Problem Set 1 - Python coding exercises with solutions

Some basic commands

Let us look at interactive input und output in Python:

1. Write code that asks the user for her name and then prints, "Hello name!".

```
In [1]: name = input("Enter your name: ")
```

```
Enter your name: Anna
```

```
In [2]: print("Hello", name+"!")
```

```
Hello Anna!
```

Data types

2. What are the data types of the following:

a) 7

b) 3.1415

c) [0, 2, -1, 'r']

d) ('GER', 'FRA', 12.9, 'ITA')

e) 3e-2 (also print this object)

f) 0b110 (also print this object)

g) 0 in (-2, 5, 0, 3) (also print this object)

h) {'Country': 'Germany', 'GDP': '4200', 'Population': '83'}

```
In [3]: type(7)

Out[3]: int

In [4]: type(3.1415)

Out[4]: float

In [5]: type([0, 2, -1, 'r'])

Out[5]: list

In [6]: type(('GER', 'FRA', 12.9, 'ITA'))

Out[6]: tuple

In [7]: print(3e-2)

        0.03

In [8]: type(3e-2)

Out[8]: float

In [9]: print(0b110)

        6

In [10]: type(0b110)

Out[10]: int

In [11]: print(0 in (-2, 5, 0, 3))

         True

In [12]: type(0 in (-2, 5, 0, 3))

Out[12]: bool

In [13]: type({'Country':'Germany', 'GDP': '4200', 'Population':'83'})

Out[13]: dict

In [ ]:
```

3. Define x = 10. What is the output of the following:

a) type(x)

b) type(str(x))

c) print(x + 4)

d) print(str(x)+4)

e) print(int(str(x))-2)

```
In [14]: x=10
         type(x)
Out[14]: int
```

```
In [15]: type(str(x))
Out[15]: str
```

```
In [16]: print(x+4)
```

```
14
```

```
In [17]: print(str(x)+4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 print(str(x)+4)

TypeError: can only concatenate str (not "int") to str
```

```
In [19]: print(int(str(x)) - 2 )
```

```
8
```

4. Define x=3. What is the output of print(type(str(x)))?

```
In [20]: print(type(str(x)))
```

```
<class 'str'>
```

5. What is the output of print(7>3)?

```
In [21]: print(7>3)
```

```
True
```

6. What is the output of print(type(7>3))?

In [22]:
```
print(type(7>3))
```
```
<class 'bool'>
```

Operators

Operators are symbols in Python that can be used to conduct math or logical computation

Arithmetic

7. Define x=7. And y=2. What is the output from the following commands? First try to think about what might be the output, then verify by running the code

In [2]:
```
x=7
y=2
```

a) print('x+y=', x+y)

In [3]:
```
print('x+y=', x+y)
```
```
x+y= 9
```

b) print('x-y=', x-y)

In [4]:
```
print('x-y=', x-y)
```
```
x-y= 5
```

c) print('x/y=', x/y)

In [5]:
```
print('x/y=', x/y)
```
```
x/y= 3.5
```

d) print('x* y=', x* y)

In [6]:
```
print('x* y=', x* y)
```
```
x* y= 14
```

e) print('Floor division of x/y yields: ', x//y)

In [8]:
```
print('Floor division of x/y yields: ', x//y)
```
```
Floor division of x/y yields:  3
```

f) print('x/y: the rest (modulo) is: ', x%y)

```
In [9]: print('x/y: the rest (modulo) is: ', x%y)
```

```
x/y: the rest (modulo) is:  1
```

g) print('x raised to the power of y is: ', x**y)

```
In [11]: print('x raised to the power of y is: ', x**y)
```

```
x raised to the power of y is:  49
```

8. Write code that prints 'ha ' 3 times using the * operator

```
In [12]: print('ha '*3)
```

```
ha ha ha
```

Comparison

9. Think about what might be the output from each of the following commands and then verify by running the code:

a) 6==6

```
In [13]:  6==6
```

```
Out[13]: True
```

b) 6==7

```
In [14]: 6==7
```

```
Out[14]: False
```

c) 6!=7

```
In [15]: 6!=7
```

```
Out[15]: True
```

d) 6>7

```
In [16]: 6>7
```

```
Out[16]: False
```

e) 6<=7

```
In [17]: 6<=7
Out[17]: True
```

Logical

10. Define x= True. And y = False. What is the output from each of the following commands? Verify:

```
In [18]: x=True
         y=False
```

a) type(x)

```
In [19]: type(x)
Out[19]: bool
```

b) x and y

```
In [20]:  x and y
Out[20]: False
```

c) x or y

```
In [21]: x or y
Out[21]: True
```

d) not x

```
In [22]: not x
Out[22]: False
```

e) (not x) and y

```
In [23]: (not x) and y
Out[23]: False
```

Assignment

11. Define m=3, and n=5. What is the output from each of the following commands? Verify and print the results on the screen:

```
In [25]: m=3
         n=5
```

a) m=m+3

```
In [26]: m=m+3
```

```
In [27]: print(m)
```

```
6
```

```
In [28]: m=m+3
```

```
In [29]: print(m)
```

```
9
```

b) n += 3

```
In [30]: n += 3
```

```
In [31]: print(n)
```

```
8
```

Membership

12. Define s= "Python is great fun!" And l=[23, 'apple', 0.6]. What is the output from each of the following commands? Verify?

```
In [33]: s= "Python is great fun!"
         l=[23, 'apple', 0.6]
```

a) 'y' in s

```
In [34]:  'y' in s
```

```
Out[34]: True
```

b) "G" not in s

```
In [35]: "G" not in s
Out[35]: True
```

c) 2 in l

```
In [37]: 2 in l
Out[37]: False
```

d) 0.8 not in l

```
In [38]: 0.8 not in l
Out[38]: True
```

Conditionals (if ... else)

The if, and the if...else constructs are used to conditionally implement code. These are very useful in functions and programs. Notice the indenting of the code below. This is an important feature of Python. It indicates the assignment of programming lines. If you do not indent correctly, an error message will show up, or the codes runs differently from what you intended it to do. Moreover, indenting enhances readability.

13. What is the output of the following commands? Verify:

```
In [39]: x = 3
         if x > 0:
             print(x, 'is positive')

3 is positive
```

```
In [40]: x = 3
         if x > 0:
         print(x, 'is positive')

  Cell In[40], line 3
    print(x, 'is positive')
    ^
IndentationError: expected an indented block after 'if' statement on line 2
```

In [41]:
```python
x = 3
l = [0, 7, -9, 3]
if x in l:
    print(x, "belongs to the list ", l)
```

```
3 belongs to the list  [0, 7, -9, 3]
```

In [42]:
```python
x = 3
if True:
    x += 1
print(x)
```

```
4
```

In [43]:
```python
x = 3
if True:
    x -= 1
    print(x)
```

```
2
```

In [44]:
```python
x = 3
if 0:
    x -= 1
print(x)
```

```
3
```

In [45]:
```python
x = 3
if 0:
    x -= 1
    print(x)
```

In [47]:
```python
x=5
y=10
if y%x==0:
    print(x, 'is a divisor of', y)
else:
    print(x, 'is not a divisor of', y)
```

```
5 is a divisor of 10
```

In [48]:
```python
x=3
y=7
if y%x==0:
    print(x, 'is a divisor of', y)
else:
    print(x, 'is not a divisor of', y)
```

```
3 is not a divisor of 7
```

Loops

Loops are common and super useful programming constructs. They allow for automation of repititive tasks. A "for" loop has the syntax:

for x in sequence: code

It iterates over each value of the sequence (e.g. a list) and implements the code in each interation. Loops are best understood by working through examples. In the following questions, we use the "range" function to generate the sequence. Look up this function online on python.org to understand what it does.

14. What is the output of the following commands? Verify

```
a) x = int(input("Enter a number: "))
for i in range(x):
    fact = 1
    fact += i
    print(fact)
```

In [50]:
```python
x = int(input("Enter a number: "))
for i in range(x):
    fact = 1
    fact += i
    print(fact)
```

```
Enter a number: 4
1
2
3
4
```

```
b) x = int(input("Enter a number: "))
for i in range(x):
    fact = 1
    fact += i
print(fact)
```

In [51]:
```python
x = int(input("Enter a number: "))
for i in range(x):
    fact = 1
    fact += i
print(fact)
```

```
Enter a number: 4
4
```

```
c) x = int(input("Enter a number: "))
for i in range(x):
    fact = 1
    fact *= i
    print(fact)
```

In [52]:
```python
x = int(input("Enter a number: "))
for i in range(x):
    fact = 1
    fact *= i
    print(fact)
```

```
Enter a number: 4
0
1
2
3
```

d) x = int(input("Enter a number: "))
for i in range(x):
    fact = 1
    fact *= *i*
print(fact)

In [53]:
```python
x = int(input("Enter a number: "))
for i in range(x):
    fact = 1
    fact *= i
print(fact)
```

```
Enter a number: 4
3
```

e) x = int(input("Enter a number: "))
for i in range(x+1):
    fact = 1
    fact *= *i*
print(fact)

In [54]:
```python
x = int(input("Enter a number: "))
for i in range(x+1):
    fact = 1
    fact *= i
print(fact)
```

```
Enter a number: 4
4
```

f) x = int(input("Enter a number: "))
fact = 1
for i in range(x):
    fact *= *i*
print(fact)

```
In [55]: x = int(input("Enter a number: "))
         fact = 1
         for i in range(x):
             fact *= i
         print(fact)
```

```
Enter a number: 4
0
```

```
g) x = int(input("Enter a number: "))
fact = 1
for i in range(x+1):
    fact *= i
print(fact)
```

```
In [56]: x = int(input("Enter a number: "))
         fact = 1
         for i in range(x+1):
             fact *= i
         print(fact)
```

```
Enter a number: 4
0
```

```
h) x = int(input("Enter a number: "))
fact = 1
for i in range(1, x+1):
    fact *= i
print(fact)
```

```
In [57]: x = int(input("Enter a number: "))
         fact = 1
         for i in range(1, x+1):
             fact *= i
         print(fact)
```

```
Enter a number: 4
24
```

15. What is the output of the following command? Verify:

s = 'Daniel'

for i in s:

```
    print(i)
```

In [61]:
```python
s = 'Daniel'

for i in s:

    print(i)
```

```
D
a
n
i
e
l
```

16. Write code that asks the user for a positive number (n) and then prints the sum of all even numbers between 0 and n.

In [62]:
```python
x = int(input("Enter a positive number: "))
sum = 0
for i in range(1, x+1):
    if i%2==0:
        sum += i
print(sum)
```

```
Enter a positive number: 12
42
```

17. For can also be combined with else. In that case, the loop iterates over the sequence until all items in the sequence are exhausted and then the else bit is executed. What is the output of the following code?

```
name = 'Daniela'
for i in name:
    print(i)
else:
    print("Name zuende buchstabiert")
```

In [63]:
```python
name = 'Daniela'
for i in name:
    print(i)
else:
    print("Name fully spelt out")
```

```
D
a
n
i
e
l
a
Name fully spelt out
```

18. The break statement can be used to add further control to a loop. To see how this can be useful, examine the following code. Notice that the else is part of the nested for loop (same indentation).

```
r = 1
x = 50
for num in range(r, x+1):
    if num > 1:
        for i in range(2, num):
            if(num%i) == 0:
                break
        else:
            print(num)
```

In [64]:
```
r = 1
x = 50
for num in range(r, x+1):
    if num > 1:
        for i in range(2, num):
            if(num%i) == 0:
                break
        else:
            print(num)
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
```

Lists

Lists are ordered collections of objects. A list is thus a sequence. The objects can be the same or mixed data types. Other Python data types that are sequences include strings and tuples.

19. What is the output of the following commands? Verify

Note that index (position) numbers in a Python list start from 0 !

In [65]:
```
list1 = [4, 2, 7.5, 3]
list2 = [[4, 5, 6], "Python", "Java"]
```

a) list1[0]

In [66]:
```
list1[0]
```
Out[66]:
```
4
```

b) list1[2]

In [67]:
```
list1[2]
```
Out[67]:
```
7.5
```

c) list2[0]

In [68]:
```
list2[0]
```
Out[68]:
```
[4, 5, 6]
```

d) list2[2][0]

In [69]:
```
list2[2][0]
```
Out[69]:
```
'J'
```

e) list1[-2]

In [70]:
```
list1[-2]
```
Out[70]:
```
7.5
```

f) list2[-2][-2]

In [71]:
```
list2[-2][-2]
```
Out[71]:
```
'o'
```

```
g) list2[0] = "R"
print(list2)
```

In [72]:
```
list2[0] = "R"
print(list2)
```
```
['R', 'Python', 'Java']
```

```
h) name = "Daniela Schmidt"
```

```
print(list(name))
```

In [73]:
```
name = "Daniela Schmidt"
print(list(name))
```

```
['D', 'a', 'n', 'i', 'e', 'l', 'a', ' ', 'S', 'c', 'h', 'm', 'i', 'd', 't']
```

20. We can add and delete items. Have a look what the insert() and extend() methods do in the code below. Notice the syntax - objects such as this list belong to the class list and this class, like many others in Python comes with built-in functions or methods. Also, note that del is a inbuilt function to remove a part of an object or the entire object e.g. list etc from memory.

In [74]:
```
list1 = [4, 3.8, 2, 9]
list1.insert(1, -1)
print(list1)
```

```
[4, -1, 3.8, 2, 9]
```

In [75]:
```
list1 = [4, 3.8, 2, 9]
list1.insert(3, -1)
print(list1)
```

```
[4, 3.8, 2, -1, 9]
```

In [76]:
```
list1 = [4, 3.8, 2, 9]
list1.extend([1, -1])
print(list1)
```

```
[4, 3.8, 2, 9, 1, -1]
```

In [77]:
```
list1 = [4, 3.8, 2, 9]
del list1[0]
print(list1)
```

```
[3.8, 2, 9]
```

In [78]:
```
list1 = [4, 3.8, 2, 9]
del list1[:]
print(list1)
```

```
[]
```

```
In [79]: list1 = [4, 3.8, 2, 9]
         del list1
         print(list1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[79], line 3
      1 list1 = [4, 3.8, 2, 9]
      2 del list1
----> 3 print(list1)

NameError: name 'list1' is not defined
```

21. Python offers many handy programming shortcuts. One such is "list comprehensions". They simplify conditionals and loops. What is the output of the following commands? Verify

```
In [80]: list = [2*x for x in range(5)]
         print(list)
```

```
[0, 2, 4, 6, 8]
```

Using "list comprehension", construct the following lists:

a) The list of cubes of integers from 0 to 9.

b) The list of odd powers of 3, starting from 1 and running up to and including 9.

```
In [81]: list1=[x**3 for x in range(10)]
         print(list1)
```

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

```
In [82]: list2=[3**i for i in range(10) if i%2==1]
         print(list2)
```

```
[3, 27, 243, 2187, 19683]
```