Python

Exercises

The code must be submitted under your name in GitHub in a repository called Python. Work individually.

Each file will have the name: exerciseX.py where X is the exercise number. You will have 13 files at most.

Do not commit code that does not compile. The code that you commit should have been tested. -10 points for code that does not compile on the top of your grade.

You will provide a hardcopy with your code to Dr. Scharff on 12/6.

Exercise 1

Explain the output of the following statements:

(a) `5 / 3`

(b) `5 % 3`

(c) `5.0 / 3`

(d) `5 / 3.0`

(e) `5.2 % 3`

a) Divides the numbers and prints the result as a decimal
   i) Result: 1.6666666666666667
b) Gives you the remainder of dividing the two numbers. In this case, the result is a whole number.
   i) Result: 2
c) Divides the numbers and prints the result as a decimal (mix of float and int returns float)
   i) Result: 1.6666666666666667
d) Divides the numbers and prints the result as a decimal (mix of float and int returns float)
   i) Result: 1.6666666666666667
e) Gives you the remainder of dividing the two numbers. In this case, the result is a decimal due to the decimal in the problem.
   i) Result: 2.2

Exercise 2

Explain the output of the following statements:

(a) `2000.3 ** 200` (compare with above)

(b) `1.0 + 1.0 - 1.0`

(c) `1.0 + 1.0e20 - 1.0e20`

a) Attempting this operation results in an OverflowError. The result is too large.
   i) Result:

```
----------------------------------------
OverflowError                Traceback (most recent call last)
<ipython-input-6-11e960b1ae3d> in <module>()
----> 1 2000.3 ** 200

OverflowError: (34, 'Result too large')
```

b) Returns the result in decimal form
   i) Result: 1.0

c) Returns a 0.0 result

Exercise 3

Very often, one wants to "cast" variables of a certain type into another type. Suppose we have variable `x = '123'`, but really we would like `x` to be an integer.

This is easy to do in Python, just use `desiredtype(x)`, e.g. `int(x)` to obtain an integer.

Try the following and explain the output

(a) `float(123)`

(b) `float('123')`

(c) `float('123.23')`

(d) `int(123.23)`

(e) `int('123.23')`

(f) `int(float('123.23'))`

(g) `str(12)`

(h) `str(12.2)`

(i) `bool('a')`

(j) `bool(0)`

(k) `bool(0.1)`

a) Because it is of type float, 123 is appended with a .0
   i) result: 123.0
b) This is a float string. 123 is still appended with .0 however future conversion from a string to a float would be necessary to go to an int.
   i) result: 123.0
c) This is a float string. 123.23 is still a float however future conversion from a string to a float would be necessary to go to an int.
   i) result: 123.23
d) Because this of type int, decimals are ignored. int cannot hold them.
   i) result: 123
e) ValueError because '123.23' can't be directly converted from a string to a float to an int.
   i) result:

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-5-5e250fe2a145> in <module>()
----> 1 int('123.23')

ValueError: invalid literal for int() with base 10: '123.23'
```

f) Gets around the previous problem by placing `float('123.23')` inside an int. Step-by-step rather than trying directly.
   i) result: 123
g) Type is string so 12 is printed (in single quotes). It is not interpreted as a number.
   i) result: '12'
h) Type is string so 12.2 is printed (in single quotes). It is not interpreted as a number.
   i) result: '12.2'
i) The value returns true because all boolean values except zero are true
   i) result: true
j) The value is zero so it results in false
   i) result: false
k) The value returns true because all boolean values except zero are true
   i) result: true

Exercise 4

Type `range(5)` in the interpreter, what does the interpreter return? So what does `for i in range(5)` mean?

Let's also find out whether the interpreter can help us understand the object '`range(5)`' better. Type `type(range(5))` in the interpreter.

The interpreter returns `range(0, 5).` `for in i range(5)` means to loop an action for five times - 0 to 4.

`type(range(5))` simply returns `range`.

Exercise 5

Use a `while` loop to find the first 20 numbers that are divisible by 5, 7 and 11, and print them Hint: store the number found so far in a variable.

Pseudo-code:

```
number found = 0
x = 11
while number found is less than 20:
    if x is divisible by 5, 7 and 11:
        print x
        increase number found by 1
    increase x by 1
```

```
y = float(0);
x = float(5);

while y < 20:
     if (x%5==0) and (x%7==0) and (x%11==0):
          print(x);
          print(x / 5);
          print(x / 7);
          print(x / 11);
          x = x + 1;
     y = y + 1;
```

Exercise 6

Don't print. Make a list.

(a) Write a function `is_prime(n)` that returns `True` only if n is prime.

(b) Note that apart from 2 and 3, all primes are of the form $6k \pm 1$ (though not all numbers of the form $6k \pm 1$ are prime of course). Using this, we can improve the computation time by a factor 3. Update your function to use this.

(c) Write a function that returns all primes up to n.

(d) Write a function that returns the first n primes.

Done in files

Exercise 7

(a) Write a function that prints the elements of a list

(b) Write a function that prints the elements of a list in reverse

(c) Write your own implementation of the `len` function that returns the number of elements in a list.

Done in files

Exercise 8

(a) Create a list `a` with some entries.

(b) Now set `b = a`

(c) Change `b[1]`

(d) What happened to `a`?

(e) Now set `c = a[:]`

(f) Change `c[2]`

(g) What happened to `a`?

Now create a function `set_first_elem_to_zero(l)` that takes a list, sets its first entry to zero, and returns the list.

What happens to the original list?

d) Both lists now hold the same entries when printed with the new value being in both a[1] and b[1].

g) Only c[2] was changed. a and b stayed the same.

The original list "a" doesn't change when set_first_elem_to_zero is used on "a".

Exercise 9

Consider having a list with lists as elements, e.g. `[[1,3], [3,6]]`.

Write a function that takes such a list, and returns a list with as elements the elements of the sublists, e.g. `[1, 3, 3, 6]`.

Done in files

Exercise 10

Use matplotlib

Plot the function

$$f(x) = \sin^2(x-2)e^{-x^2}$$

over the interval $[0, 2]$. Add proper axis labels, a title, etc.

Done in files

Exercise 11

Don't order the list. Multiply when possible.

Write two functions, one that uses iteration, and the other using recursion, that achieve the following: The input of the function is a list with numbers. The functions return the product of the numbers in the list.

Exercise 12

Write Fibonacci in Python

The Fibonacci sequence $\{F_i\}_i = 0^\infty$ starts with $F_0 = 0, F_1 = 1$. Every subsequent value in the sequence is the sum of the last elements in the sequence:

$$F_n = F_{n-1} + F_{n-2}$$

Done in files

Exercise 13

Write a Python program that extracts the email addresses of a file. An email file emails.txt is provided to test your program.

http://rubular.com/ is a site that can be useful to get familiar with regular expressions.

References

Stanford courses on Python https://web.stanford.edu/~schmit/cme193/exercises.html