EE2703 - Week 1

D.Somesh Reddy <ee21b043@smail.iitm.ac.in>

February 4, 2023

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.*

Changing author name
1. Go to property inspector 2. Go to advanced tools
3. In document meta data section,change the author name and email address which is in **JSON** format

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
```

Here '/' is division operator and it performs **floating point division.** Even if both the numbers are integers it return a floating point number.

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

```
2
```

'//' is a **floor division operator.** It return the largest integer integer which is less than or equal to the answer of the actual division.

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

```
10 10 1.0
```

Here a and b variables are declared as integers and they are equated to 10.

2.2 Strings and related operations

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

```
Hello
```

Here we declared a variable ‘a’.We are printing it.

Correct Code

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

```
Hello 10
```

Here b is integer and a is string,so first we are converting integer to string using str function and then we are concatenating with a.

[6]:
```
# 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

for i in range(40):
    print("-",end="")
print("\n",end="")
print(f"{42:>42}")

# Then print one more line of length 40, but with the pattern '*-*-*-'

print("*-"*20)
```

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

Here line is printed using a for loop,and to avoid printing a new line after every ‘-’ sign **end parameter** is used,it specifies what character should be used at end of output line

To print ‘–*’ i have multiplied the string ” *-” i output the string 20 times to get output of length 40

[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
```

Here “f” in print function is for string formatting.
It is used to embed expressions inside a string using curly braces
{b:>10} means b is right justified to 10 spaces

[8]:
```
# 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

courses = []
course1 = dict(ID = "EE2703",name = "Applied Programming Lab")
course2 = dict(ID = "EE2003",name = "Computer Organization")
course3 = dict(ID = "EE5311",name = "Digital IC design")

courses.append(course1)
courses.append(course2)
courses.append(course3)

for course in courses:
    print("{:<10} {:>40}".format(course["ID"], course["name"]))
```

```
EE2703                       Applied Programming Lab
EE2003                         Computer Organization
EE5311                             Digital IC design
```

Here first I have created a course list,and then created dictionaries with name and ID.
course.append() this means,i am adding each dictionary into course list.
Then printing all the courses in the list using for loop.
“{:<10} {:>40}” this syntax tells that ID is left justified by 10 spaces and course name is right justified with 40 spaces

3 Functions for general manipulation

[12]:
```
# 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.
def twosc(x, N):
    if(x<0):
        x=(2**N)+x
    s=""
    while N>0:
        if x%2==0:
            s="0"+s
        else:
            s="1"+s
        x=x//2
        N=N-1
    return s
n=int(input())
print(twosc(n,16))
```

```
 40
```

```
0000000000101000
```

If the number is negative then the number is converted to 2^n + n or else it is continued.
Running a while loop until N reaches 0.Inside while loop the current binary string is updated after every iteration with a new binary digit addition to old string.

4 List comprehensions and decorators

[13]:
```
# Explain the output you see below
[x*x for x in range(10) if x%2 == 0]
```

[13]:
```
[0, 4, 16, 36, 64]
```

The output is a **list comprehension** that creates a list of the squares of all even numbers between 0 and 9. The list comprehension contains of three parts:
1.The expression x*x is calculated
2.In the for loop x is iterated from 0 to 9
3.The if statement filter only even numbers and first expression is calculated only when if statement is true

[14]:
```
# 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]
```

[14]: `[1, 2, 3, 4, 5, 6, 7, 8, 9]`

The output is a list comprehension.The output contains all the values of a matrix.
The *for row in matrix* for loop iterates over all the sublists in matrix.
The *for v in row* nested for loop iterates over each value in sublist.

[15]:
```
# 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

def is_prime(x):
    if x<2:
        return False
    for i in range(2,(x//2)+1):
        if x%i == 0:
            return False
    return True
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, 53, 59, 61, 67, 71, 73,
79, 83, 89, 97]
```

The given number is checked with all the numbers from 2 to n/2,whether it is divisible or not,using a for loop.
Finally the output is a list comprehension,which is printed as a list.

[17]:
```
# 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
```

The f1 function takes an argument x and returns a string “happy” concatenated with x
The f2 function takes a function f as an argument and returns a new function wrapper.
The f3 variable is assigned the result of calling f2 with f1 as an argument.
f1 returns “happy flappy” and f3 returns “hello happy flappy world”

[18]:
```
# Explain the output below
@f2
def f4(x):
    return "nappy " + x

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

```
Hello nappy flappy world
```

The @f2 syntax is a decorator in Python.
That means here f4=f2(f4).
Initially f4 returns "happy nappy" as argument to f2,which inturn returns "Hello happy nappy world".

5 File IO

[20]:
```
# 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.

def is_prime(x):
    if x<2:
        return False
    for i in range(2,(x//2)+1):
        if x%i == 0:
            return False
    return True

def write_primes(N, filename):
    with open(filename,"w") as file:
        for i in range(1,N+1):
            if is_prime(i):
                file.write(str(i)+" ")
n=int(input())
write_primes(n,"primes")
```

```
20
```

6 Exceptions

[21]:
```
# 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.
def check_prime(x):
    try:
        x=int(x)
        if(is_prime(x)):
            print(f"{x} is a prime number")
        else:
            print(f"{x} is not a prime number")
    except ValueError:
        print(f"{x} is not an integer")
```

```
x=input("Enter a number ")
check_prime(x)
```

```
Enter a number  5
```

```
5 is a prime number
```

The check_prime function tries to convert the input x to an integer using the int(x) function.
If the conversion is successful, the function checks if the integer is a prime number by calling the is_prime function. The result of the is_prime function is passed to an if statement that prints a message indicating whether the number is a prime or not.
If the conversion of x to an integer fails, a ValueError exception is raised, and the code in the except block is executed to print a message indicating that x is not an integer.

7 Process of running code

The note book is written in jupyter * Run the cells in sequence to generate output

Jupyter notebook can be opened in several other platforms like * Microsoft Azure * Google colab * Kaggle

No other packeges are required.