Python

250H
Python

- Python is a general purpose programming language
Python

- Python is a general purpose programming language
- Python is much closer to pseudocode than most languages
Python

- Python is a general purpose programming language
- Python is much closer to pseudocode than most languages
- Python has a lot of mathematical libraries which allow it to be used for scientific computing, symbolic math, testing proofs, etc.
Python

- Python is a general purpose programming language
- Python is much closer to pseudocode then most languages
- Python has a lot of mathematical libraries which allow it to be used for scientific computing, symbolic math, testing proofs, etc.
- We recommend using Python for any programming projects
  - You are not required to
  - You can use Python or Java
Basic Math

- \(+\), \(-\), \(*\), \(/\)
  - Normal addition, subtraction, multiplication, division
Basic Math

- +, -, *, /
  - Normal addition, subtraction, multiplication, division
- //
  - Floor division
Basic Math

- +, -, *, /
  - Normal addition, subtraction, multiplication, division
- //
  - Floor division
- %
  - Mod or Remainder
Basic Math

- +, -, *, /
  - Normal addition, subtraction, multiplication, division
- //
  - Floor division
- %
  - Mod or Remainder
- **
  - Calculate powers
Math Module

- `math.ceil(x)`
  - Ceiling Function
- `math.comb(n, k)`
  - n choose k
- `math.factorial(x)`
  - Factorial
- `math.floor(x)`
  - Floor
Math Module

- math.ceil(x)
  - Ceiling Function
- math.comb(n, k)
  - n choose k
- math.factorial(x)
  - Factorial
- math.floor(x)
  - Floor
- math.perm(n, k)
  - Permutation
- math.sqrt(x)
  - Square Root
- math.pi
  - $\pi$ constant
- math.e
  - $e$ constant
Strings

- # comment
  - Comment
Strings

- # comment
  - Comment
- “String”
  - ‘String’
    - Quotes tell python what is inside is a string
    - Double quotes and single quotes do the same thing in Python
Strings

- # comment
  - Comment
- “String”
  - ‘String’
  - Quotes tell python what is inside is a string
  - Double quotes and single quotes do the same thing in Python
- print()
  - The print() function produces a more readable output, by omitting the enclosing quotes and by printing escaped and special characters
Lists

- Lists work almost exactly like arrays in Java
Lists

- Lists work almost exactly like arrays in Java
- list = []
  - Creates an empty list called list
Lists

- Lists work almost exactly like arrays in Java
- list = []
  - Creates an empty list called list
- list = [1, 2, 3, 4, 5]
  - list[0] #returns 1
  - list[-1] #returns 5
  - list.append(6) #adds 6 to the list after 5
  - len(list) #returns length of the list
Conditions

- Equals: \( a == b \)
- Not Equals: \( a != b \)
- Less than: \( a < b \)
- Less than or equal to: \( a <= b \)
- Greater than: \( a > b \)
- Greater than or equal to: \( a >= b \)
Conditions

- Equals: \( a == b \)
- Not Equals: \( a \neq b \)
- Less than: \( a < b \)
- Less than or equal to: \( a \leq b \)
- Greater than: \( a > b \)
- Greater than or equal to: \( a \geq b \)

- And: \( \text{and} \)
- Or: \( \text{or} \)
- True: \( \text{true} \)
- False: \( \text{false} \)
- Not: \( \text{not} \)
If Statements

- if condition:
  
  #insert code

- Tabs in Python matter!!!!!!!!
If Statements

- if condition:
  #insert code
- if condition:
  #insert code
- else:
  #insert code

- Tabs in Python matter!!!!!!!
If Statements

- if condition:
  #insert code
- if condition:
  #insert code
else:
  #insert code
- if condition:
  #insert code
elif condition:
  #insert code
else:
  #insert code

- Tabs in Python matter!!!!!!
Loops

- list = [1, 2, 3, 4, 5, 6]
  for x in list:
    #insert code
Loops

- list = [1, 2, 3, 4, 5, 6]
  for x in list:
    #insert code

- for x in range(6):
  #insert code
Loops

- list = [1, 2, 3, 4, 5, 6]
  for x in list:
    #insert code
- for x in range(6):
  #insert code
- Note that range(6) is not the values of 0 to 6, but the values 0 to 5
Loops

- list = [1, 2, 3, 4, 5, 6]
  for x in list:
    #insert code
- for x in range(6):
  #insert code
- Note that range(6) is not the values of 0 to 6, but the values 0 to 5
- range(2, 6) means values from 2 to 6 (but not including 6)
Loops

- `list = [1, 2, 3, 4, 5, 6]`
  
  ```python
  for x in list:
      #insert code
  ```

- `for x in range(6):`
  
  ```python
  #insert code
  ```

- Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5

- `range(2, 6)` means values from 2 to 6 (but not including 6)

- `range(2, 30, 3)` means values from 2 to 30 but will add by 3
  - 2, 5, 8, 11, 14, 17, 20, 23, 26, 29
Loops

- list = [1, 2, 3, 4, 5, 6]
  for x in list:
    #insert code
- for x in range(6):
  #insert code
- Note that range(6) is not the values of 0 to 6, but the values 0 to 5
- range(2, 6) means values from 2 to 6 (but not including 6)
- range(2, 30, 3) means values from 2 to 30 but will add by 3
  - 2, 5, 8, 11, 14, 17, 20, 23, 26, 29
- while condition:
  #insert code
Functions

- def foo():
  #insert code
Functions

- def foo():
  #insert code
- To call the function you only have to use the name of the function with parentheses
  - foo()
Functions

- `def foo()`:
  
  #insert code

- To call the function you only have to use the name of the function with parentheses
  
  - `foo()`

- `def bar(arg1, arg2)`:
  
  #insert code
Functions

- `def foo()`:
  
  #insert code

- To call the function you only have to use the name of the function with parentheses
  - `foo()`

- `def bar(arg1, arg2)`:
  
  #insert code

- To call the function with arguments you just add the arguments inside the parentheses
  - `bar(a,b)`
“Main Method”

- if __name__ == "__main__"
  - Creates a “main method”
“Main Method”

- if __name__ == "__main__"
  - Creates a “main method”
- def foo():
  #insert code
  def bar(arg1):
    #insert code
    if __name__ == "__main__":
      bar(foo())
Helpful Links

- https://docs.python.org/3/
  - Python documentation
- https://www.w3schools.com/python/
  - Examples and Tutorials
- https://www.geeksforgeeks.org/python-programming-language/
  - Examples and Tutorials