# Loops In Python: Part 2

- Basic introduction into the use of the for loop
- Application of loops for lists and strings

James Tam

## Loops In Python

- Already covered (last section): `While`-loop
  - The most flexible (powerful) type of loop.
  - It can be used almost any time repetition is needed.
    - Situations when it can't be used are very specific (when back tracing during 'recursion' is needed).
- New (this section): `for`-loop
  - **Python**: can be used when the program can step through ('iterate') through a sequence.
    - E.g. 1: count through a numerical sequence (1, 2, 3…)
    - E.g. 2: the sequence of characters in a string
    - E.g. 3: the sequence of lines in a file.
  - **Strength of python**:
    - With most other languages `for-loops` can only count through a numerical sequence (5, 25, 125…). Consequently referred to as "counting loops".
    - With python `for-loops` they can not only count through (iterate) a sequence but also iterate through other things as well e.g. read in lines in a text file
  - **Drawback of python**: Python `for-loops` can only count through a sequence using addition or subtraction (with the application of bad style one can force a `for-loop` to execute as a `while-loop` (awful programming style).

James Tam

## General Use: The For Loop

- In Python a `for`-loop is used to iterate (step through) a sequence e.g., count through a series of numbers or step through the lines in a file.
- **General syntax:**

```
for <name of loop control> in <something that can be
 iterated>:
    body
```

- **Syntax, counting loop (steps through number sequence):**

```
for <name of loop control> in range():
    body
```

- **Example, counting loop (steps through number sequence):**

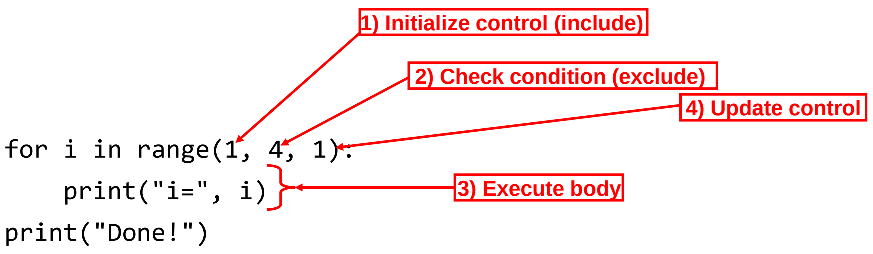
```
for i in range(1,4,1):
    print("i=", i)
```

  - The python for-loop is **used when it's known in advance** (before loop executes) **how many times it will execute.**

James Tam

## Example Use Of The For Loop

- **Program name:** `1for_counting_up.py`
- Learning objective: a simple for counting loop stepping through a sequence (1 - 3)



```
for i in range(1, 4, 1):
    print("i=", i)
print("Done!")
```

James Tam

## Example Use Of The For Loop

- **Program name:** `1for_counting_up.py`
- Learning objective: a simple for counting loop stepping through a sequence (1 - 3)

```
for i in range(1, 4, 1):
    print("i=", i)
print("Done!")
```

An equivalent While-loop:

```
i = 1
while(i < 4):
    print("i=", i)
    i = i + 1
print("Done!")
```

- Loop executes: when control (i) is within range e.g. 1..3 (initial value and after update).
- Loop ends: when control is not within the range.

James Tam

## Counting Down With A For Loop

- **Program name:** `2for_counting_down.py`
- Learning objective: a simple counting loop stepping down through a sequence (3 - 1)

```
for i in range(3, 0, -1):
    print("i = ", i)
print("Done!")
```

An equivalent While-loop:

```
i = 3
while(i > 0):
    print("i=", i)
    i = i - 1
print("Done!")
```

**Reminder:** the python for-loop cannot do anything other than count up (add) or down (subtract).

- But you can add or subtract by **values other than one**

  e.g. (`0,5,10…90.95,100`)   `for i in range(0,105,5):`

James Tam

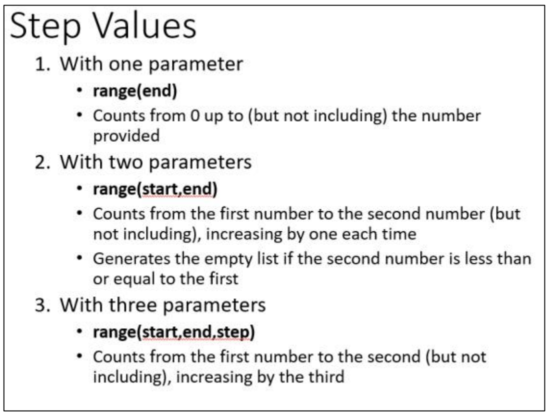
## In Range()

- Sometimes referred to as a 'type' sometimes referred to as a function.
  - Type ("Sequence types"): https://docs.python.org/3/library/stdtypes.html
  - Function: https://docs.python.org/3/library/functions.html
- To follow good programming conventions you should make your code as self explanatory as possible.
- With respect to Range() you should specify it with all 3 values
- **Format:**
  `range(start value, end value, update value)`
- **Example** (iterate through 1-3):
  `range(start value, end value, update value)`
- Unfortunately you may have to work with code that may not follow all style conventions.
  - Python stipulates that only the 2nd value is mandatory, the 1st and 3rd values are optional.
  - 1st value excluded: default starting value of **0** is used e.g. range(3) **#0..2**
  - 3rd value excluded: default update value of **1** (positive one) is used. e.g. range(3) **#0,1,2 (automatically increases by 1)**

James Tam

## Alternative Rules Of Thumb

- From the lecture notes of Michelle Cheatham.



Step Values

1. With one parameter
   - **range(end)**
   - Counts from 0 up to (but not including) the number provided
2. With two parameters
   - **range(start,end)**
   - Counts from the first number to the second number (but not including), increasing by one each time
   - Generates the empty list if the second number is less than or equal to the first
3. With three parameters
   - **range(start,end,step)**
   - Counts from the first number to the second (but not including), increasing by the third

James Tam

## Students-Do: Program Traces

- All values specified.

```
for i in range(1,4,1):
    print(i, end=" ")
```

- 3rd value omitted

```
for i in range(1,4):
    print(i, end=" ")
```

- 1st and 3rd value omitted

```
for i in range(4):
    print(i, end=" ")
```

James Tam

## Students Do: Code Writing

- Write a python program which uses loops to calculate these math operations.
  - The idea is for you to learn how to develop your skill at writing a program that uses loops.
  - Consequently you are not to use pre-created function e.g. `pow()` or `factorial()`.
  - Nor should you use the exponent operator: **
  - Program 1: prompt the user for a base and power and the program can calculate the exponent.
  - Program 2: calculate the factorial for any user entered integer zero or greater (FYI: 0! = 1)

James Tam

Cover after midterm

## But Wait: The Python Loop Can Do More!

- **Python**: can be used when the program can step through ('iterate') through a sequence.
  - E.g. 1: count through a numerical sequence (1, 2, 3…)
  - E.g. 2: the sequence of characters in a string
  - E.g. 3: the sequence of lines in a file
- **Examples 2 & 3 illustrate the strength of python**:
  - With most other languages `for-loops` can only count through a numerical sequence (5, 25, 125…). Consequently referred to as "counting loops".
  - With python `for-loops` they can not only count through (iterate) a sequence but also iterate through other things as well e.g. read in lines in a text file

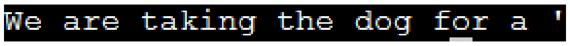
James Tam

Cover after midterm

## While Loop: Stepping Through A Sequence Characters

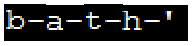
- **Program name:** `3while_iterating_string.py`
  - Learning objectives:
    - How to access the characters in a string using an index
    - Using a while loop stepping through a sequence in a string

```
activity = input("What are you doing with dog now: ")
print("We are taking the dog for a '", end="")
```



```
We are taking the dog for a '
```

```
i = 0
ch = activity[i] #i=0 here so accessing 1st character: b
aLength = len(activity) #len returns 4 with string: bath
#Display characters at indices 0-3 using loop
while(i<aLength):
    print(activity[i] + "-", end="")
    i = i + 1
```



```
b-a-t-h-'
```

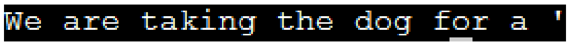
James Tam

## For Loop: Stepping Through A Sequence ( Characters

Cover after midterm

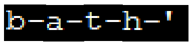
- **Program name:** `4for_iterating_string.py`
  - Learning objective: a for loop stepping through a sequence in a string

```
activity = input("What are you doing with dog now: ")
print("We are taking the dog for a '", end="")
```



```
We are taking the dog for a '
```

```
for ch in activity:
    print(ch + "-", end="")
print("'")
```



```
b-a-t-h-'
```

James Tam

## New Type Of Variable: List

Cover after midterm

- This is only a very basic introduction.
  - For the keeners: more details will come later.
- **String**: consists of individual elements that can be accessed via an index (zero to length of the string minus one) `s1 = "Jim tam"`

```
0 1 2 3 4 5 6
Jim tam
```

- **List:** need not consist only of characters nor does it have to be homogeneous (e.g. all integers, all Booleans)
  - i.e. Python lists can be heterogeneous
  - `list1 = [1, "a",True]`

```
0   1   2
1   a   True
```

James Tam

Cover after midterm

## Creating A List (Fixed Size)

- **Format ('n' element list):**

```
<list_name> = [<value 1>,  <value 2>, ... <value n>]
               Element 0   Element 1      Element n-1
```

**Example:**

```
#List with 5 elements, index ranges from 0 to (5-1)

percentages = [50.0, 100.0, 78.5, 99.9, 65.1]
                0      1     2     3     4
```

**Other Examples:**

```
letters = ["A", "B", "A"]
names = ["The Borg", "Klingon ", "Hirogin", "Jem'hadar"]
```

1 These 4 names (Borg, Klingon, Hirogin, Jem'hadar) © are CBS

James Tam

Cover after midterm

## Accessing/Displaying A List

- Because a list is composite you can access the entire list or individual elements.

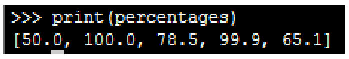
List

```
percentages = [50.0, 100.0, 78.5, 99.9, 65.1]
```

Elements

- Name of the list accesses the whole list

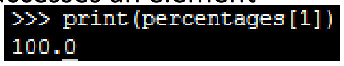
```
print(percentages)
```



```
>>> print(percentages)
[50.0, 100.0, 78.5, 99.9, 65.1]
```

- Name of the list and an index "`[index]`"accesses an element

```
print(percentages[1])
```



```
>>> print(percentages[1])
100.0
```

James Tam

Cover after midterm

## Basic List Operations

- **Name of the online example**: `5modifying_displaying_list`
- Common list operations:
  - **Create a new fixed size list:**
    ```
    aList = [2,6,2]
    ```
  - **Displaying entire list:**
    ```
    i = 0
    size = len(aList)
    while(i < size): #i takes on values from 0 – (size-1)
        print(aList[i], end=" ")
        i = i + 1
    ```
  - **Modifying a single element**
    ```
    aList[size-1] = 3
    ```
  - **Modifying all elements**
    ```
    while(i < size): #i takes on values from 0 – (size-1)
        aList[i] = aList[i] * 2
        i = i + 1
    ```

James Tam

Cover after midterm

## Additional List Operations

- **Name of the online example**: `6adding_2_end_modify_select_while`

```
aList = ["A","a","z","B"]
```

New list operations:

- **Adding new elements: adding new elements to end** (append method):
  ```
  aList.append(ch)
  ```
- **Modifying select elements (based upon a condition):**
  ```
  i = 0
  size = len(aList)
  while(i < size): #A=ASCII 65, Z=90
      if((aList[i]>="A") and (aList[i]<="Z")):
          aList[i] = aList[i] + "!" #Applies to caps only
      i = i + 1
  ```

James Tam

Cover after midterm

## For Loops Can Be Used To Iterate Lists

- **Name of the online example**: `7adding_2_select_for`

```
aList = ["A","a","z","B"]
```

- Iterating list using a for-loop:

```
for ch in aList:
    print(ch)
```

James Tam

Cover after midterm

## Students-Do: Programming Problems

- You can find some online examples of extra problems to work through at the following URL (simple 1D lists):
  - https://cspages.ucalgary.ca/~tam/2025/217F/exercises/1D_lists/

James Tam

## After This Section You Should Now Know

- When and why are loops used in computer programs.
- How to trace the execution of a while-loop.
- How to properly write the code for a while-loop in a program.
- What is a sentinel controlled loop and when should they be employed.
- How to access the individual elements of a string.
- How to use basic operations of a new type of variable (list):
  - Creating a new fixed size list.
  - Stepping through the entire list.
  - Accessing/modifying list elements.
  - Display an entire list.
  - Modifying the elements of a list.
  - Modifying select elements of a list.

James Tam

## Copyright Notification

slide 22

James Tam