In [6]:
```
def is_palindrome(x):
    str_x = str(x)
    print(str_x)
    print(str_x[::-1]) ### only string or list
    return str_x == str_x[::-1]
```

In [7]:
```
is_palindrome(1234)
```

```
1234
4321
```

Out[7]: False

In [8]:
```
is_palindrome(1221)
```

```
1221
1221
```

Out[8]: True

In [9]:
```
is_palindrome('anna')
```

```
anna
anna
```

Out[9]: True

In [12]:
```
def is_palindrome(x):
    print(x)
    print(x[::-1]) ### only string or list
    return x == x[::-1]
```

In [13]:
```
is_palindrome(1221)
```

```
1221
---------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
Cell In[13], line 1
----> 1 is_palindrome(1221)

Cell In[12], line 3, in is_palindrome(x)
      1 def is_palindrome(x):
      2     print(x)
----> 3     print(x[::-1]) ### only string or list
      4     return x == x[::-1]

TypeError: 'int' object is not subscriptable
```

In [21]:
```python
def is_pally(x):
    ## -121 reverse 121-
    if x < 0:
        return False
    if x % 10 == 0:
        return False
    else:
        half = 0
        ## half = 0
        ## x = 12521
        ## half = 1, x = 1252
        ## half = 12, x = 125
        ## half = 125 , x = 12
        while half < x:
            half = (half * 10) +  x % 10
            x = x // 10

        return half == x or half // 10 == x
```

In [22]:
```python
is_pally(12521)
```

Out[22]:
```
True
```

Algorithm for hangman

1. Pick word
2. Draw lines
3. Draw hook
4. Ask user for letter
5. if letter not in word -> add body part to hook and decrease chances and add letter to already guessed
6. if letter in word -> replace '_' with that letter. add letter to already guessed letters
7. repeat 4-6 until all '_' are replaced with letters (you win) or chances <= 0 (you lose)

In [ ]:
```python

```