Example: How to Store Student Info

- so far, can store using separate lists for every info

```
names = ['Ana', 'John', 'Denise', 'Katy']
grade = ['B', 'A+', 'A', 'A']
course = [2.00, 6.0001, 20.002, 9.01]
```

- a **separate list** for each item
- each list must have the **same length**
- info stored across lists at **same index**

Python Tuples

- an **ordered sequence** of elements, can be mix element types
- cannot change element values, **immutable**
- create a tuple with **parentheses**

```
thistuple = ()  # a empty tuple

thistuple = ("apple","banana","cherry")
```

- access tuple items by referring to the index number

```
thistuple=(2,"mit",3)
thistuple[0]
```

→ evaluates to 2

Python Tuples

- negative indexing means **beginning from the end**, -1 refers to the last item

```
thistuple = ("apple", "banana", "cherry")

print(thistuple[-1])
```

- slice a range of indexes by specifying *start* and *end*

```
thistuple = (2,"mit",3)
thistuple[1:2]
thistuple[1:3]
```

→ slice tuple, evaluates to `("mit")`

→ slice tuple, evaluates to `("mit",3)`

Python Tuples

- conveniently used to **swap** variable values

```
x = y
y = x
```

```
temp = x
y = temp
```

```
x = y
```

```
(x, y) = (y, x)
```

- used to **return more than one value** from a function

```
def quotient_and_remainder(x, y):
    q = x // y
    r = x % y
    return (q, r)
```

Packing/unpacking

- Tuple packing is used to “pack” a collection of items into a tuple. We can unpack a tuple using Python’s multiple assignment feature.

```
s =("Susan", 19, "CS") # tuple packing
name, age, major = s # tuple unpacking
print(name)
print(age)
print(major)
```

Susan
19
CS

When to use Tuples

- When storing elements that **will not need to be changed**.
- When you want to store your data **in logical immutable pairs**, triples, etc.