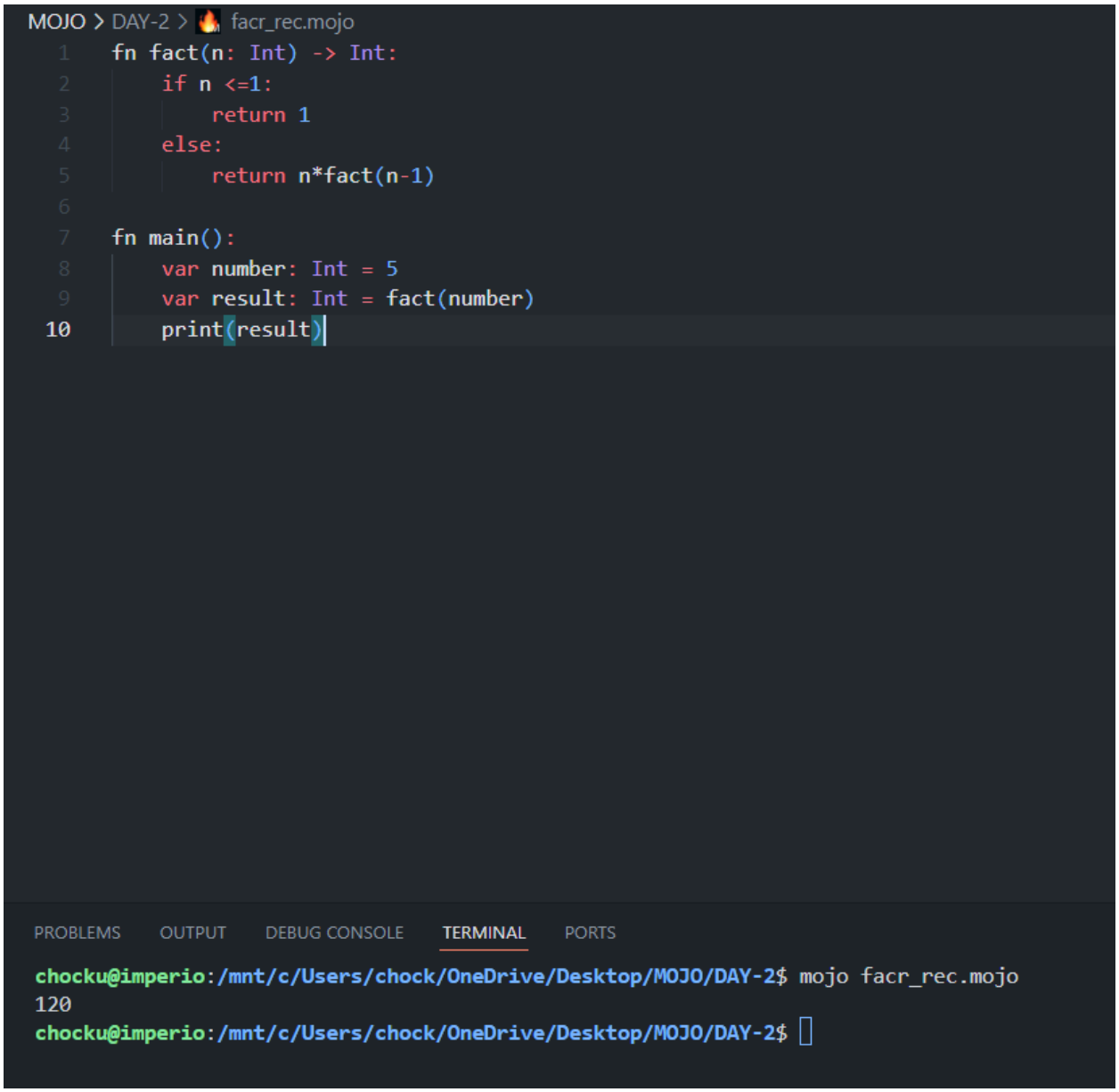
## Day2 programs to execute in Mojo

1. WAP to find the factorial of a given using the recursion function.



2. Define a global variable and show how it can be accessed and modified from different functions. Also, define a local variable within a function and demonstrate how it retains its value between function calls but is not accessible outside the function.

MOJO > DAY-2 > glob.mojo

```
1  var glob = 20
2
3  def func():
4      a = 1
5      print("a inside func")
6
7  def main():
8      print("accesing global x",glob)
9      a = 10
10     func()
11     a+=a
12     func()
13     print("a outside function",a)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$ mojo glob.mojo
accesing global x 20
a inside func
a inside func
a outside function 20
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$
```

3. Write an example program for declaring arguments with a default value and perform some mathematical operations.

math.mojo U

MOJO > DAY-2 > math.mojo

```
fn math(a: Float16=1.0, b:Float16=5.0):
    var sum = a+b
    var diff = a-b
    var mul = a*b
    var div = a/b

    print("sum",sum)
    print("diff",diff)
    print("mul",mul)
    print("div",div)

fn main():
    print("using default values")
    math()
    print("\n using custom")
    math(10.0,5.0)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$ mojo math.mojo
using default values
sum 6.0
diff -4.0
mul 5.0
div 0.199951171875

 using custom
sum 15.0
diff 5.0
```

4. WAP to add two numbers using normal parameter declaration, inout, and borrowed argument convention. Also, print the original values in the main code after calling each function.

MOJO > DAY-2 > ex.mojo

```
fn add(x:Int, y:Int)->Int:
    return(x+y)

fn borrow(borrowed x: Int, borrowed y: Int)->Int:
    return x+y

fn add_inout(inout x:Int, inout y:Int)->Int:
    x=20
    y=10
    return x+y

fn main():
    var x:Int = 2
    var y:Int = 4
    print("after calling func",x,y)
    print(borrow(x,y))
    print("after calling fun",x,y)
    print(add_inout(x,y))
    print("after calling func",x,y)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$ mojo ex.mojo
after calling func 2 4
6
after calling fun 2 4
30
after calling func 20 10
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$
```

5. WAP to demonstrate struct to a print student details with a constructor.

MOJO > DAY-2 > struct.mojo

```
struct Student:
    var name:String
    var age:Int
    var grade:String
    def __init__(inout self, name:String, age:Int,grade:String):
        self.name = name
        self.age = age
        self.grade = grade
    def print_deets(self):
        print("Student deets:")
        print("name",self.name)
        print("age",self.age)
        print("grade",self.grade)

def main():
    var stu1: Student = Student(name="Alice",age=15,grade="A")
    var stu2: Student = Student(name="Alex",age=20,grade="A")

    stu1.print_deets()
    stu2.print_deets()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$ mojo struct.mojo
Student deets:
name Alice
age 15
grade A
Student deets:
name Alex
age 20
grade A
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$
```

6. WAP to demonstrate a constructor for allocating the memory and a special function to deallocate the memory using struct.

MOJO > DAY-2 > const.mojo

```
@value
struct Friend:
    var name:String
    var age: Int

    fn __init__(inout self, name: String, age: Int):
        self.name = name
        self.age = age
    fn __del__(owned self):
        print("deleted self",self.name)

fn friends():
    var fri = Friend("max",20)
    print(fri.name)
    print(fri.age)

def main():
    friends()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$ mojo const.mojo
max
20
deleted self max
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$
```

7. WAP to demonstrate multiple constructors with the help of method overloading.

MOJO > DAY-2 > mulconst.mojo

```
struct Student:
    var name:String
    var sem: Int

    fn __init__(inout self):
        self.name = "Name not found"
        self.sem = 0

    fn __init__(inout self, name:String, sem:Int):
        self = Student()
        self.name = name
        self.sem = sem

def main():

    var stud1 = Student()
    print(stud1.name)
    print(stud1.sem)

    var stud2 = Student("Chocku",6)
    print(stud2.name)
    print(stud2.sem)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$ mojo mulconst.mojo
Name not found
0
Chocku
6
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$
```

8. Write example program to demonstrate __copyinit__() constructor.

MOJO > DAY-2 > copy.mojo

```
struct Book:
    var title: String
    var author: String

    fn __init__(inout self, title: String, author: String):
        self.title = title
        self.author = author
    fn __copyinit__(inout self, existing:Self):
        self.title = existing.title
        self.author = existing.author

def main():
    var originalBook = Book("Harry Potter","JK Rowling")

    var copied = originalBook

    print(copied.title)
    print(originalBook.title)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$ mojo copy.mojo
Harry Potter
Harry Potter
chocku@imperio:/mnt/c/Users/chock/OneDrive/Desktop/MOJO/DAY-2$
```