2nd Midterm Exam

2023.05.11 (Thursday) 10:10 – 12:00

1. **[Python Basics, Conditionals & Loops]** In each of the following questions, you are asked to show what will be printed out. If there is an error, please explain why it is an error. (21%)

(a)

```
s = "1.23"
print(float(int(s)))
```

(b)

```
_a, _b, _c = 2, 3, 5
x, y, z = _a*_b, _b*_c, _c*_a
print(x, y, z)
```

(c)

```
x, y = 5, 0
while x < y or y < x:
    x, y = x - 1, y + 1
print(x, y)
```

(d)

```
a, b = 72, 120
while b != 0:
    t = b
    b = a % b
    a = t
print(a)
```

(e)

```
score = 20
while score > 1:
    score = score/2 - 1
print(score)
```

(f)

```
s = "yadyam evol i"
print(s[::-1])
```

(g)

```
n = 7
a, b, i = 1, 1, 1
while i < n:
    a, b = b, a + b
    i += 1
print(a)
```

2. **[Lists]** In each of the following questions, you are asked to show what will be printed out. If there is an error, please explain why it is an error. (36%)

(a)

```
friends = ['Sunny', 'Jenny', 'Pinky']
for f in friends:
    print("Happy New Year:", f)
```

(b)

```
a = [1, 2, 3]
b = [a]*3
b[0][0] = 0
print(a)
```

(c)

```
smallest_so_far = -3
for v in [6, 2, -2, 1, 5, -1]:
    if v < smallest_so_far:
        smallest_so_far = v
print(smallest_so_far)
```

(d)

```
numbers = [-1, 0, 5, -3, 9, -11, 4]
positives = []
for num in numbers:
    if num > 0:
        positives.append(num)
print(positives)
```

(e)

```
values = [2, 0, 2, 3]
for v in values:
    v = 2 * v
print(values)
```

(f)

```
a = [1, 2, 3, 4, 5]
for i in range(len(a)-1, -1, -1):
    print(a[i], end=" ")
```

(g)

```
days = [[['Tue','May',9], ['Thu','May',11]], ['Fri','Jun',30]]
print(days[1][0], days[1:], days[0][0][1][1])
```

(h)

```
l = [1, 2, 3]
l.extend([4])
l.append(5)
l.append([6])
print(l)
```

(i)

```
l = ['n', 'b', 'a']
for i in range(len(l)):
    for j in range(i+1, len(l)+1):
        print(l[i:j])
```

(j)

```
nums = [1, 2, 3, 4, 5, 6, 7]
k = 3
n = len(nums)
k %= n
nums[:] = nums[-k:] + nums[:-k]
print(print(nums))
```

(k)

```
a, b = 18, 48
small = a if a > b else b
cf, pf = [], 1
while pf <= small:
    if a % pf == 0 and b % pf == 0:
        cf.append(pf)
    pf += 1
print(cf)
```

(l)

```
nums = [1, 3, 5, 7, 4, 6, 8, 10, 9, 11]
for i in range(len(nums)-1):
    if nums[i] % 2 != 0:
        continue
    if nums[i+1] % 2 == 0:
        print("Adjacent evens: %d, %d" % (nums[i], nums[i+1]))
        break
```

3. **[Dictionary]** In each of the following questions, you are asked to show what will be printed out. If there is an error, please explain why it is an error. (12%)

(a)

```
D = {4:3, 3:2, 2:1}
for x in D.values():
    print(D[x])
```

(b)

```
foo = {1:'1', 2:'2', 3:'3'}
del foo[1]
foo[1] = '10'
del foo[2]
foo[5] = 'bar'
print(foo)
```

(c)

```
result = {}
for i in range(5):
    for j in range(10):
        result[i] = j
print(result)
```

(d)

```
d = {2: ['free', 'and', 'easy'],
     0: ['laid', 'back'],
     1: ['happy', 'go', 'lucky']}
print(d[1][-1][1:])
```

(e)

```
x = []
x.append({2: "two"})
x.append({3: "three"})
print(x[0][2])
```

(f)

```python
d = {"a": 1, "b": 2, "c": 3}
reversed_dict = {}
for key, value in d.items():
    reversed_dict[value] = key
print(reversed_dict)
```

4. **[Function Basics]** In each of the following questions, you are asked to show what will be printed out. If there is an error, please explain why it is an error. (20%)

(a)

```python
x = "2023"
print("sleepy")
def play_music():
    print("la la la ~~~")
print("happy")
```

(b)

```python
h, w = 1.7, 70
def computeBMI(h, w):
    bmi = w / (h**2)
    return bmi
print("his BMI: %.2f" % (computeBMI(h,w)))
```

(c)

```python
def myfunc(x):
    x = 17
    return x + 3
print(x)
```

(d)

```python
def change(s):
    s = s + "!"
word = "holiday"
change(word)
print(word)
```

(e)

```python
def mdicts(dict1, dict2):
    result = dict1.copy()
    for key, value in dict2.items():
        if key in result:
            result[key] += value
        else:
            result[key] = value
    return result
dict1 = {"a": 1, "b": 2, "c": 3}
dict2 = {"b": 3, "c": 4, "d": 5}
print(mdicts(dict1, dict2))
```

(f)

```
def alter(d):
    d["Dec"] = "school"
months = {"Jan": "vacation!"}
alter(months)
print(months)
```

(g)

```
def sumup(n):
    tot = n*(n+1) // 2
    return tot
def num2sum(x):
    mylist = []
    for i in range(x):
        mylist.append(sumup(i))
    return mylist
print(num2sum(5))
```

(h)

```
def equal(d1, d2):
    for k in d1:
        if (not (k in d2)) or (d1[k] != d2[k]):
            return False
    return True
print(equal({1:'1', 2:'2'}, {1:'1', 2:'2', 3:'3'}))
```

(i)

```
def sum_lists(d):
    dsl = {}
    for key in d:
        sum = 0
        for val in d[key]:
            sum = sum + val
        dsl[key] = sum
    return dsl
d = {0:[1,2,3], 3:[2,3,4,0], -1:[-1,3,4]}
print(sum_lists(d))
```

(j)

```
def dp_fib(n):
    pf, cf = 0, 1
    if n <= 1:
        return n
    for i in range(n-1):
        nf = pf + cf
        pf = cf
        cf = nf
    return cf
print(dp_fib(8))
```

5. Given the following defined function `modify_collection(col)`, now suppose you aim to call `modify_collection` with each of the following five collections. Write down what will be printed after `modify_collection` is called. If the function throws an error for some reason, please explain the reason. (20%)

```
def modify_collection(col):
    new_col = {}
    for i in range(1, len(col)):
        new_col[i-1] = col[i-1] + col[i//2]
    print(new_col)
```

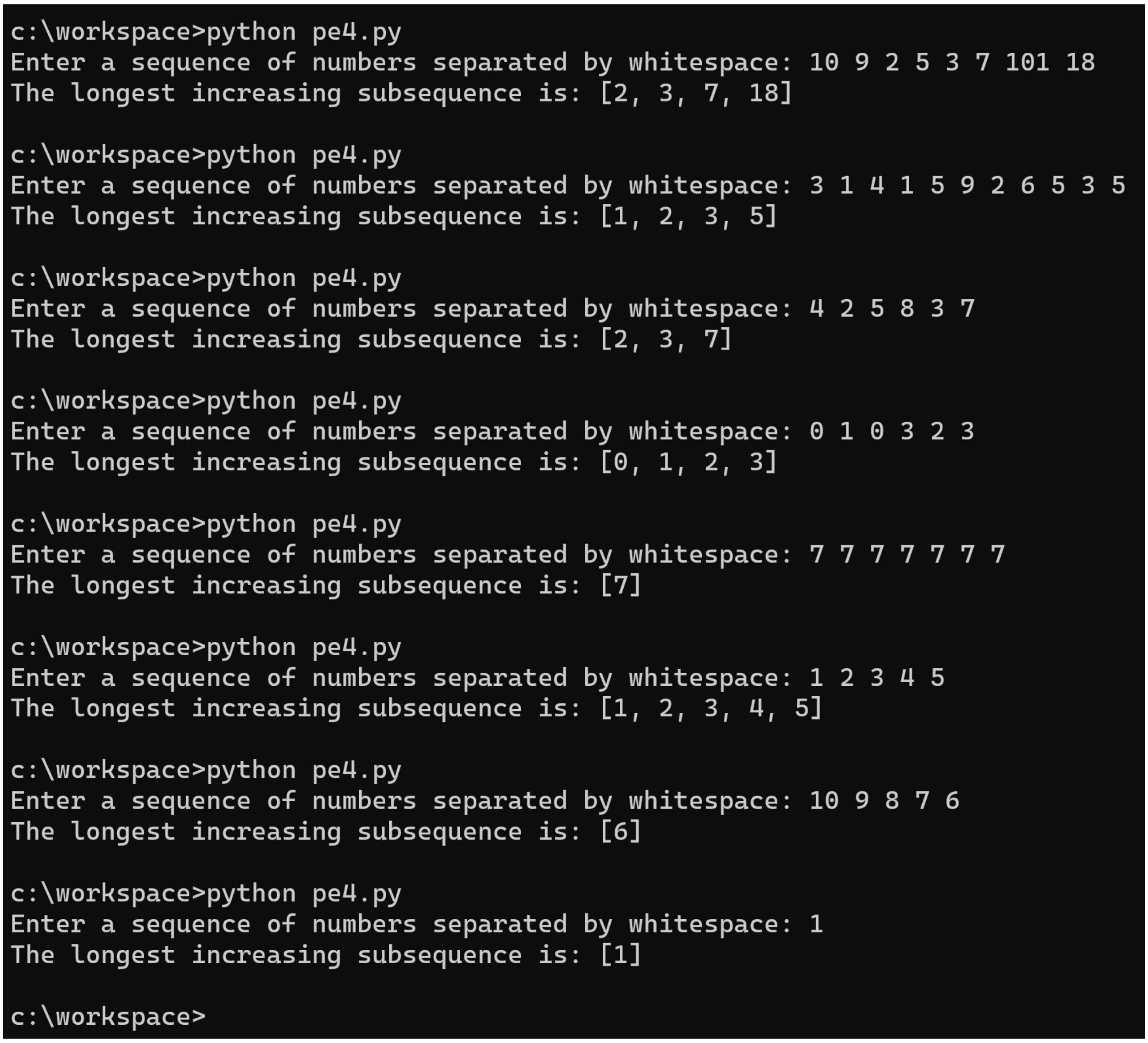
Call `modify_collection` function	**Printed?**
`modify_collection([1, 2, 3, 4])`	**(1)**
`modify_collection(["a", "b", "c", "d"])`	**(2)**
`modify_collection("abcd")`	**(3)**
`modify_collection({-1:"z", 0:"a", 1:"b", 2:"c"})`	**(4)**
`modify_collection({-1:[-1,0], 0:[0,1], 1:[1,2], 2:[2,3]})`	**(5)**

6. The function `add_matrix(mat1, mat2)` is created to perform summation of two matrices, `mat1` and `mat2`, which are in the format of 2-dimensional lists. We have provided you the sample input `mat1` and `mat2`, as below, for the function call of `add_matrix`. The printed result is also displayed. Complete the missing part of `add_matrix`. (11%)

```
def add_matrix(mat1, mat2):
    result = []
    for i in range(len(mat1)):
        row = []
        for j in range(len(mat1[0])):
            row.append(______________________)  # Fill in the blank
        result.append(row)
    return result

mat1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
mat2 = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]
print(add_matrix(mat1, mat2))
# printed: [[10, 10, 10], [10, 10, 10], [10, 10, 10]]
```

7. **Longest Increasing Subsequence.** Your task is to find the longest increasing subsequence of a given sequence of numbers. Given an input sequence, your program should find a subsequence of that sequence that is strictly increasing, meaning that each number in the subsequence is greater than the previous number. The subsequence should be as long as possible, meaning that there is no longer increasing subsequence possible. The program should take user input of a sequence of numbers, and output the longest increasing subsequence. If there are multiple same-length longest increase subsequences, you need to output any one of them. (20%)

Sample Input/output:

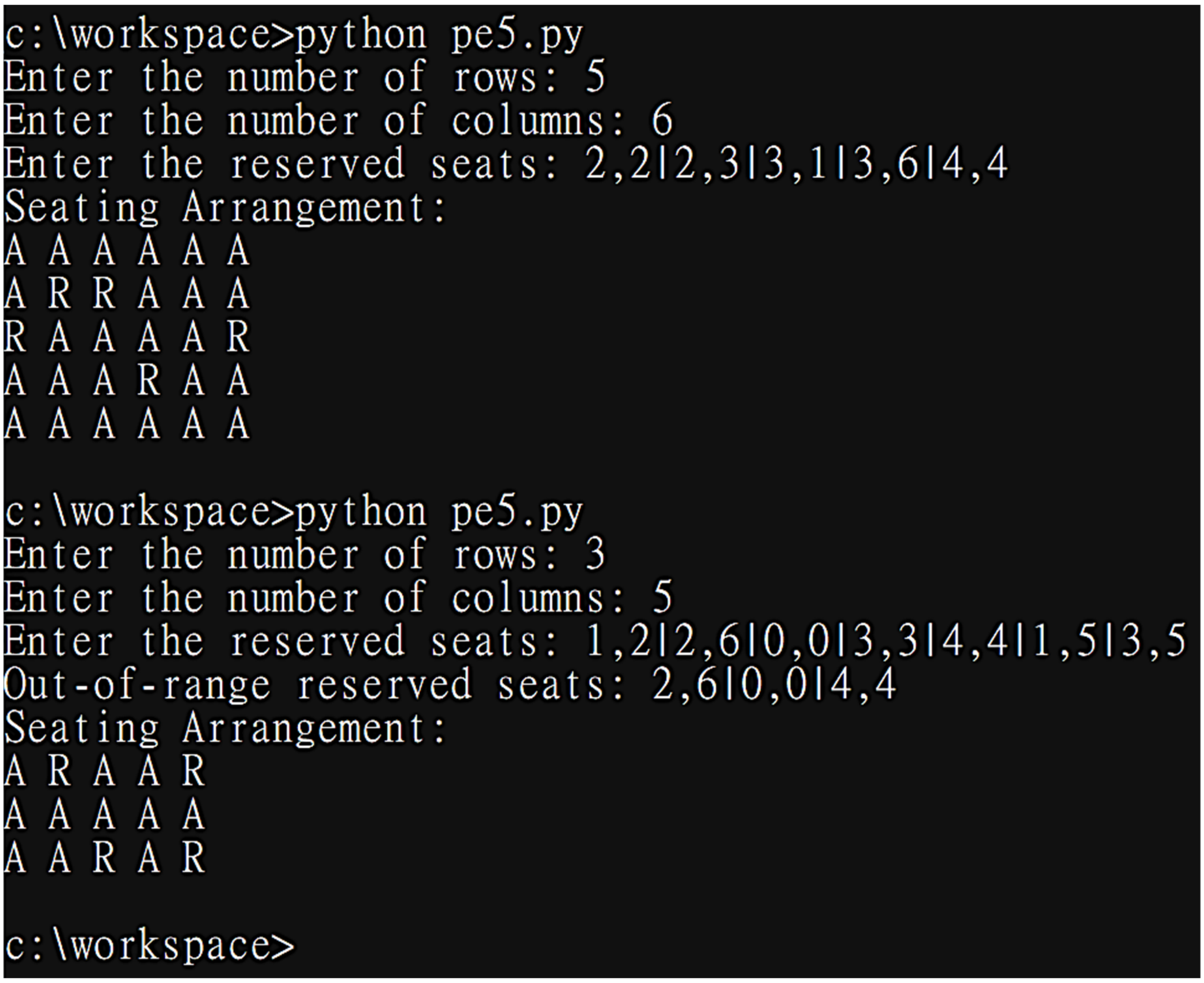


```
c:\workspace>python pe4.py
Enter a sequence of numbers separated by whitespace: 10 9 2 5 3 7 101 18
The longest increasing subsequence is: [2, 3, 7, 18]

c:\workspace>python pe4.py
Enter a sequence of numbers separated by whitespace: 3 1 4 1 5 9 2 6 5 3 5
The longest increasing subsequence is: [1, 2, 3, 5]

c:\workspace>python pe4.py
Enter a sequence of numbers separated by whitespace: 4 2 5 8 3 7
The longest increasing subsequence is: [2, 3, 7]

c:\workspace>python pe4.py
Enter a sequence of numbers separated by whitespace: 0 1 0 3 2 3
The longest increasing subsequence is: [0, 1, 2, 3]

c:\workspace>python pe4.py
Enter a sequence of numbers separated by whitespace: 7 7 7 7 7 7 7
The longest increasing subsequence is: [7]

c:\workspace>python pe4.py
Enter a sequence of numbers separated by whitespace: 1 2 3 4 5
The longest increasing subsequence is: [1, 2, 3, 4, 5]

c:\workspace>python pe4.py
Enter a sequence of numbers separated by whitespace: 10 9 8 7 6
The longest increasing subsequence is: [6]

c:\workspace>python pe4.py
Enter a sequence of numbers separated by whitespace: 1
The longest increasing subsequence is: [1]

c:\workspace>
```

8. **Seating Arrangement in a Theater.** Your task is to simulate a simple seating arrangement in a theater. The theater will have multiple rows and columns, and each seat will be either available (represented by 'A') or reserved (represented by 'R'). Write a Python program that takes the dimensions of the theater and a list of reserved seats, and then displays the seating arrangement as strings, with each row separated by a newline and seats separated by a whitespace. (20%)

- Take the dimensions of the theater as input (number of rows and columns).
- Take the list of reserved seats as input with format **'row,col|row,col|...'**. [hint: use **split**]
- Validate the reserved seats to ensure they are within the theater's dimensions.
- Create seating arrangement with available & reserved seats. [hint: store seats in **nested list**]
- Display the seating arrangement as strings. [hint: use **join** function]

Sample Input/output:



```
c:\workspace>python pe5.py
Enter the number of rows: 5
Enter the number of columns: 6
Enter the reserved seats: 2,2|2,3|3,1|3,6|4,4
Seating Arrangement:
A A A A A A
A R R A A A
R A A A A R
A A A R A A
A A A A A A

c:\workspace>python pe5.py
Enter the number of rows: 3
Enter the number of columns: 5
Enter the reserved seats: 1,2|2,6|0,0|3,3|4,4|1,5|3,5
Out-of-range reserved seats: 2,6|0,0|4,4
Seating Arrangement:
A R A A R
A A A A A
A A R A R

c:\workspace>
```