Dennis Gega | Section 024L

Exam 2 Review Session

My github profile (slides are posted here)

Lists

Features:

- Initialized using square brackets
- Items can be modified after creation
- Lists can store different data types together
- For a list of size n, indexes start at zero and end at n - 1
- Lists can grow and shrink in size as needed

Operations you should know (assume list is named arr):

- Length or array: len(arr)
- Access the ith element of list: arr[i]
- Modify ith element of list: arr[i] = …
- Access last element of list: arr[-1]
- Remove last element of list: arr.pop()
- Insert to end of list: arr.append(…)
- Add list to end of another list: arr.extend(arr2)
- Check for membership: if “Dennis” in arr: …

```
>>> my_list = [1, 2, 3, 4]
>>> print(my_list)
[1, 2, 3, 4]
>>> my_list[0] = "Dennis"
>>> print(my_list)
['Dennis', 2, 3, 4]
>>> my_list.append(True)
>>> print(my_list)
['Dennis', 2, 3, 4, True]
>>> my_list.pop()
True
>>> print(my_list)
['Dennis', 2, 3, 4]
>>> print(my_list[1])
2
>>>
```

Tuples

Features:

- Initialized using parethenses
- Items **CANNOT** be modified after creation
- Lists can store different data types together
- For a list of size n, indexes start at zero and end at n - 1

Operations you should know (assume tuple is named tup):

- Access the ith element of tuple: tup[i]
- Access last element of tuple: tup[-1]
- Check for membership: if “Dennis” in tup: ...

```
>>> my_tuple = (1, )
>>> print(my_tuple)
(1,)
>>> print(my_tuple + (10, 12))
(1, 10, 12)
>>> print(1 in my_tuple)
True
>>> print(len(my_tuple))
1
>>>
```

For Loops

Use cases:

- Used to repeat code for a set amount of times
- Can be used to go through every item (iterate) in a list

Syntax:

- Start: first value of i (can use another name instead of i)
- Stop: The last value i takes on (loop stops right away)
- Step: How much i is incremented each time the loop restarts
- If only one parameter in range, start is 0, stop is the parameter and step is 1
-

```
>>> for i in range(start, stop, step):
```

```
>>> my_list = [1, 2, 3, 4, 5]
>>> for i in range(len(my_list)):
...     print(my_list[i])
...
1
2
3
4
5
>>> for num in my_list:
...     print(num)
...
1
2
3
4
5
```

While Loops

Use cases:

- Used to repeat code indefinitely

Syntax:

- condition: must be either true or false, loop will repeat until condition becomes false
- Can exit right away using break (this goes for any loop)
- Can skip current iteration using continue (also goes for any loop)

```
>>> while <condition>:
```

```
secret_number = 11

while True:
    user_guess = int(input())

    if secret_number == user_guess:
        break

print("Correct!")
```

Question 1

What will be the output of the following code?

```
for i in range(5):
    for j in range(i + 1):
        print(j)
```

a:	b:	c:	d:
1	0	error	12345
12	01		1234
123	012		123
1234	0123		12
12345	01234		1

Question 2

What will be the output of the following code?

```
count = 1
limit = 10
while count < limit:
    limit -= 3
print(limit)
```

a: 10

b: infinite loop

c: 1

d: -2

Question 3

What will be the output of the following code?

```
city = "Montreal"
team = "Canadiens"
result = city[:3] + team[4:]
print(result)
```

a: error

b: MontrealCanadiens

c: Mondiens

d: Montiens

Question 4

What will be the output of the following code?

```
values = [3, 6, 9, 12, 15, 18]
for i in range(1, len(values), 2):
    print(values[i], end=' ')
```

a:
6 12 18

b:
3 9 12 18

c:
6
12
18

d:
infinite loop

Question 5

What will be the output of the following code?

```
nums = [4, 5, 6, 7]
i = len(nums)
while i >= 0:
    print(nums[i], end=' ')
    i -= 1
```

a:
4 5 6 7

b:
7 6 5 4

c:
7
6
5
4

d:
error

Question 6

What will be the output of the following code?

```
letters = ['X', 'Y']
nums = [1, 2, 3]
for l in letters:
    for n in nums:
        print(f"{l}{n}", end=' ')
```

a:
X1 Y1
X2 Y2
X3 Y3

b:
X1 X2 X3 Y1 Y2 Y3

c:
error

d:
X1 Y1 X2 Y2 X3 Y3

Question 7

What will be the output of the following code?

```
def adjust(lst):
    for i in range(len(lst)):
        if lst[i] % 2 == 1:
            lst[i] += 5
    return lst

numbers = [1, 2, 3, 4]
print(adjust(numbers))
```

a:
[6, 2, 8, 4]

b:
[1, 2, 3, 4]

c:
[1, 7, 3, 9]

d:
error

Question 8

What will be the output of the following code?

```
def adjust(lst):
    for i in range(len(lst)):
        if lst[i] % 2 == 1:
            lst[i] += 5
    return lst

numbers = [1, 2, 3, 4]
print(adjust(numbers))
```

a:
[5, 2, 8, 4]

b:
[1, 2, 3, 4]

c:
[1, 7, 3, 9]

d:
error

Classes

Features:

- Brings variable (attributes) and functions (methods) together in a single entity

Special properties:

- def __init__(self, ...): special function that is called every time you create a new object
- self (parameter): refers to the specific instance of the class and allows it to modify its attributes
- Can create functions as normal (must have self if you want to access instance attributes)

```python
class Car:
    # Class attribute
    wheels = 4

    def __init__(self, make, model, year):
        # Instance attributes
        self.make = make
        self.model = model
        self.year = year
        self.speed = 0

    def accelerate(self, increment):
        self.speed += increment
        print(f"The {self.make} {self.model} is now going {self.speed} mph.")

    def brake(self):
        self.speed = 0
        print(f"The {self.make} {self.model} has stopped.")

# Create instances of the Car class
car1 = Car("Toyota", "Camry", 2023)
car2 = Car("Honda", "Civic", 2024)

# Access attributes
print(f"{car1.make} {car1.model} has {car1.wheels} wheels.")

# Call methods
car1.accelerate(50)
car2.accelerate(60)
car1.brake()
```

Question 10

What will be the output of the following code?

```
class Animal:
    def __init__(self, name):
        self.name = name

dog = Animal("Rex")
cat = Animal("Milo")
print(dog.name, cat.name)
```

a:
Milo Milo

b:
Rex Rex

c:
Error

d:
Rex Milo

Question 11

What will be the output of the following code?

```
class Counter:
    count = 0
    def __init__(self):
        Counter.count += 1

a = Counter()
b = Counter()
print(a.count, b.count)
```

a:
1 2

b:
2 2

c:
1 1

d:
error

Question 12

What will be the output of the following code?

```python
class Car:
    def __init__(self, brand, year):
        self.brand = brand
        self.year = year

c1 = Car("Toyota", 2010)
c2 = Car("Honda", 2020)
c1.year = 2025
print(c1.year, c2.year)
```

a:
error

b:
2025 2025

c:
2010 2020

d:
2025 2020