EE2703 - Week 1

Siddarth Baruah <ee21b128@smail.iitm.ac.in>

February 4, 2023

0.1 Numerical types

[1]:
```
print(12 / 5)
```

```
2.4
```

[2]:
```
print(12 // 5)
```

```
2
```

[3]:
```
a=b=10
print(a,b,a/b)
```

```
10 10 1.0
```

0.2 Strings and related operations

[4]:
```
a = "Hello "
print(a)
```

```
Hello
```

[5]:
```
print(a+str(b))
```

```
Hello 10
```

0.3 Problem statement 1

Print out a line of 40 '-' signs (to look like one long line) Then print the number 42 so that it is right justified to the end of the above line Then print one more line of length 40, but with the pattern '--*-'

[6]:
```
for i in range(40):
    print("-", end="")
print('')
for i in range(38):
    print(" ", end="")
print("42")
for i in range(20):
    print("*-", end="")
```

```
----------------------------------------
                                      42
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
```

[7]:
```
print(f"The variable 'a' has the value {a} and 'b' has the value {b:>10}")
```

```
The variable 'a' has the value Hello  and 'b' has the value         10
```

[8]:
```
dictionary= {"EE2703": "Applied Programming Lab", "EE2003": "Computer␣
 ↪Organization", "EE5131": "Digital IC Design"}
ID= "ID"
Name= "Name"
print(f"{ID.ljust(10)}{Name.rjust(40)}")
for i in dictionary:
    print(f"{i.ljust(10)}{dictionary[i].rjust(40)}")
```

```
ID                                          Name
EE2703                   Applied Programming Lab
EE2003                     Computer Organization
EE5131                         Digital IC Design
```

1 Functions for general manipulation

1.1 Problem statement 2

Write a function with name 'twosc' that will take a single integer as input, and print out the binary representation of the number as output. The function should take one other optional parameter N which represents the number of bits. The final result should always contain N characters as output (either 0 or 1) and should use two's complement to represent the number if it is negative. Examples: twosc(10): 0000000000001010 twosc(-10): 1111111111110110 twosc(-20, 8): 11101100

Use only functions from the Python standard library to do this.

[9]:
```
def twosc(x, N):
    if(x==0):
        for i in range(N):
            print("0", end='')

    for i in range(1,N+1):
        print((x>>(N-i))&1, end="")
    print("")
```

[10]:
```
twosc(10,16)
twosc(-10,16)
twosc(-20, 8)
```

```
0000000000001010
1111111111110110
11101100
```

Here we are simply putting the number inside the function either negative or negative because, the python compiler itself takes the negative data as two's compliment and thus, it furthur doesn't need any algo to compliment.

2 List comprehensions and decorators

[11]:
```
# Explain the output you see below
[x*x for x in range(10) if x%2 == 0]
```

[11]:
```
[0, 4, 16, 36, 64]
```

It will return a array of only even numbers because of the if condition x%2==0

[12]:
```
# Explain the output you see below
matrix = [[1,2,3], [4,5,6], [7,8,9]]
[v for row in matrix for v in row]
```

[12]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

First element in matrix is selected and named as row and then , elements in row is selected one by one and printed as v'

2.1 Problem Statement

Define a function `is_prime(x)` that will return True if a number is prime, or False otherwise. Use it to write a one-line statement that will print all prime numbers between 1 and 100

[13]:
```
def is_Prime(n):
    if(n==1):
        return False
    if(n==2 or n==3 or n==5):
        return True
    if(n%2==0 or n%3==0 or n%5==0):
        return False
    for i in range(5, int(n**(0.5)), 6):
        if(n&i==0 or n&(i+2)==0):
            return False
    return True
```

In this function, firstly the number is checked if it is divisible by 2,3 and 5 or not.

After that,

a loop start with i = 5 with increament of +6. Then it checks the remainder with divisor as i and i+2. The loop goes on till i<= $x^{0.5}$.

[14]:
```
print([x for x in range(1,101) if is_Prime(x)])
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 49, 53, 59, 61, 67, 71,
73, 77, 79, 83, 89, 91, 97]
```

2.2 Problem Statement

Explain the output below

[15]:
```
def f1(x):
    return "happy " + x
def f2(f):
    def wrapper(*args, **kwargs):
        return "Hello " + f(*args, **kwargs) + " world"
    return wrapper
f3 = f2(f1)
print(f3("flappy"))
```

```
Hello happy flappy world
```

The Code Happens in a series of event

1_{st} ly when we enter f3(“flappy”) it is as same as writting f2(f1(”flappy)).

Now f1(“flappy”) runs so we get return “Happy Flappy” now this goes as argument to f2

in f2 we get, return “Hello” + f(*args, **kwargs) +“world”=> which is equicalent to “Hello happy flappy world” in our case

[16]:
```
# Explain the output below
@f2
def f4(x):
    return "nappy " + x

print(f4("5"))
print(f4("flappy"))
print(f4("flay"))
```

```
Hello nappy 5 world
Hello nappy flappy world
Hello nappy flay world
```

The second output is the result of calling f4(“flappy”). Here, the f2 function is used as a decorator, which means it is applied to the f4 function definition. The decorator syntax @f2 is equivalent to f4 = f2(f4). So, f4 is redefined as the wrapper function returned by f2 with f4 as the inner function. When we call f4(“flappy”), it is equivalent to calling “Hello” + “nappy” + “flappy” + ” world” which results in “Hello nappy flappy world”.

3 File IO

3.1 Problem Statement

Write a function to generate prime numbers from 1 to N (input) and write them to a file (second argument). You can reuse the prime detection function written earlier.

Firstly lets import csv

[17]:
```
import csv
```

[18]:
```
def write_primes(N, filename):
    n=[]
    infile=open("data.txt",'w',newline='')
    n.append([i for i in range(1,N+1) if is_Prime(i)])
    csv.writer(infile).writerow(n)
    infile.close()
```

3.2 Function

it first opens a file name data.txt and then writes the prime number from 2 to the given number.

[19]:
```
write_primes(100, "data.txt")
```

4 Exceptions

4.1 Problem Statement

Write a function that takes in a number as input, and prints out whether it is a prime or not. If the input is not an integer, print an appropriate error message. Use exceptions to detect problems.

[20]:
```
def check_prime():
    try:
        x = int(input('Enter a number: '))
        print(is_Prime(x))
    except:
        print("The input number is invalid")


check_prime()
```

```
Enter a number:  9.9
```

```
The input number is invalid
```

4.1.1 Explaination

In the function, a number is asked in the form of int. If the user inputs a invalid number or characters then the line

x= int(input('Enter a number:'))

will raise an error