremerhaven'}
# Get 'Bremerhaven'

name["university"]
```

[98]:
```
'Bremerhaven'
```

[107]:
```
name = {'institution':{'name':'Bremerhaven'}}
# Get 'Bremerhaven'

name["institution"]["name"]
```

[107]:
```
'Bremerhaven'
```

[115]:
```
name = {'region':[{'University':'Oldenburg','Hochschule':'Bremerhaven'}]}
# Get Bremerhaven
name['region'][0]['Hochschule']
```

[115]:
```
'Bremerhaven'
```

1.5 5. What is the major difference between tuples and lists?

Tuples are immutable objects but lists are mutable

1.6 6. Sets

1.6.1 6.a. What is unique about a set?‘

the set list is unordered, so the result will display the items in a random order.

-Set only stores a value once even if it is inserted more then once

1.6.2 6.b. Use a set to find the unique values of the list below:

[128]:
```
# create the list
unsorted_list = [1,2,2,1,3,5,4,8,7,74,8,8,9,9,5,4,45,12,4,2]

x = set(unsorted_list)

print(x)
```

```
{1, 2, 3, 4, 5, 7, 8, 9, 74, 12, 45}
```

1.7 6. Boolean

What will be the value of the following boolean?

[99]:
```
4**0.5 != 2
```

[99]:
```
False
```

[119]:
```
a = 1 < 4
a
```

[119]:
```
True
```

[120]:
```
b = 'b' < 'c'
b
```

[120]:
```
True
```

[125]:
```
c = (a == b)
c
```

[125]: True

[126]:
```
d = (c or False)
d
```

[126]: True

[127]:
```
e = (c and False) # equivalent to 'e=((a==b) and False)' <=>␣
↪'e=(((1<4)==('b'<'c')) and False'
e
```

[127]: False

[ ]:
```

```