



# ***DATA STRUCTURE SECOND CLASS LAB 7***

***PREPARED BY:  
Ahmed Eskander Mezher  
AYOOB ABDULMUNEM***



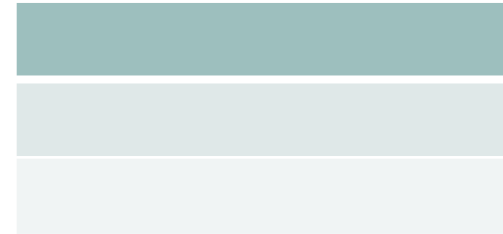
# **Lecture Outline: -**

## **1- Stack**



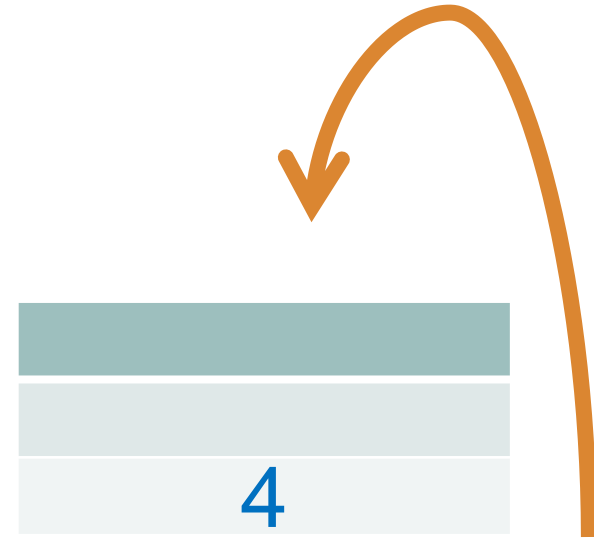
```
def create():  
    return []  
  
def push(stack,item):  
    stack.append(item)  
def pop(stack):  
    if len(stack)==0:return ('stack is empty')  
    else:  
        return stack.pop()  
→ stack = create()  
push(stack,4)  
push(stack,8)  
push(stack,2)  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))
```

Stack



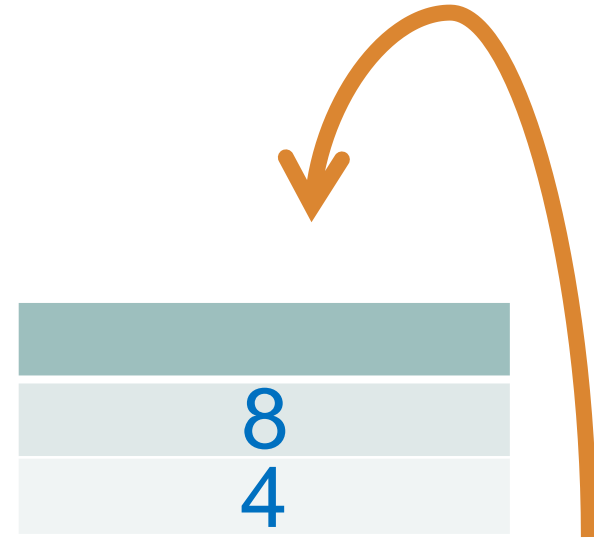
```
def create():  
    return []  
  
def push(stack,item):  
    stack.append(item)  
def pop(stack):  
    if len(stack)==0: return ('stack is empty')  
    else:  
        return stack.pop()  
stack = create()  
→ push(stack,4)  
  push(stack,8)  
  push(stack,2)  
  print(pop(stack))  
  print(pop(stack))  
  print(pop(stack))  
  print(pop(stack))
```

Stack



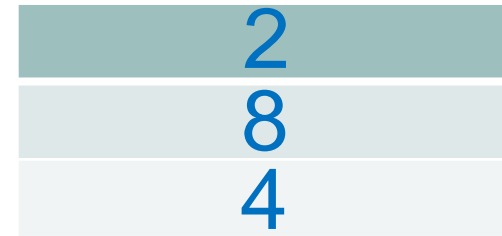
```
def create():  
    return []  
  
def push(stack,item):  
    stack.append(item)  
def pop(stack):  
    if len(stack)==0: return ('stack is empty')  
    else:  
        return stack.pop()  
stack = create()  
push(stack,4)  
→ push(stack,8)  
push(stack,2)  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))
```

Stack



```
def create():  
    return []  
  
def push(stack,item):  
    stack.append(item)  
def pop(stack):  
    if len(stack)==0: return ('stack is empty')  
    else:  
        return stack.pop()  
stack = create()  
push(stack,4)  
push(stack,8)  
→ push(stack,2)  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))
```

Stack



```
def create():  
    return []
```

```
def push(stack,item):  
    stack.append(item)
```

```
def pop(stack):  
    if len(stack)==0: return ('stack is empty')  
    else:  
        return stack.pop()
```

```
stack = create()
```

```
push(stack,4)
```

```
push(stack,8)
```

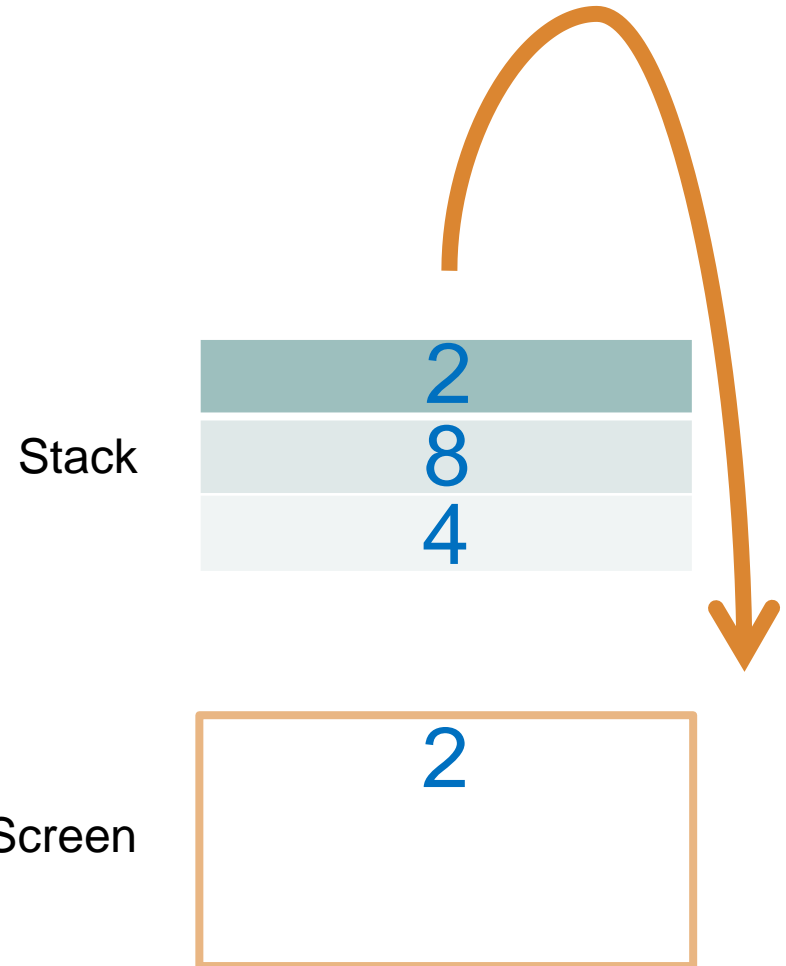
```
push(stack,2)
```

```
print(pop(stack))
```

```
print(pop(stack))
```

```
print(pop(stack))
```

```
print(pop(stack))
```



```
def create():  
    return []
```

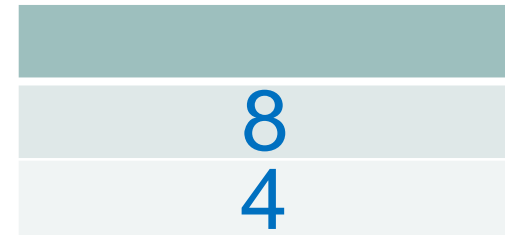
```
def push(stack,item):  
    stack.append(item)
```

```
def pop(stack):  
    if len(stack)==0: return ('stack is empty')  
    else:  
        return stack.pop()
```

```
stack = create()  
push(stack,4)  
push(stack,8)  
push(stack,2)  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))
```



Stack



Screen



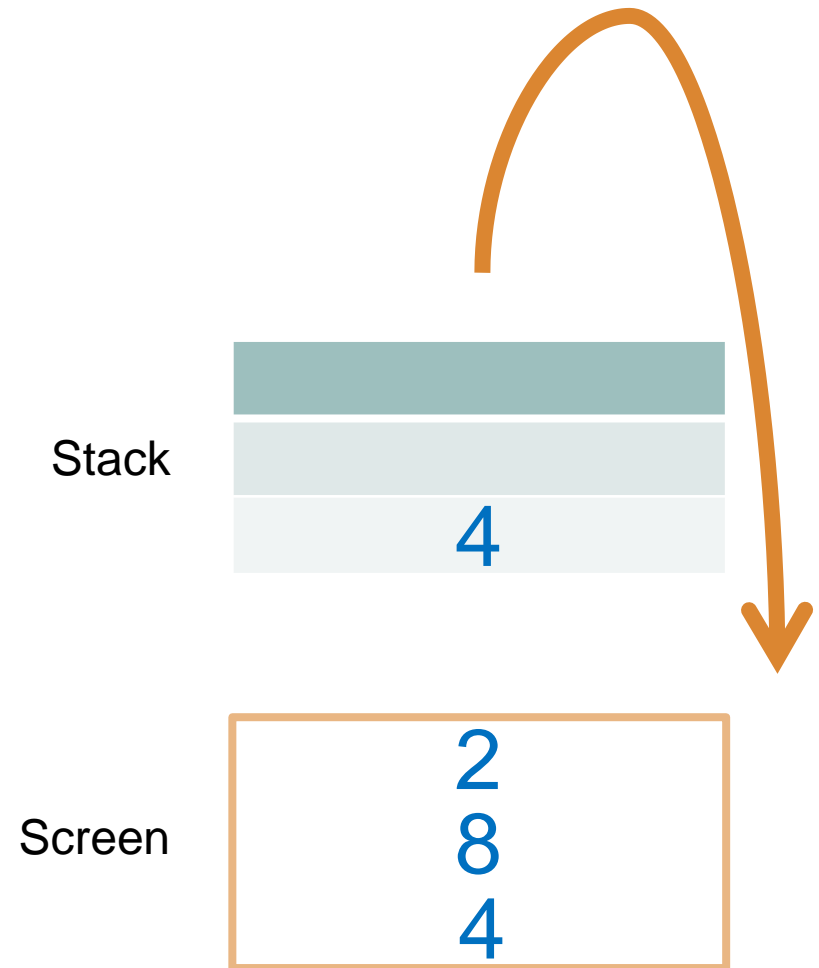


```
def create():  
    return []
```

```
def push(stack,item):  
    stack.append(item)
```

```
def pop(stack):  
    if len(stack)==0: return ('stack is empty')  
    else:  
        return stack.pop()
```

```
stack = create()  
push(stack,4)  
push(stack,8)  
push(stack,2)  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))  
print(pop(stack))
```



```
def create():  
    return []
```

```
def push(stack,item):  
    stack.append(item)
```

```
def pop(stack):
```

```
    if len(stack)==0: return ('stack is empty')
```

```
    else:
```

```
        return stack.pop()
```

```
stack = create()
```

```
push(stack,4)
```

```
push(stack,8)
```

```
push(stack,2)
```

```
print(pop(stack))
```

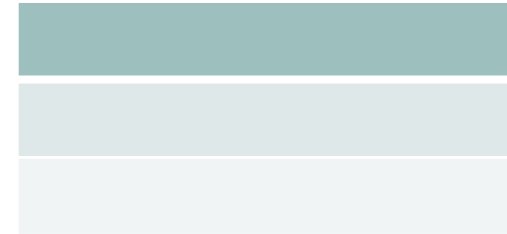
```
print(pop(stack))
```

```
print(pop(stack))
```

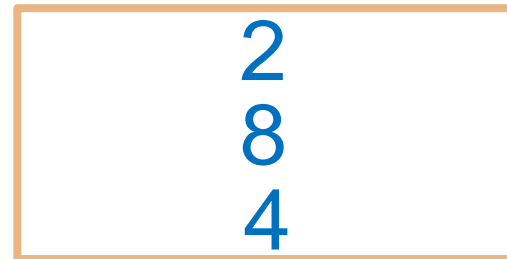
```
print(pop(stack))
```



Stack



Screen

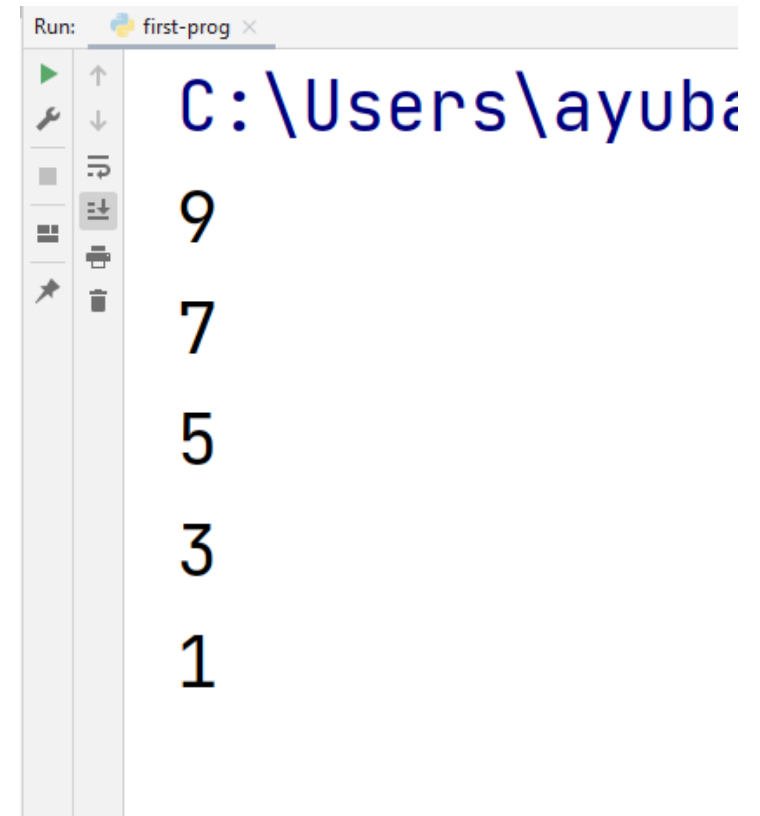


Stack is empty



Write a program to add all odd numbers from 1 to 10 in the stack.

```
def create():  
    return []  
def push(stack,item):  
    stack.append(item)  
def pop(stack):  
    if len(stack)==0: return ('stack is empty')  
    else:  
        return stack.pop()  
def main():  
    stack = create()  
    for i in range(1,10,2):  
        push(stack,i)  
    for i in range(len(stack)):  
        print(pop(stack))  
main()
```



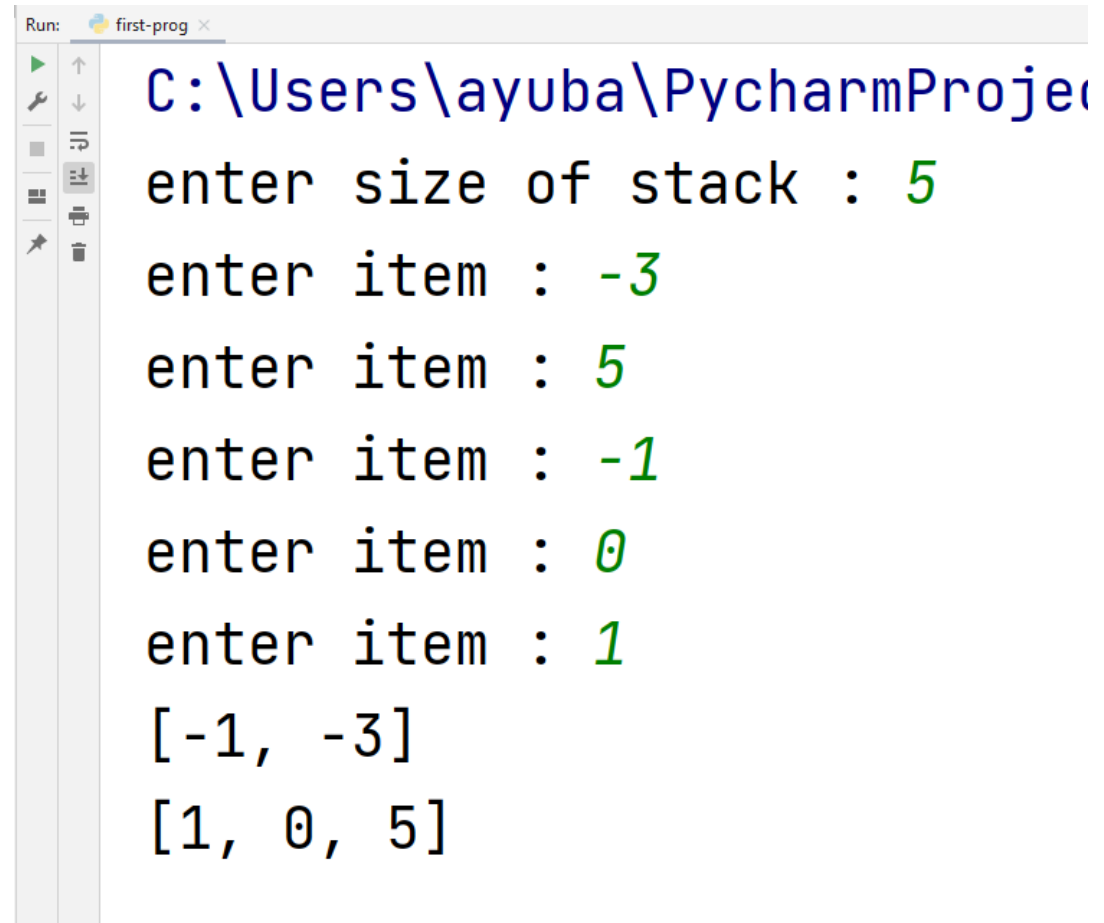
Run: first-prog x

```
C:\Users\ayuba  
9  
7  
5  
3  
1
```



Write a program to add numbers to a stack by the user,  
then split the stack into two stacks, negative and positive stack.

```
def create():  
    return []  
  
def push(stack,item):  
    stack.append(item)  
def pop(stack):  
    if len(stack)==0: return ('stack is empty')  
    else:  
        return stack.pop()  
def main():  
    stack = create()  
    size=int(input('enter size of stack : '))  
    for i in range(size):  
        x=eval(input('enter item : '))  
        push(stack,x)  
    stackNeg=create()  
    stackPos=create()  
    for i in range(len(stack)):  
        w=pop(stack)  
        if w<0:  
            push(stackNeg,w)  
        else:  
            push(stackPos,w)  
    print(stackNeg)  
    print(stackPos)  
main()
```



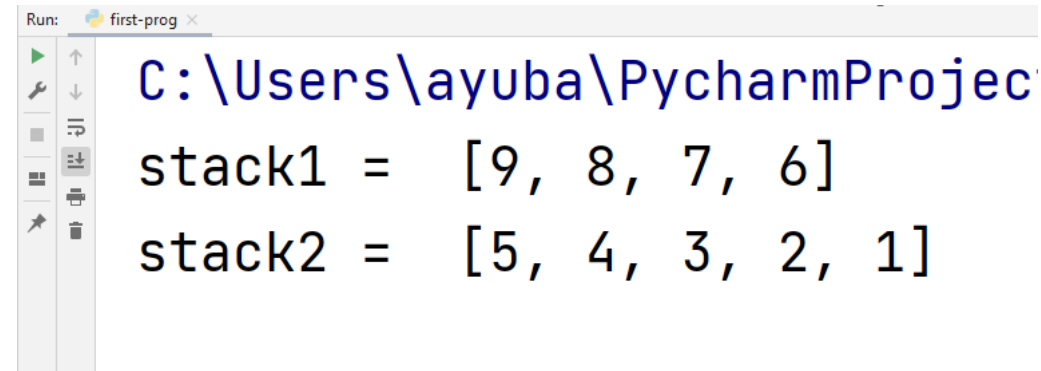
The screenshot shows a PyCharm Run window titled 'first-prog'. The output of the program is displayed in a monospaced font. It shows the user entering the size of the stack as 5, then entering five items: -3, 5, -1, 0, and 1. The program then splits these into two lists: [-1, -3] for negative numbers and [1, 0, 5] for positive numbers.

```
Run: first-prog x  
C:\Users\ayuba\PycharmProject  
enter size of stack : 5  
enter item : -3  
enter item : 5  
enter item : -1  
enter item : 0  
enter item : 1  
[-1, -3]  
[1, 0, 5]
```



Write a program to add numbers from 1 to 9, then split the stack into two stacks.

```
def create(): return []
def push(stack,item):
    stack.append(item)
def pop(stack):
    if len(stack)==0: return ('stack is empty')
    else:
        return stack.pop()
def main():
    stack = create()
    for i in range(1,10,1):
        push(stack,i)
    stack1=create()
    stack2=create()
    for i in range(len(stack)//2):
        push(stack1, pop(stack))
    for i in range(len(stack)):
        push(stack2, pop(stack))
    print('stack1 = ',stack1)
    print('stack2 = ',stack2)
main()
```



Run: first-prog x

```
C:\Users\ayuba\PycharmProjec
stack1 = [9, 8, 7, 6]
stack2 = [5, 4, 3, 2, 1]
```



## Worksheet:

- 1- Write a program to concatenate two stacks.
- 2- Write a program to create a stack with the following numbers [-3, -1, 1, 3, 5] by using Loop then split it into two stacks, negative and positive stack.



**Thank You**

