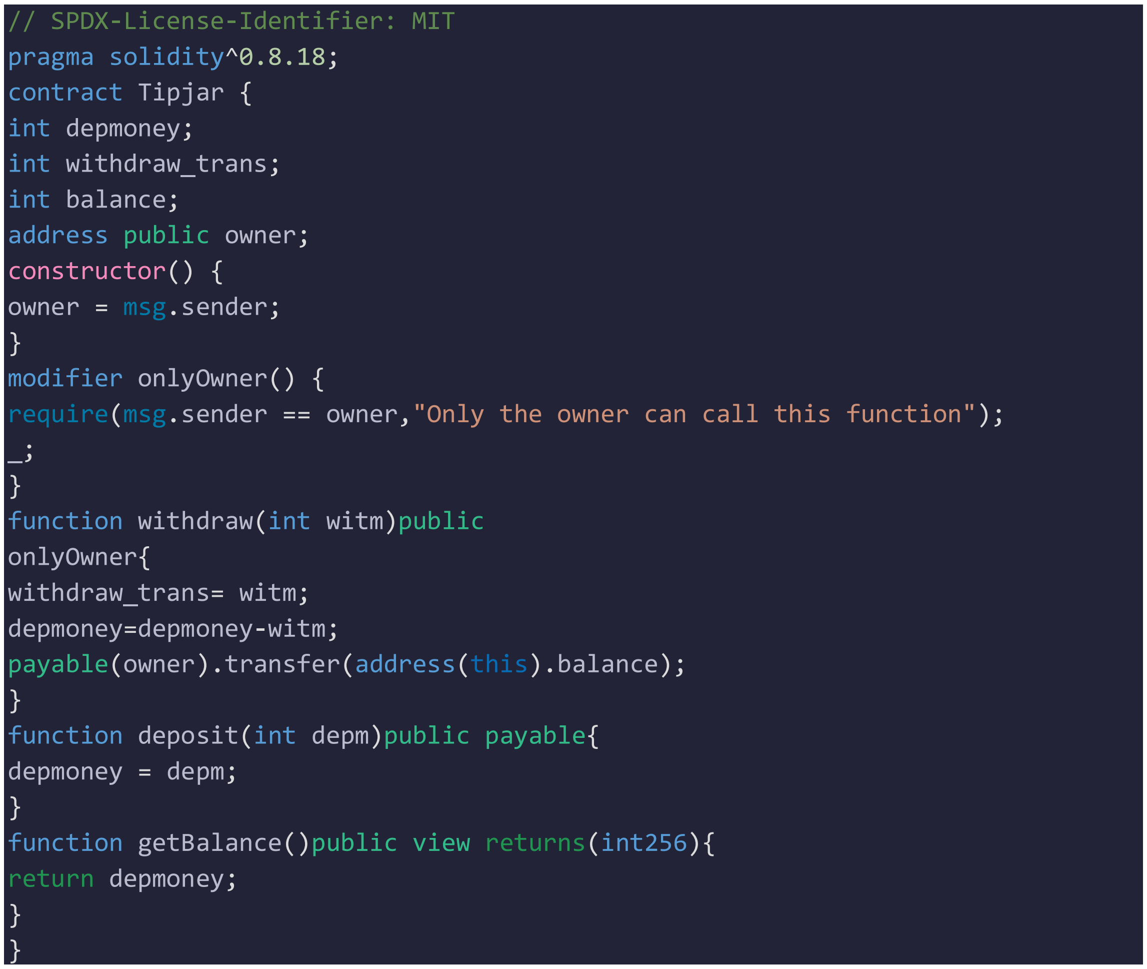
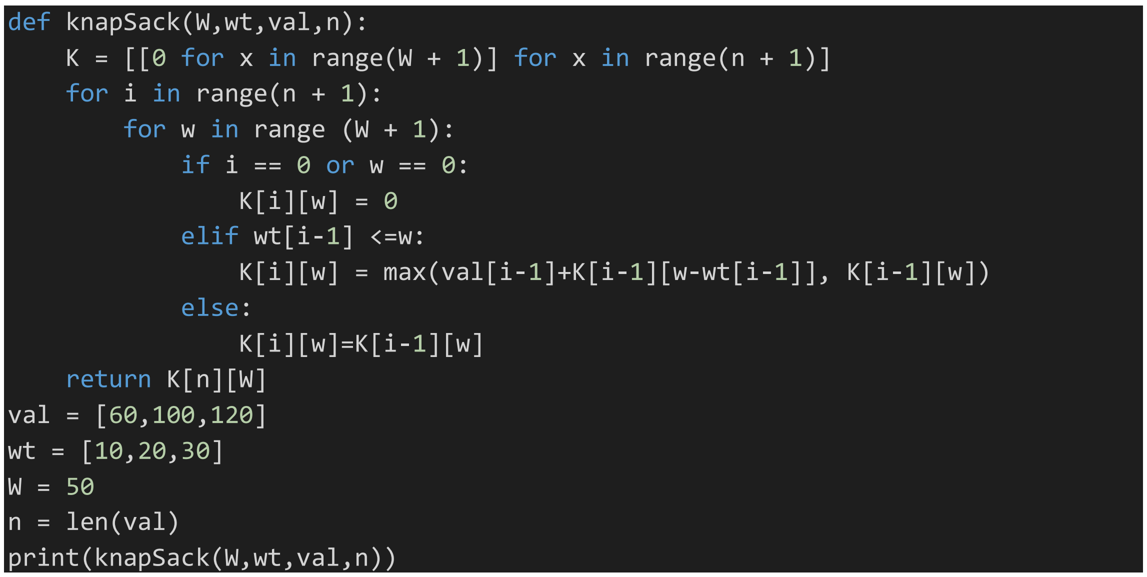
```
def knapSack(W,wt,val,n):
   K = [[0 for x in range(W + 1)] for x in range(n + 1)]
   for i in range(n + 1):
      for w in range (W + 1):
         if i == 0 or w == 0:
            K[i][w] = 0
         elif wt[i-1] <=w:
            K[i][w] = max(val[i-1]+K[i-1][w-wt[i-1]], K[i-1][w])
         else:
            K[i][w]=K[i-1][w]
   return K[n][W]
val = [60,100,120]
wt = [10,20,30]
W = 50
n = len(val)
print(knapSack(W,wt,val,n))
```



```
// SPDX-License-Identifier: MIT
pragma solidity^0.8.18;
contract Tipjar {
int depmoney;
int withdraw_trans;
int balance;
address public owner;
constructor() {
owner = msg.sender;
}
modifier onlyOwner() {
require(msg.sender == owner,"Only the owner can call this function");
_;
}
function withdraw(int witm)public
onlyOwner{
withdraw_trans= witm;
depmoney=depmoney-witm;
payable(owner).transfer(address(this).balance);
}
function deposit(int depm)public payable{
depmoney = depm;
}
function getBalance()public view returns(int256){
return depmoney;
}
}
```



```python
def knapSack(W,wt,val,n):
    K = [[0 for x in range(W + 1)] for x in range(n + 1)]
    for i in range(n + 1):
        for w in range (W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif wt[i-1] <=w:
                K[i][w] = max(val[i-1]+K[i-1][w-wt[i-1]], K[i-1][w])
            else:
                K[i][w]=K[i-1][w]
    return K[n][W]
val = [60,100,120]
wt = [10,20,30]
W = 50
n = len(val)
print(knapSack(W,wt,val,n))
```