EE2703 - Week 1

Aman Kumar EE21B013 <ee21b013@smail.iitm.ac.in>

February 4, 2023

0.0.1 How to run the Code file ?

Sol : The file in which the code is is a jupyter notebook (.ipynb) file which can be run directly in jupyter notebook

1 Document metadata

> *Problem statement: modify this document so that the author name reflects your name and roll number. Explain the changes you needed to make here. If you use other approaches such as LaTeX to generate the PDF, explain the differences between the notebook approach and what you have used.*

I have change the Notebook Metadata , In the edit menu there is a option to Edit Notebook Metadata I have changed the author name to : Aman Kumar EE21B013 < ee21b013@smail.iitm.ac.in >

2 Basic Data Types

Here we have a series of small problems involving various basic data types in Python. You are required to complete the code where required, and give *brief* explanations of your answers. Remember that the documentation and explanation is as important as the answer.

For each of the following cells, first execute them, and then give a brief explanation of why the answer comes out to be the way it does. If there is an error during execution of the cell, explain how you fixed it. **Add a new cell of type Markdown with the explanation** after the corresponding cell. If you are using plain Python, add suitable comments after each line and explain this in the documentation (clearly you would be better off using Notebooks here).

2.1 Numerical types

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

```
2.4
```

2.1.1 Explanation of above code :

In python we have two type of division one is *float division* and other is *integer division(floor division)* ignore the remainder. The above is float division.

[60]:
```
print(12 // 5)
```

2

2.1.2 Explanation of above code

This is *Integer division(floor division)* give only integer value by rounding down the **actual result** to the nearest integer less than or equal to the actual quotient

[11]:
```
a=b=10
print(a,b,a/b)
```

```
10 10 1.0
```

2.1.3 Explanation of above code :

We can do **Multiple assignment** in python, here we are doing the same **a=10** and **b=10** we have written these two statement in a single line using multiple assignment . We can also print more than one thing using print statement.

2.2 Strings and related operations

[12]:
```
a = "Hello "
print(a)
```

```
Hello
```

2.2.1 Explanation :

We can s*tore string in variables* and using that variable we can operate on string, Here we are print the string using that variable

2.2.2 Problem :

Output should contain "Hello 10" ##### CODE :

[14]:
```
b=str(b)

print(a+b)  # Output should contain "Hello 10"
```

```
Hello 10
```

2.2.3 Explanation Of above code:

ERROR : We can't concatenate **strings with integers Solution** : Changing the type of b from *integer to string* we can concatenate the a and b using **'+'** operator

2.3 Problem :

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 **'--*-'** #### CODE Output and Explanation Below:

[28]:
```
ch1='_'
print(40*ch1)
temp=42
print(f"{temp:>42}")#using fstring to format string
ch2='_*'
print(20*ch2)
```

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

2.3.1 Explanation :

I have used the string property in python we can use **addition** and **multiplication** to concatenate string

[42]:
```
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
```

2.3.2 Explanation :

Using fstring we can format fstring in the above code we are printing the value of a and b(with right aligned with 10 character gap)

2.3.3 Problem:

Create a list of dictionaries where each entry in the list has two keys:

- **id**: this will be the ID number of a course, for example 'EE2703'
- **name**: this will be the name, for example 'Applied Programming Lab'
- Add 3 entries: **EE2703 -> Applied Programming Lab EE2003 -> Computer Organization EE5311 -> Digital IC Design** Then print out the entries in a neatly formatted table where the ID number is left justified to 10 spaces and the name is right justified to 40 spaces. That is it should look like: EE2703 Applied Programming Lab EE2003 Computer Organization EE5131 Digital IC Design

CODE Output and Explanation Below:

[8]:
```
# Output should looks like :
# EE2703                    Applied Programming Lab
# EE2003                      Computer Organization
# EE5131                          Digital IC Design
CoursesStructure=[{'id':'EE2703','name':'Applied Programming Lab'},{'id':
 ↪'EE2003','name':'Computer Organization'},
        {'id':'EE5131','name':'Digital IC Design'}]
for course in CoursesStructure:
    id=course["id"]
    name=course["name"]
```

```
    print(f'{id:<10}{name:>40}')
```

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

2.3.4 Explanation of above code:

I have created a *list with name CourseStructure* and inside that i have created *three dictionary* in which in each dictionary i am storing two **keys : id and name** and using those i am printing the whole dictionary and for aligning i am using the string formatting syntax.

3 Functions for general manipulation

3.1 Problem :

- 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.

CODE Output and Explanation Below:

[39]:
```
#This function calculate the binary representation of a positive no and return␣
 ↪the binary representation of no in
#form of list
def find_bin(n):
    l1=[]
    while(n>0):
        l1.append(n&1)
        n=n//2
    l1.reverse()
    return l1


def twosc(k,N=16):
    x=bin(k) # getting the binary representation of k using standard library␣
 ↪function
    l=[]

    if k<0 :
        x=x[3:]
        if(N<=len(x)):
```

```
            print(f"we can't represent the given number in less than {len(x)+1}␣
↪binary digit")
            return
        #converting the binary to one's complement and inserting into list l
        for i in range(len(x)):
            if(x[i]=='1'):
                l+=[1]
            else:
                 l+=[0]

        for ele in range(len(l),N):#inserting value 0 for making the length of␣
↪list upto N
            l.insert(0,0)
        for i in range(len(l)):
            if(l[i]==1):
                l[i]=0
            else:
                 l[i]=1
        carry=1
        #converting into two's complement by adding carry =1 to the one's␣
↪complement
        for i in range(len(l)-1,-1,-1):
            if(carry==1):
                if(l[i]==1):
                    l[i]=0
                elif(l[i]==0):
                    l[i]=1
                    carry=0
            elif(carry==0):
                break

        for digit in l:
            print(digit,end='')
        print("")

    if(k>=0):
        x=x[2:]
        if(N<=len(x)):
            print(f"we can't represent the given number in less than {len(x)}␣
↪binary digit")
            return
        for i in range(len(x)):
            if(x[i]=='1'):
                l+=[1]
            else:
                 l+=[0]
        for ele in range(len(l),N):
```

```
            l.insert(0,0)
        for digit in l:
            print(digit,end='')
        print("")
twosc(15)
```

```
0000000000001111
```

3.1.1 Explanation of above code:

I have completed the function twosc which will take any integer input and prints out it's binary representation if the **Input number is negative number it will print it's two's complement**. I have used **bin()** function from standard library but I have also made a function **find_bin()** which will find the binary representation of a whole no and return it in form of list. In finding the binary representation of negative no I have firstly change the no into it's one's complement and than converted into two's complement by adding 1 to it's one's complement

4 List comprehensions and decorators

4.1 Problem :

Explain the output you see below

CODE Output and Explanation Below:

[40]:
```
[x*x for x in range(10) if x%2 == 0]
```

[40]:
```
[0, 4, 16, 36, 64]
```

4.1.1 Explanation :

The code above is iterating from **0 to 9(included) with step 1** and checking if the **number is even** then it append the **square of the number** in the list

4.2 Problem

Explain the output you see below

CODE Output and Explanation Below:

[41]:
```
matrix = [[1,2,3], [4,5,6], [7,8,9]]
[v for row in matrix for v in row]
```

[41]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

4.2.1 Explanation :

The above code is iterating for **each row and inside row for each element** of that row and appending each element

4.3 Problem

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 #### CODE Output and Explanation Below:

[2]:
```
def is_prime(n):
    ans=True
    # Checking if the no is less than or equal to 1 than it's not prime
    if(n<=1):
        print("Error! : Prime numbers are positive integers greater than 1")
        print("Enter a valid value")
        return False
    if(n==0 or n==1 or n<0): return False
    #looping from 2 to n and breaking the loop when when square(no) is greater␣
 ↪than n+1
    for no in range(2,n,1):
        if(no*no>n+1): #break the loop if we can't find the factor till root(n)␣
 ↪than that is prime no
            break
        if(n%no==0):  #checking if there is a factor
            ans=False
            break
    return ans
print([no for no in range(2,101) if is_prime(no)])
```

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

4.4 Explanation of above code

In function is_prime if number entered for check is less than or equal to **1** than **it's not a prime** and i will return from the function if it is greater than **1** than i am iterating no from **2 to n** and breaking the loop as "**' {square(no)} is greater than {n+1}**" because when there is no factor of **n less than or equal to sqrr(n)** than there is also no factor of that **n** from **sqrt(n) to n** and the moment I find a **factor of n** i am returning **false** from the fucntion otherwise it will return **true** as the **n will be prime no**

[6]:
```
# Explain the output below
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
```

4.4.1 Explanation of above code:

In the above code, we defined **two function f1(x) and f2(f)**.In python, we can store fucntion inside a variable, we can pass a function as argument to function we can also return a fucntion from a function, we are doing the same thing here we are refrencing the function f2(f1) by f3(storing under variable f3) and calling the variable f3 means we are calling the function f2 and inside we passed f1 as argument with "flappy" as argument so inside the f2 function there is also a function defined which will take as many arguments we will give using the *args and **kwargs and return the wrapper function which is defined inside the f2 which will take the arguments passed using *args and **kwargs and return "hello" + call the function which is passed inside the the function f2 as an argument with the arguments *args and **kwargs+"world",The function f1 return "happy"+the argument passed inside it (here that is "flappy").so finally, the function f2 will return** "Hello happy flappy world"** which will get printed.

4.4.2 Explanation of above code :

[7]:
```
# Explain the output below
@f2
def f4(x):
    return "nappy " + x

print(f4("flappy"))
```

```
Hello nappy flappy world
```

4.4.3 Explanation of above code :

These are **decorators**, In decorators functions are taken as argument inside function and then called inside another function,it will add some functionality to the function without changing the original function permanently,it will wrap the function inside another function.Here f2 is wrapper function and f4 is wrapped function, i.e. whenever f4 is called It will be passed inside f2 as wrapped function (f2(f4("flappy")), here when f4 is called with "flappy" as argument, It will go inside f2 where wrapper function is called inside that wrapper function function f4 will be called and then finally the wrapper function will return **"Hello nappy flappy world"** which will be returned by f2 and will got printed.

5 File IO

5.1 Problem

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. #### CODE Output and Explanation Below:

[3]:
```
#function to write the the prime number upto 'N' to the file given to the␣
 ↪fucntion as input
def write_primes(N, filename):
    lst_prime_no=[]
    for num in range(2,N+1):
```

```
        if (is_prime(num)):
           lst_prime_no.append(num)
    with open(filename,'w') as f:    #opening filename in write mode to write␣
 ↪the prime no upto N
        for ele in lst_prime_no:
            f.write(str(ele))
            f.write("  ")
write_primes(101,'untitled.txt')
```

5.1.1 Explanation of above code:

The fucntion takes two input **numebr and filename** and write all the **prime no upto the given vlaue of number(N) to the file(filename)**. I am Iterating from **2 to N(included)** and checking if the number is prime than i will add the number to a list and than opening the file to write the content of list into the opened file.

6 Exceptions

6.1 Problem :

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. #### CODE and Explanation below

[55]:
```
def check_prime(x):

    try:
        num=int(x)
        # print(is_prime(num))
        if(num<=1):
            raise ValueError
        if(is_prime(num)):
            print("Prime ")
        else :
            print("Not Prime")
    except ValueError:
        print("ERROR! : Enter a positive integer value ")
x = input('Enter a number: ')
check_prime(x)
```

```
Enter a number: 97
Prime
```

6.1.1 Explanation of above code:

I am checking if the input is in correct format or not using try and except and if input is in wrong than raise the error else i am checking if the no given is prime or no using the fucntion that i previously made and if it is prime it will print **prime** othewise it will print **Not Prime**.