Python Basics: Data Structures & Control Flow

Prof. Dnyanesh Khedekar, Big Data Analytics, GIM

1. Lists

Example 1: Creating and accessing a list

```
fruits = ['apple', 'banana', 'cherry']
print(fruits[0])  # apple
```

Lists are ordered collections. You can access items by index. Here, index 0 gives 'apple'.

Example 2: Adding items to a list

```
numbers = [1, 2, 3]
numbers.append(4)
print(numbers)  # [1, 2, 3, 4]
```

The append() method adds an element to the end of the list.

Example 3: Removing items from a list

```
tasks = ['email', 'meeting', 'coding']
tasks.remove('meeting')
print(tasks)
```

The remove() method deletes the first occurrence of a value.

Example 4: Looping through a list

```
for item in ['pen', 'book', 'ink']:
    print(item)
```

A for loop lets you process each element in the list one by one.

Example 5 (Business): Processing sales order IDs

```
orders = [101, 102, 103]
for order in orders:
    print('Processing order:', order)
```

Lists are often used to store multiple business records like sales order IDs.

2. Tuples

Example 1: Creating a tuple

```
coordinates = (10, 20)
print(coordinates[0])
```

Tuples are ordered but immutable collections. Here we store x and y coordinates.

Example 2: Tuple unpacking

```
person = ('Alice', 30)
name, age = person
print(name, age)
```

You can unpack tuple elements directly into variables.

Example 3: Single-element tuple

```
single = (5,)
print(type(single))
```

Always use a comma for single-element tuples, otherwise Python treats it as a normal value.

Example 4: Nested tuples

```
nested = (1, (2, 3))
print(nested[1][0])
```

Tuples can contain other tuples. Here we access the inner value 2.

Example 5 (Business): Employee record as a tuple

```
employee = ('E101', 'John', 'Manager')
print(employee)
```

Tuples are great for storing fixed data like employee records that should not change.

3. Sets

Example 1: Creating a set

```
unique_ids = {1, 2, 3, 3}
print(unique_ids)
```

Sets store only unique values. Duplicate '3' is removed automatically.

Example 2: Adding to a set

```
colors = {'red', 'blue'}
colors.add('green')
print(colors)
```

The add() method inserts new elements into the set.

Example 3: Union of two sets

```
a = {1, 2}
b = {2, 3}
print(a | b)
```

Union combines elements from both sets without duplicates.

Example 4: Intersection of sets

```
a = {1, 2, 3}
b = {2, 3, 4}
print(a & b)
```

Intersection returns common elements from both sets.

Example 5 (Business): Unique customers across two days

```
day1 = {'cust1', 'cust2'}
day2 = {'cust2', 'cust3'}
print(day1 | day2)
```

Sets can be used to quickly find all unique customers across multiple days.

4. Dictionaries

Example 1: Creating a dictionary

```
student = {'name': 'Sam', 'age': 20}
print(student['name'])
```

Dictionaries store key-value pairs. Here 'name' is the key.

Example 2: Adding a key-value pair

```
student['grade'] = 'A'
print(student)
```

You can add new key-value pairs by assignment.

Example 3: Updating values

```
student['age'] = 21
print(student)
```

Reassigning a key updates its value.

Example 4: Iterating through dictionary items

```
for key, value in student.items():
    print(key, value)
```

items() lets you loop through both keys and values.

Example 5 (Business): Product prices

```
prices = {'pen': 10, 'book': 40}
print(prices['pen'])
```

Dictionaries are commonly used to store product-price mappings.

5. Control Flow Statements

Example 1: If-Else statement

```
x = 10
if x > 5:
    print('Greater')
else:
    print('Smaller')
```

If-Else allows conditional execution.

Example 2: Elif chain

```
score = 75
if score > 90:
    print('A')
elif score > 60:
    print('B')
else:
    print('C')
```

Elif is used for multiple conditions.

Example 3: For loop

```
for i in range(3):
    print(i)
```

For loops repeat a block of code for a sequence of values.

Example 4: While loop

```
count = 0
while count < 3:
    print(count)
    count += 1
```

While loops keep running until the condition is false.

Example 5 (Business): Checking inventory stock

```
stock = 5
if stock == 0:
    print('Out of stock')
```

```
else:
    print('Available')
```

Control flow is useful to check business conditions like inventory levels.

6. List Comprehensions

Example 1: Squares of numbers

```
squares = [x*x for x in range(5)]
print(squares)
```

List comprehensions provide a short way to build lists.

Example 2: Filtering even numbers

```
evens = [x for x in range(10) if x%2==0]
print(evens)
```

Conditions can be added to filter values.

Example 3: Converting strings to uppercase

```
words = ['data', 'python']
upper = [w.upper() for w in words]
print(upper)
```

You can transform each element in a list comprehension.

Example 4: Flattening a matrix

```
matrix = [[1,2],[3,4]]
flat = [num for row in matrix for num in row]
print(flat)
```

Nested comprehensions can flatten multi-dimensional lists.

Example 5 (Business): Extracting active product IDs

```
products = [{'id':1,'active':True},{'id':2,'active':False}]
active = [p['id'] for p in products if p['active']]
print(active)
```

Comprehensions are handy for filtering and extracting business data quickly.