**Python**

| **Function** | **Format** | **Example** | **Other Details** |
| --- | --- | --- | --- |
| Basics | | | |
| # | #comment | #this is a comment | A comment, line will be ignored by python. |
| input | input(X) |  | Input prompts user to input information. The X is the prompt. |
| Variable | Variable\_name = Definition | Texas\_capital = Austin | You can overwrite the variable by redefining it but the original value will be lost. |
| Combine | X + Y |  | Combines the x string and the y string. Can’t combine different types of data. EG; Strings + Integers |
| Print Formatting | | | |
| Print | print() | print(house) | Displays text in ( ), strings denoted as ‘ \_ ‘. Commas can be used to combine values even if they aren’t the same type |
| Decimals | Number: .x |  | Limits decimal to x # of decimal |
| Float | Number: f |  | Turns the number into a float number |
| Whitespace | String: x  String: >15 |  | Reserves x amount of whitespace for string  > arrow shows which direction whitespace should be in |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Sep | sep=”x” |  | Separates item in printed string by x |
| End | end=”x” |  | Sets the end of a printed string to be x. Default is to start a new line, this contradicts that |
| Lists | | | |
| List | x=[a,b,c] |  | A list is a variable with multiple values. You can change the value of items in the list with  **list[x] = y** where x is the old value and y is the new one |
| Append | list.append(item) |  | Adds x to list Y |
| Insert | list.insert(position, item) |  | Inserts z to position x in list y |
| Pop | list.pop(index) |  | Removes item in the list at the index specified. Also **RETURNS** the item popped from the list |
| Remove | List.remove(value) |  | Removes the first iteration of the value within the list |
| Sort | list.sort() |  | Sorts list smallest to greatest |
| Sorted | sorted(list) |  | Returns the list sorted from smallest to greatest, but doesn’t actually sort the list itself |
| Max | max(list) |  | Returns Highest value in list |
| Min | min(list) |  | Returns lowest value in list |
| Sum | sum(list) |  | Returns sum of items in list |
|  |  |  |  |
| Information type conversions | | | |
| str | str(**x**) |  | Converts X to a string |
| int | int(x) |  | Converts x to an integer |
| float | float(x) |  | Converts x to a decimal point number |
| Boolean | bool(x) |  | True/False values |
| Lower | **string**.lower(x) |  | Converts string to lower case |
| f | f’**\_\_\_\_\_{x}\_\_\_**’ |  | Returns value in ‘\_\_\_’ and inserts the string inside {x} |
| \n | **\n** |  | New line, empty |
| end | end=”” |  | Added at end of print command to not make a new line |
| Arithmetic | | | |
|  | + |  | Add |
|  | - |  | Subtract |
|  | \* |  | Multiply |
|  | / |  | Divide, results are decimal |
|  | // |  | Divide , result are integers |
|  | % |  | Modulo. Gives remainder after a division |
|  | \*\* |  | Exponent |
|  | += | **X += 1** | Adds y into the integer variable x |
| If | if **X** > **Y:**  **“\_\_\_\_”** |  | X is the condition, > is the operator, and Y is the value met. “ \_\_\_\_” is the conditional data output. Every line of “\_\_” needs to be indented equally |
| Value Comparisons | | | |
|  | > |  | Greater than. Can be used to organize alphabetically too, but letters need to be lowercase. |
|  | >= |  | Greater than or equal to |
|  | == |  | Equal to |
|  | != |  | Not Equal to |
|  | and | if X > Y **and** A < B | Both need to be true |
|  | or | if X > Y **or** A < B | Either need to be true |
|  | not | if **not** ( X > Y ) |  |
| Else | | | |
| Elif | if X > Y:  “\_\_a\_\_”  **elif X > Y:**  **“\_\_b\_\_”**  else:  “\_\_c\_\_” |  | Another possible conditional statement before the else clause. There can be an infinite amount if needed |
| Else | if X > Y:  “\_\_a\_\_”  **else:**  **“\_\_b\_\_”** |  |  |
| Index | | | |
| Len | len(**x)** |  | Returns # of characters in string. X = variable. |
| Index | [x] |  | Prints character x # into the length of the string. Index count begins from 0, not 1. So last character is len - 1. You can count back to front by using - values. -1 is right most, -2 follows, etc. |
|  | [ x : y] |  | Gives characters from x position to y # position |
| In | X in Y |  | Gives true or false value of whether X substring is in Y string |
| Find | string.find(x) |  | Finds the first position where X substring is in input\_string. Returns -1 if the string is not found. |
| Reverse Find | string.rfind(x) |  | Same as find but from the end of the string |
| Loops | | | |
| While & Break | **while True:**  **x = Y**  **if x == Z:**  **break** |  | While = Begins loop, everything in block is loop.  True = Condition  x= condition  Y = Variable that enables loop  Z = Variable that ends loop  Break = Ends loop |
| Continue | **if X >= Y:**  **continue** |  | The if condition can be anything. Continue just restarts the script from the start of the while command |
| For | | | |
| For | **for** variable **in** collection**:** |  | Loop that runs for the amount of times the variable determines within the collection |
| Character | for **character** instring**:** |  | Variable Character is the amount of letters in a string |
| Item | for **item** in list**:** |  | Variable item is amount of items in list |
| I & Range | for **i** in **range(a,b,c)** |  | I = integer  A = starting number  B = ending number  C = size of steps. Can be negative if you want to go in reverse |
| Functions | | | |
| Define | def x(Z):  y | def message(house):  print(house) | Message = function name, used to call function. House = variable. Can have multiple variables with a , between. Variables can be used in the defined function |
| Return | def x(Z):  Y  return A |  | Returns the value a |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Definitions

* Statement: a part of the program which executes something.
  + Ex: **print("Hi!")**
* Block: a group of consecutive statements that are at the same level in the structure of the program
  + Ex: if age > 17:
  + **print("You are of age!")**
  + **age = age + 1**
  + **print("You are now one year older...")**
* Expression: An *expression* is a bit of code that results in a determined data type
  + Ex:
    - 2 + 4 + 3 9 integer int
    - "abc" + "de" "abcde" string str
    - 11 / 2 5.5 floating point number float
    - 2 \* 5 > 9 True Boolean value bool

Debugging

* Hard-coding: in cases where an input is required, hard coding the input saves time
* Visualization Tool: Goes line by line in code <https://pythontutor.com/visualize.html#mode=edit>