# CS65: Introduction to Computer Science

Sequence
The for Loop


Md Alimoor Reza
Assistant Professor of Computer Science

# Topics

- Sequence
  - Strings
  - List

- Two different ways to solve a repetitive task in Python
  - The **for** loop
  - The **while** loop - we already covered

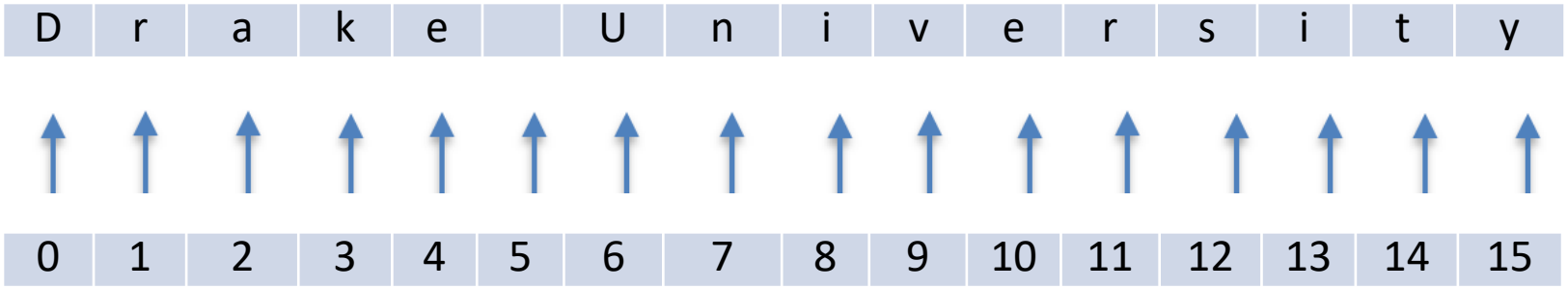
# Sequence: Strings

- Sequence is an ordered group of elements (numbers, characters, etc)

- String is a sequence of characters
  - "Drake University"
  - "cs65:introduction_to_computer_science!"

- *length* of a String is the total number of characters

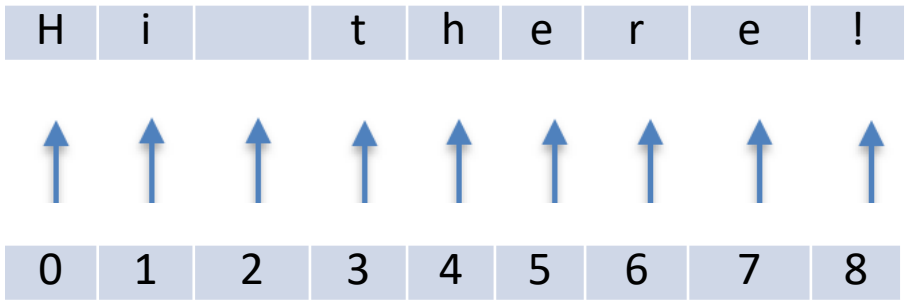
- Each position in a sequence is marked with an **index** or **position**
  - Starts (from left) at position *0* and ends at position (*length-1*)
  - Start indexing from the *left* to *right*



# Strings

- String is a sequence of characters
  - “”
  - “Hi there!”

- Each position is marked with an **index**
  - What are the *lengths* of the strings above?
  - Starts (from left) at position *0* and ends at position (*length-1*)



# Strings

- String is a sequence of characters
  - "Drake University"
  - "cs65:introduction_to_computer_science!"

- Each position in a sequence is marked with an **index** or **position**
  - Starts (from left) at position *0* and ends at position (*length-1*)
  - Start indexing from the *left* to *right*
  - Python reports with an **IndexError** if the index goes out of bound

# Length of a Sequence

- String is a sequence of characters
  - “”
  - “Hi there!”

- How can you find the length of a string?
  - Use built-in ***len()*** function

# Demo: Length of a Sequence

- How can you find the length of a string?
  - Use built-in ***len()*** function

```
my_string1 = "hello@world"
my_string2 = "Hi there!"
my_string3 = ""

print("Length of \"hello@world\" is: ", len(my_string1))
print("Length of \"Hi there!\" is: ", len(my_string2))
print("Length of \"\" is: ", len(my_string3))
```

```
Shell
Python 3.7.9 (bundled)
>>> %cd /Users/reza/Class_and_Resea
    slides/lecture10
>>> %Run lec10_demo.py

  Length of "hello@world" is:  11
  Length of "Hi there!" is:  9
  Length of "" is:  0

>>>
```

# Accessing Sequence Items with **Positive** Index
# Left — — —> Right

- String is a sequence of characters

```
my_string1 = "Drake University"
```

- Access a specific item by appending ***brackets []*** containing an index

```
my_string1[0]     to access D
my_string1[1]     to access r
my_string1[2]     to access a
...
...
...
my_string1[15]    to access y
```

# Accessing Sequence Items with **Negative** Index
# Left <— — — Right

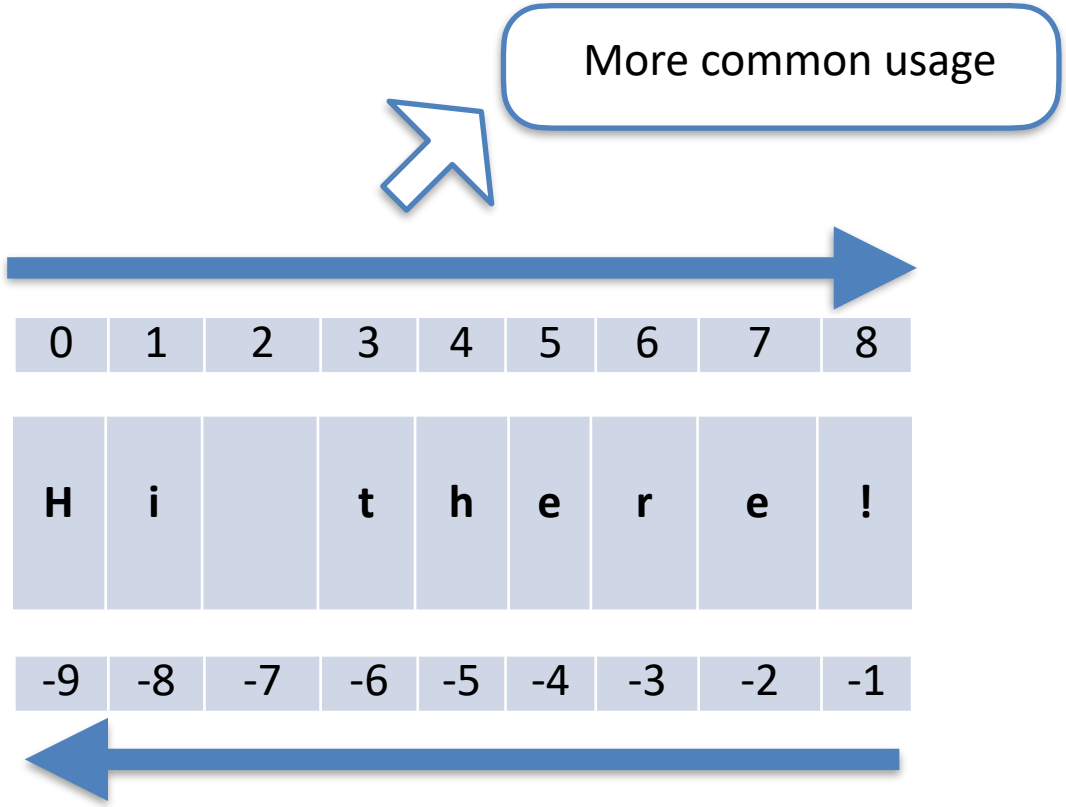
- String is a sequence of characters and negative indexing begins at the end with a **-1** (not zero anymore)

```
my_string1 = "Drake University"
```

- Access a specific item by appending ***brackets []*** containing an index

```
my_string1[-1]    to access  y
my_string1[-2]    to access  t
my_string1[-3]    to access  i
…
my_string1[-16]   to access  D
```

# Summary of Indexing



# Demo: Accessing Items with Index or Position

- How can you access an item in a sequence?
  - Use ***variable_name[index]***

```
15  # ----------------------------------------
16  # demo 2 accessing elements in a string
17  my_string1 = "Drake University"
18  my_string2 = "Hi there!"
19
20  vis = 1
21  if (vis):
22      print("Character at index = 0 is ", my_string1[0])
23      print("Character at index = 1 is ", my_string1[1])
24      print("Character at index = 2 is ", my_string1[2])
25      print("Character at index = 15 is ", my_string1[15])
26
27
```

Shell

```
>>> %Run lec10_demo.py

Character at index = 0 is  D
Character at index = 1 is  r
Character at index = 2 is  a
Character at index = 15 is  y
```

# Sequence: List

- Sequence is an ordered group of elements (numbers, characters, etc)

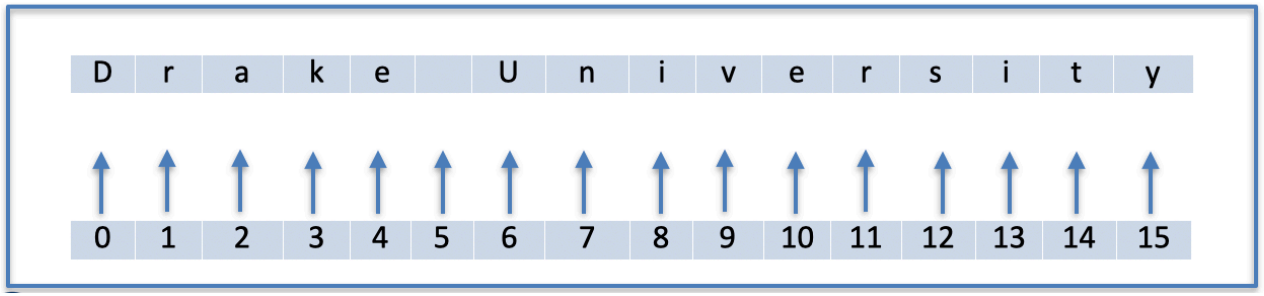
- **List** is another type of sequence whose members can be i) numbers, ii) strings, or iii) even another list
  - `["Drake University", "hello", "world"]`
  - `[1, 2, 3, 4, 5]`
  - **List will be discussed in greater detail in a separate lecture**

- Recall that **String** is a type of sequence whose members are characters
  - `"Drake University"`
  - `"cs65:introduction_to_computer_science!"`

# Poll: String and index

- Please participate in poll below
  - **https://tinyurl.com/2p83m3jp**

Previous example's reference in case that is helpful!



# Topics

- Sequence
- Two different ways to solve a repetitive task in Python
  - The **for** loop
  - The **while** loop

# Solving Repetitive Task with for Loop

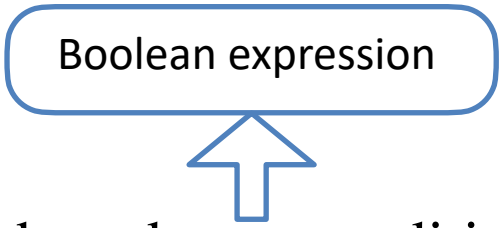
- Designed to solve a repetitive task — runs a block of code for a finite number of times

- Why do we need this alternative to while loop?
  - When we need to iterate for a finite number: **count-controlled**
  - When the location information is important for a task
  - When we need to access or update locations of a sequence
    - from beginning to end
    - from end to beginning

# Solving Repetitive Task with **for** loop

`for` loop

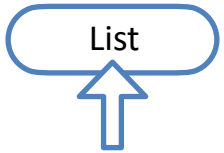
- use it when there is a **fixed** and **finite number of iterations**
  - "Do a calculation *10* or *N* times"
  - "Do a calculation from first to last item in a sequence"




`while` loop

- use it for **an indefinite number** of iterations based on a condition:
  - "Do until user enters END"
  - "Do until the number becomes negative"
  - "Do until we reach the end of the file with a special marker"

# Syntax of **for** loop



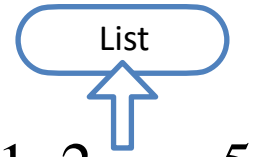

- **for** variable **in** [$val_1$, $val_2$, …, $val_5$] **:**
  **statements**

- This is also called **value for loop**
  - There is another form called **index for loop**

- Statements will be repeated sequentially from first to last item in a sequence (here it will be repeated 5 times since there are 5 numbers in the List)

  - Iteration 1: variable will be assigned **$val_1$**
  - Iteration 2: variable will be assigned **$val_2$**
  - …
  - Itera

# Syntax of **for** loop: concrete example

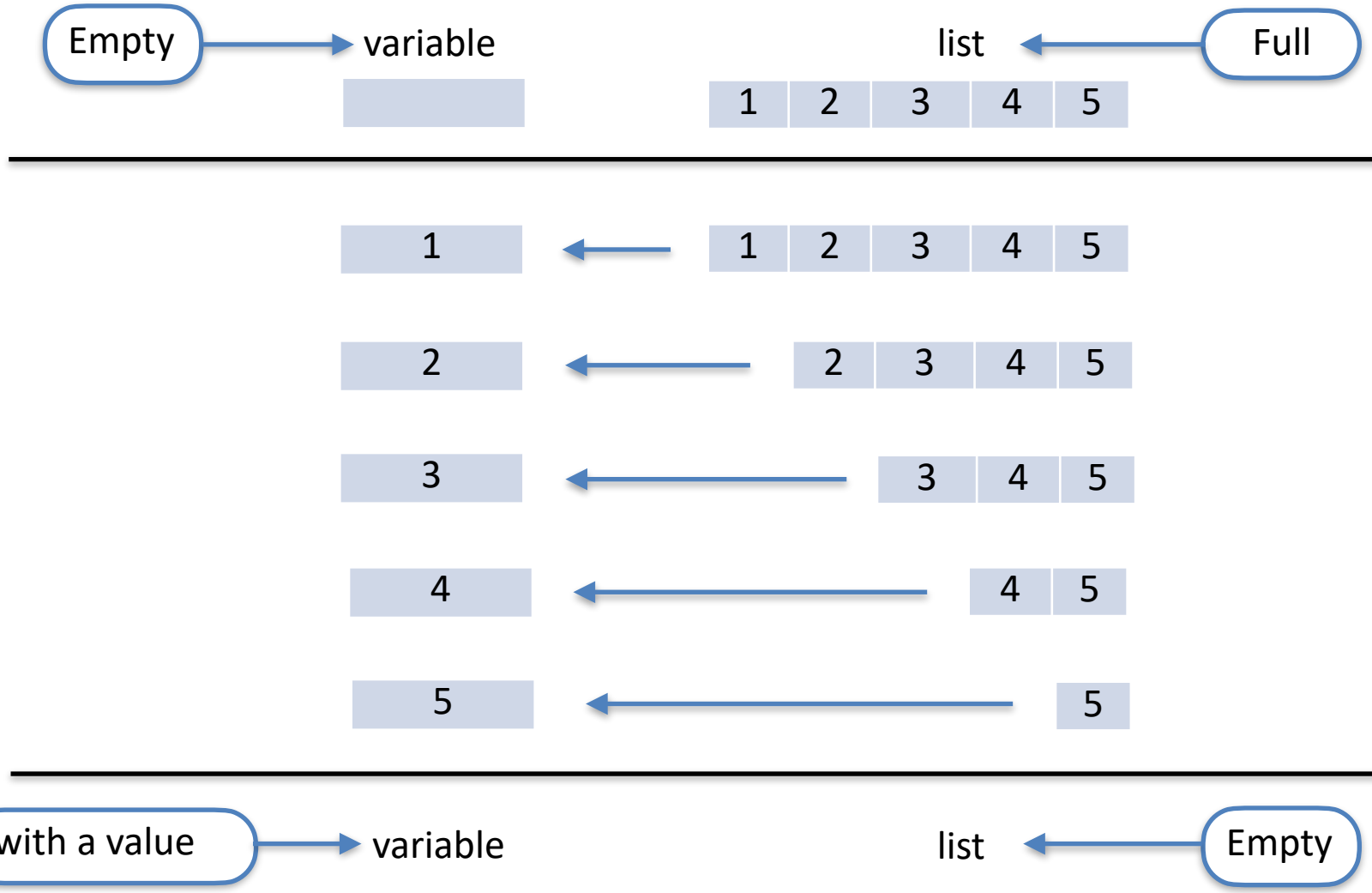



- **for** var **in** [1, 2, …, 5] **:**
  **statements**

- Statements will be repeated sequentially from first to last item in a sequence (here it will be repeated 5 times since there are 5 numbers in the List)
  - Iteration 1: var will be assigned a value of **1**
  - Iteration 2: var will be assigned a value of **2**
  - …
  - Iteration 5: var will be assigned a value of **5**

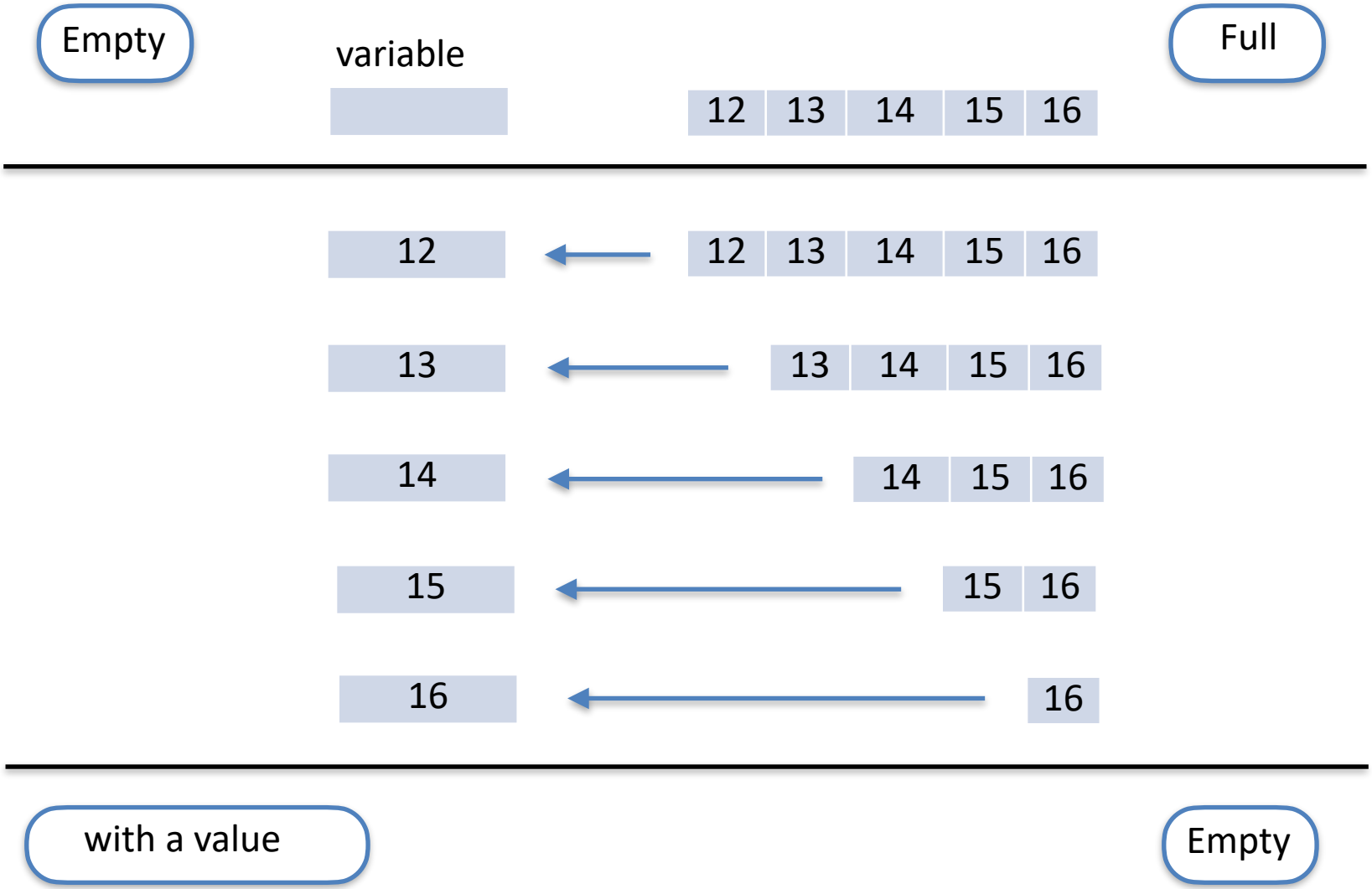
# **For** loop: concrete visualization

```
for variable in [1, 2, …, 5] :
    statements
```



# **For** loop: concrete visualization

```
for var in [12, 13, 14, 15, 16]:
    print("current num is: ", var)
```



# Demo: **for** loop

- **for** var **in** [1, 2, …, 5] **:**
  **statements**

- Python code: here statement is a simple print() function call

```
for var in [1, 2, 3, 4, 5]:
    print("current num is: ", var)
```

```
>>> %Run lec10_demo.py

  current num is:  1
  current num is:  2
  current num is:  3
  current num is:  4
  current num is:  5
```

# Demo: **for** loop with sequence of strings

- **for** var **in** [“one”, “two”, “three”, “four”, “five”] **:**
  **statements**

- Python code: here statement is a simple print() function call

```
for var in ["one", "two", "three", "four", "five"]:
    print("current num is: ", var)
```

```
>>> %Run lec10_demo.py

  current num is:  one
  current num is:  two
  current num is:  three
  current num is:  four
  current num is:  five
```

# Demo: **for** loop doing more than mere print

- `for` var `in` [1, 2, 3, 4, 5] `:`
  new_var = var*10
  print("10 times", var, " is ", new_var)

- Python code

```
for var in [1, 2, 3, 4, 5]:
    new_var = var*10
    print("10 times", var, " is: ", new_var)
```

```
>>> %Run lec10_demo.py

  10 times 1  is:  10
  10 times 2  is:  20
  10 times 3  is:  30
  10 times 4  is:  40
  10 times 5  is:  50
```

# Syntax of `for` loop vs Syntax of `while` loop

- **for** variable **in** [$val_1$, $val_2$, …, $val_{15}$] **:**
  **statements**

- Statements will be repeated sequentially from first to last item in a sequence

checking a condition

- **while condition expression :**

  **statements**

- **condition expression**: a boolean expression

- **statements** will repeatedly be executed until the **condition expression** becomes False

# Function `range`()

- The `range`() function simplifies the process of for loop writing
- Creates a sequence of numbers on the fly
- These numbers can be used to index the sequence
- It can be called with several variations

```
print("range() function version 1:")
for var in range(5):
    print(var)
```

```
print("range() function version 2:")
for var in range(0, 5):
    print(var)
```

```
print("range() function version 3:")
for var in range(0, 10, 2):
    print(var)
```

# Demo: Function `range`()

- The `range`() function simplifies the process of for loop writing
- It can be called with several variations

```
# version 1:
print("range() function version 1:")
for var in range(5):
    print(var)


# version 2: start, stop
print("range() function version 2:")
for var in range(0, 5):
    print(var)


# version 3: start, stop, step_size
print("range() function version 3:")
for var in range(0, 10, 2):
    print(var)
```

# Value `for loop` vs Index `for loop`

- So far we have seen the syntax of **value** `for loop`

```
for var in [10, 20, 30, 40, 50]:
    print(var)
```

- There is another form called **index** `for loop`

```
my_list = [10, 20, 30, 40, 50]
length = len(my_list)
for i in range(length):
    print( my_list[i] )
```

common practice is to name the index variables with **i, j, k**

# Value `for` loop vs Index `for` loop

- **value** `for loop`
  - directly assigns a value to the variable from the sequence
  - don't keep track of the indices
  - we have access to only value
    - **good**

- **index** `for loop`
  - generates all the indices of all elements in the list
  - each element can be accessed indirectly by that index
  - we have access to both *i) index* and *ii) value*
    - **better**

# Exercise 1:

- Write a code that will do the following:
  - prompts the user for an integer (between 1 to 100)
  - then **computes** the sum of all number from 0 to the given number

- You have done it using while loop last time, now try it with for loop

# Exercise 2

- Write a code that will do the following:
  - Prompts the user for an integer number (between 1 to 100)
  - then **prints** all the even numbers between 0 and the given number

- You have done it using while loop last time, now try it with for loop