# Intro to Programming

3-5-19

## Sign In!

https://tinyurl.com/CADIntroPySp19-3

https://github.com/cadtexas/sp19-intro-to-python

## Last Week's Review

```
1.  (15 < 15) or ("hook" != "em")
2.  string1 = "hello" + "world"
    (24 <= (18 + 6)) and (string1 == "hello world")
3.  (2 < 10) and not (2033 < 5)
```

## Last Week's Review

```
1.  (15 < 15) or ("hook" != "em")-> True
2.  string1 = "hello" + "world"
    (24 <= (18 + 6)) and (string1 == "hello world")-> False
3.  (2 < 10) and not (2033 < 5)-> True
```

https://github.com/cadtexas/sp19-intro-to-python

## Modifying a List

- Elements of a list can be changed using a syntax similar to accessing them
    - Given the list `name_list = [“Sean”, “Claire”]`
    - We can change the first element using the following syntax:
    - `name_list[0] = “Katie”`
    - The contents of `name_list` are now `[“Katie”, “Claire”]`

## Adding List Elements

- The simplest way to add new elements to a list is using the method `append()`
  - `name_list.append("Hannah")`
  - `name_list` is now `["Katie", "Claire", "Hannah"]`
- You can also use the `.insert()` method
  - The syntax is `list.insert(index,value)`
  - Every other value will be shifted to the right
  - `name_list.insert(0, "Sean")`
  - `name_list` is now `["Sean", "Katie", "Claire", "Hannah"]`

## Removing List Elements

- Items can be deleted from a list using the `del` keyword.
  - `del name_list[1]`
  - `name_list` is now `[“Sean”, “Claire”, “Hannah”]`
  - You can no longer access the deleted element
- If you want to use removed values later, use the `.pop()` method
  - `instructor = name_list.pop()`
  - If no index is specified, Python will default to the last element
  - `name_list` is now `[“Sean”, “Claire”]`
  - `print(instructor)` outputs `“Hannah”`

## Removing List Elements (cont’d)

- Items can be removed by value if the index is unknown with the method `remove()`
    - `name_list.remove(“Sean”)`
    - `name_list` is now `[“Claire”]`
    - Note that the `remove()` method will only delete the first occurence of the value you specify

## List Exercises

```
engineering_majors = ["Chemical", "Civil", "Electrical",
"Mechanical"]
print(engineering_majors[2])
engineering_majors.append("Biomedical")
print(engineering_majors)

print(engineering_majors.pop(1))
engineering_majors[0] = "Petroleum"
print(engineering_majors)
```

## List Exercises

```
engineering_majors = ["Chemical", "Civil", "Electrical",
"Mechanical"]
print(engineering_majors[2]) Electrical
engineering_majors.append("Biomedical")
print(engineering_majors)['Chemical', 'Civil',
'Electrical', 'Mechanical', 'Biomedical']
print(engineering_majors.pop(1)) Civil
engineering_majors[0] = "Petroleum"
print(engineering_majors)['Petroleum', 'Electrical',
'Mechanical', 'Biomedical']
```

## if/else

**Structure of if/else:**

```
if {condition}:
    {code that executes
     if condition is true}
else:
    {code that executes
     for all other cases}
```

- Indents matter!
- We can nest them

```
x = 5
if x == 5:
    print(“x is equal to 5”)
else:
    print(“x is not 5”)
    x = x + 1
print(x)

OUTPUT:
```

## if/else

**Structure of if/else:**

```
if {condition}:
    {code that executes
     if condition is true}
else:
    {code that executes
     for all other cases}
```

- Indents matter!
- We can nest them

```
x = 5
if x == 5:
    print("x is equal to 5")
else:
    print("x is not 5")
    x = x + 1
print(x)
```

```
OUTPUT:
x is equal to 5
5
```

## if/else

**Structure of if/else:**

```
if {condition}:
    {code that executes
     if condition is true}
else:
    {code that executes
     for all other cases}
```

- Indents matter!
- We can nest them

```
x = 4
if x == 5:
    print("x is equal to 5")
else:
    print("x is not 5")
    x = x + 1
print(x)
```

**OUTPUT:**

## if/else

**Structure of if/else:**

```
if {condition}:
    {code that executes
     if condition is true}
else:
    {code that executes
     for all other cases}
```

- Indents matter!
- We can nest them

```
x = 4
if x == 5:
    print("x is equal to 5")
else:
    print("x is not 5")
    x = x + 1
print(x)
```

**OUTPUT:**

```
x is not 5
5
```

## elif

**Multiple conditions can be checked using `elif` (else-if) statements)**

```
if {condition1}:
    {code that executes
     if condition1 is true}
elif {condition2}:
     {code that executes
     if condition2 is true}
else:
    {code that executes
     for all other cases}
```

Only one block of code will be executed for each if-elif-else chain.

```
age = 14
if age < 13:
    print("You are a child.")
elif 13 <= age < 18:
    print("You are a teen.")
else:
    print("You are an adult.")
```

```
OUTPUT:
You are a teen.
```

## Boolean Operators Review

- `==` (equals) – True if both are same
- `and` – True only if all are True
- `or` – False only if all are False (True if at least one is True)
- `not` – flips the truth value
- `in` – checks if a value is in a list or string

## if/else Exercises

- Given the following code,

```
if x <= 10:
    x = x * 10
else:
    x = x + 10
print(x)
```

- If `x` has the value of `2` before the if-else block is executed, what will the output be?
- What about when `x` is `42`?

## if/else Exercises

- Given variables `a` and `b`, return True if one of them is 10 or if their sum is 10.
- Listed in the table are some cases for you to test your code.

| a | b | output |
|---|---|---|
| 9 | 10 | True |
| 9 | 9 | False |
| 1 | 9 | True |

## Thanks for coming!

- Next week - more lists, if/else (maybe)!
- Please fill out our feedback form, especially if you'd like to specify which topics we cover next week!
    - https://tinyurl.com/CADIntroPyFeedback3