Hour 1

Lists review:

- Access list elements by using index and square brackets: `list[0]` is first element
- Can also be created using `list()` function
- Heterogenous lists are lists with different datatype values stored in it.
- Nested lists are lists inside other lists
- Empty lists are lists without any elements

List Slicing:

- Slice of list is a subset of another list
- Colon operator can be used to slice: `list[start:end:step]`

List Chopping:

- Make a function that chops the list.
- Chopping a list means to remove first and last element of the list
- Use pop to actually remove the item from the list and not just the create a copy
- Similarly del operator can be used as `del a[1]`

```python
def chop(lst):
    """function removes the first and last elements of a list. It mutates
        incoming list.
    """

    # Base cases

    # Empty list
    if not lst: return

    # List with just 1 element
    if len(lst) < 2: lst.pop(0)

    # For all lists with more than 1 element
    lst.pop(0)
    lst.pop(-1)
```

Design:

- Meta-data: data that keeps track of other data.

Hour 2

Refactoring grades program with meta-data:

```python
"""
    Refractor the grade program to make us rich!
"""

def main():
    grades = []
    my_grade = float(input("Enter a grade, negative to stop: "))

    while my_grade >= 0:
        semesters = ["F22", "F23", "SP23", "SP24"]
        semester = input(f"Which semester {semesters}: ")
        if semester.upper() not in semesters:
            continue

        found = False
        for grade in grades:
            if semester.upper() in grade: # Semester exists already
                grade.append(my_grade)
                found = True
                break

        if not found:
            grades.append([semester.upper(), my_grade])

        my_grade = float(input("Enter another grade, negative to stop: "))

    print(grades)

if __name__ == "__main__":
    main()
```

- Here, found is called a flag variable. It keeps track of a boolean value and is a common practice to use for conditionals

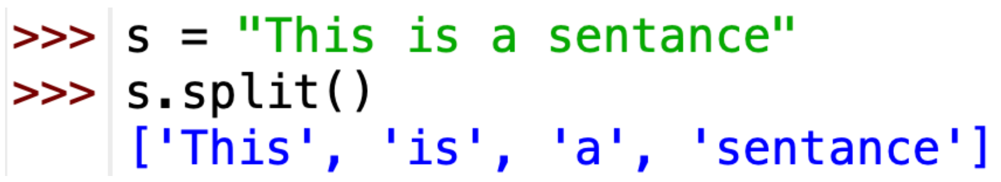
Strings:

- Sequence of characters
- Immutable, so cannot be changed
- Negative indices can be used to call characters of string from last
- 2 strings can be concatenated by using + operator.

Common String Methods:

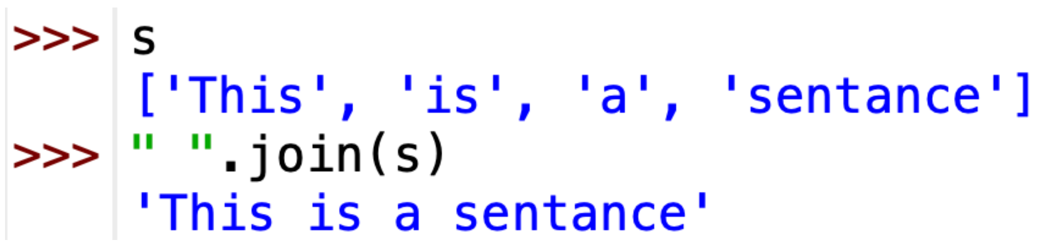
upper ( )	Converts a string into upper case
title ( )	Converts the first character of each word to upper case
find ( )	Searches the string for a specified value and returns the position of where it was found
count( )	Returns the number of times a specified value occurs in a string
replace( )	Returns a string where a specified value is replaced with a specified value

`string.split()` : Splits the string with words in a list if no delimiter specified.



```
>>> s = "This is a sentance"
>>> s.split()
['This', 'is', 'a', 'sentance']
```

`string.join()` : Joins the list with specified delimiter and coverts to string.



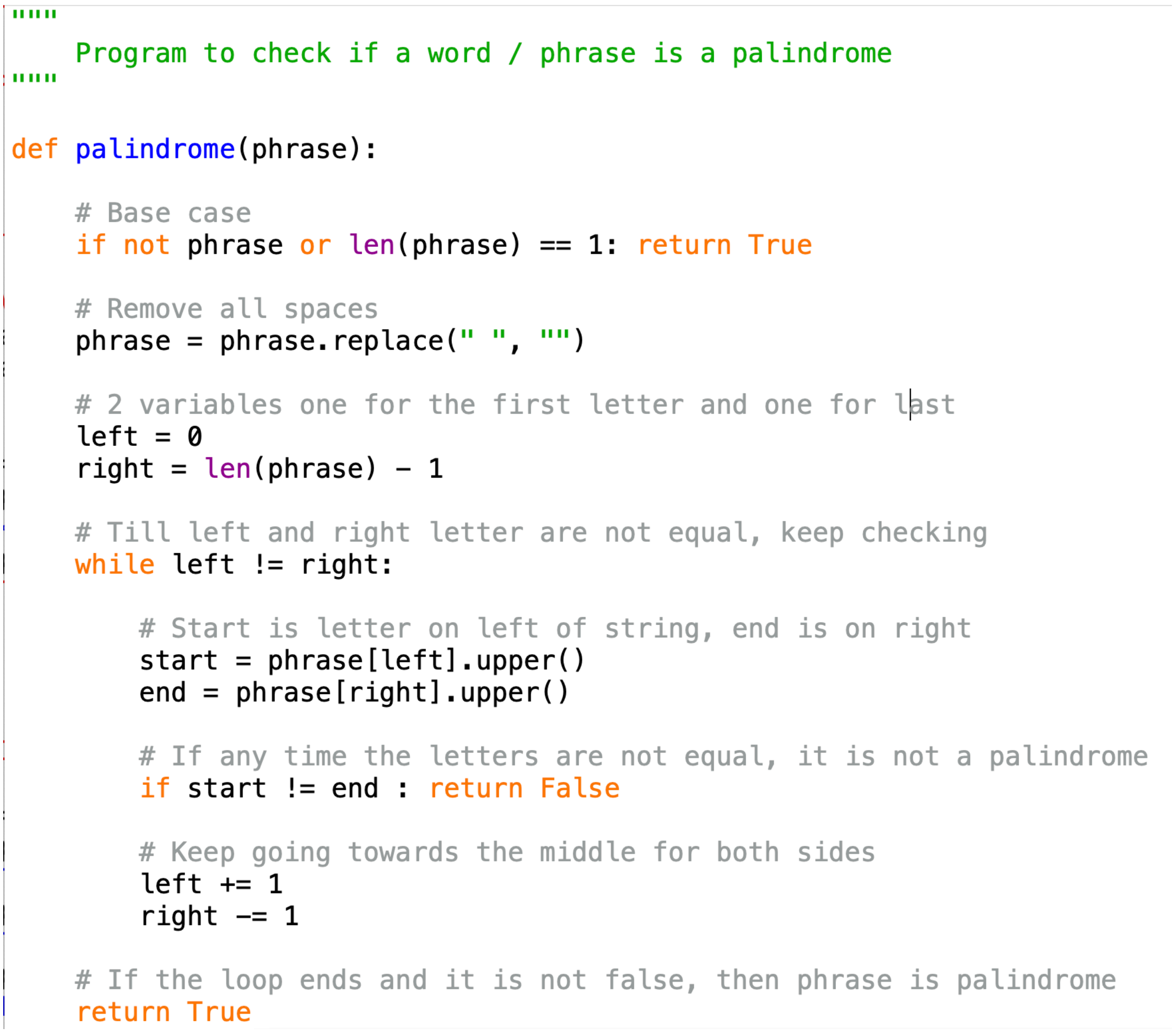
```
>>> s
['This', 'is', 'a', 'sentance']
>>> " ".join(s)
'This is a sentance'
```

Hour 3

Binary to decimal conversion program:

```python
"""
    Binary digits to decimal number
"""

def decimal(binary):

    binary = binary[-1::-1]
    decimal = 0

    for i in range(len(binary)):
        decimal += int(binary[i]) * (2 ** i)

    return decimal

def main():

    input_num = input("What is the binary number you want to convert (Q to quit): ")

    while input_num.upper() != "Q":

        print(f"Decimal is: {decimal(input_num)}")
        input_num = input("Another binary number to convert (Q to quit): ")


if __name__ == "__main__":
    main()
```

Valid Palindrome Program:



```python
"""
    Program to check if a word / phrase is a palindrome
"""

def palindrome(phrase):

    # Base case
    if not phrase or len(phrase) == 1: return True

    # Remove all spaces
    phrase = phrase.replace(" ", "")

    # 2 variables one for the first letter and one for last
    left = 0
    right = len(phrase) - 1

    # Till left and right letter are not equal, keep checking
    while left != right:

        # Start is letter on left of string, end is on right
        start = phrase[left].upper()
        end = phrase[right].upper()

        # If any time the letters are not equal, it is not a palindrome
        if start != end : return False

        # Keep going towards the middle for both sides
        left += 1
        right -= 1

    # If the loop ends and it is not false, then phrase is palindrome
    return True
```

Hour 3.5

Tuples:

- Tuples are immutable, thus can't be modified
- Created using parenthesis or with just commas.
- Parenthesis are not required
- `my_tuple = (1, 2, 3, 4, 5, 6)`
- Same as `my_tuple = 1, 2, 3, 4, 5, 6`
- Easy way to swap values of 2 variables with tuples: `a, b = b, a`

Enumerate:

- Iterates through any sequence and gives index and a value for it
- Can be used as: `for index, value in enumerate(list)`