

# STOP 0 – Debugging & Test Cases

---

- Debugging
  - Log analysis
- Income tax bounty
  - Identifying appropriate test cases
    - Boundaries
      - On, above and below boundary values
      - Six tax bands
      - Two age ranges— $<65$  and  $> 65$





# *UCT Department of Computer Science*

## *Computer Science 1017F*

# Lists



*Lighton Phiri <lphiri@cs.uct.ac.za>*  
*April 2015*

# STOP 1 – Data Types

---

- What is a data type
  - ?
- List Python native data types
  - ?



# Introduction

---

- ❑ Data structures are used to store collections of information by compounding them in a single variable
- ❑ Python has a number of native data structures
  - Lists []
  - Tuples ()
  - Sets {}
  - Dictionaries {:}
- ❑ Data structures can store items of different types
  - Items are separated by commas
- ❑ Lists and dictionaries are most commonly used
  - For the purposes of this course, we shall NOT look at tuples



# Lists

---

- ❑ Lists are used to store an ordered collection of elements
- ❑ Elements can be of any type or even another data structure
- ❑ Lists are optionally defined using square brackets []
  
- ❑ Lists are indexed, like strings, from 0
  - List items/elements can be referenced using [index]
  - Last item/element???
- ❑ Lists are mutable
  - Most data types we have used thus far are immutable!



# Creating Lists

---

- There are a number of ways to create a list

```
1 var_list = [1, 2, 3, "CSC1017F", True, ["a", "b"]]
2 type (var_list) # <class 'list'>
3
4 var_another_list = list(range(10))
5 type (var_another_list) # <class 'list'>
6
7 var_yet_another_list = list("This is a list")
8 type (var_yet_another_list) # <class 'list'>
```

- Defined with multiple types
- Notice Line 7



# STOP 2 – Palindrome Primes

---

```
1 def fxn_primes (a, b):
2     result = []
3     # LOGIC
4     return result # list
5
6 def fxn_reverse (a, b)
7     result = []
8     # LOGIC
9     return result
10
```

```
1 def fxn_palip (a_1, b_1):
2     # LOGIC
3     # print out results
4
5
6 var_1 = eval(input("a:"))
7 var_2 = eval(input("b:"))
8
9 fxn_palip(var_1, var_2)
10
```



# List Operations

---

## □ Concatenation

```
1 [1, 2] + ["a", "b"] # [1, 2, "a", "b"]
```

## □ Repetition

```
1 [1, 2] * 2 # [1, 2, 1, 2]
```

## □ Indexing

```
1 [1, 2, 3][-1] # 3
```

## □ Slicing

```
1 [1, 2, 3][::-1] # [3, 2, 1]
```

## □ Iteration

```
1 for i in [1, 2, 3]: # This is incomplete!
```

## □ Membership check

```
1 "a" in [1, 2, "b"] # False
```





# STOP 3 – Methods vs Functions

---

```
1 import math
2
3 def add (a, b):
4     return a+b
5
6 math.pow(10, 2) # function invocation from module
7 add (1, 1) # standalone function invocation
8 course_name = "Singh, Shekhar"
9 course_name.upper () # method call—class function
invocation
```

❑ What is a function?

❑ What is similar in lines 6, 7& 9? What is different?



# List Methods

---

## □ `<list>.append(x)`

- Adds an item at the end of list
- Parameter(s): item of any data type; Return value: None

```
1 var_list = [1, 2, 3, 4]
2 var_list.append(5) # [1, 2, 3, 4, 5]
```

## □ `<list>.extend(x)`

- Extend list by appending items in given iterable
- Parameter(s): iterable data type; Return value: None

```
1 # var_list: [1, 2, 3, 4, 5]
2 var_list.extend("CT") # [1, 2, 3, 4, 5, "C", "T"]
```



# List Methods

---

## □ `<list>.insert(i, x)`

- Insert item at specified position
- Parameter(s): index& item to insert; Return value: None

```
1 # var_list: [1, 2, 3, 4, 5, "C", "T"]
2 var_list.insert(len(var_list), 6) # [1, 2, 3, 4, 5,
  "C", "T", 6]
```

## □ `<list>.remove(x)`

- Remove first occurrence of item
- Parameter(s): item; Return value: None

```
1 # [1, 2, 3, 4, 5, "C", "T", 6]
2 var_list.remove("C") # [1, 2, 3, 4, 5, "T", 6]
```



# List Methods

---

## □ `<list>.pop(<i>)`

- Remove item at specified position and return it; remove last item if no argument specified
- Parameter(s): Optional item index; Return value: item at index

```
1 # var_list: [1, 2, 3, 4, 5, "T", 6]
2 var_list.pop(???) # 6
```

## □ `<list>.index(x)`

- Return index of first item with value x
- Parameter(s): item; Return value: int

```
1 # [1, 2, 3, 4, 5, "T"]
2 var_list.index("T") # ???
```



# List Methods

---

## □ `<list>.count(x)`

- Return number of occurrences of x
- Parameter(s): item; Return value: number of occurrences

```
1 # var_list: [1, 2, 3, 4, 5, "T"]
2 var_list.count("X") # ???
```

## □ `<list>.sort(key=None, reverse=False)`

- Sort items in list
- Parameter(s): Optional parameters; Return value: None

```
1 # [1, 2, 3, 4, 5]
2 var_list.sort(reverse=True) # [5, 4, 3, 2, 1]
```



# List Methods

---

## □ `<list>.reverse()`

- Reverse the elements in list
- Parameter(s): No parameters; Return value: None

```
1 # var_list: [5, 4, 3, 2, 1]
2 var_list.reverse() # [1, 2, 3, 4, 5]
```

## □ `<list>.copy()`

- Return copy of list
- Parameter(s): No parameters; Return value: List

```
1 # [1, 2, 3, 4, 5]
2 var_list.copy() # [1, 2, 3, 4, 5]
```



# List Methods

---

## □ `<list>.clear()`

- Purge all items from list

- Parameter(s): No parameters; Return value: None

```
1 # var_list: [1, 2, 3, 4, 5]
```

```
2 var_list.clear() # []
```



# STOP 4 – CSC1017 Vula DB 1/2

---

- DB records are in format below

```
['Marufu,  
Anesu;mrfmuf001;MRFMUF001@myuct.ac.za;Support staff',  
'Mbogo, Chao;mbgcha002;MBGCHA002@myuct.ac.za;Support  
staff', 'Zhou, Yin  
Hong;zhxyin002;ZHXYIN002@myuct.ac.za;Student', 'Zide,  
Zikhona;zdxzik001;ZDXZIK001@myuct.ac.za;Student',  
'Zimuto,  
Tanaka;zmttan001;ZMTTAN001@myuct.ac.za;Student']
```





# STOP 4 – CSC1017 Vula DB 2/2

---

- Using list methods can we figure out the following?
  - How many students, staff and tutors are there?
  - Can we find students with first names: Ian & Dorothy?
  - Let's rename 'Zhou, Yin Hong' to 'Zhou Yin'
  - Can we purge those that have dropped off from course?

