Author: Cole Cummins

Python Toolkit

---CONTROL STRUCTURES---

if: else: elif:

Basic logic statements

ternary operator:

Variable assignment and if statement

```
ex. var = “yes!” if True else “no....”
```

while:

While the statement remains true, loop

for:

Modified while loop, iterates and changes each iteration

break:

Breaks out of the current loop immediately, does not continue looping

continue:

Skips over the current loop, continues looping

---BUILT IN FUNCTIONS---

print()

Prints out to the console

```
ex. print(“Hello, World!”)
ex. print(1, 2, 3)
```

input()

Takes user input from the console

```
ex. var = input(“Type a number:”)
    print(var)
```

len()

Returns the length of the given string or list

```
ex. len(“iD Tech!”)
ex. len([1, 2, 3])
```

range()

Returns a range from the first number to the number before the last number

```
ex. range(0, 12)
ex. range(0, len(“hello”))
```

abs()

Returns the absolute value of a number

```
ex. abs(10 - 11)
```

pow()

Returns the first number to the power of the second number, same as ** operator

```
ex. pow(3, 3)
```

max(), min()

Returns the largest and smallest number in a list of numbers respectively

```
ex. max([1, 3, 2])
ex. min([1, 3, 2])
```

sum()

Returns a sum of the given list

```
ex. sum([1, 2, 3])
```

hash()

Returns the hash value for a given item, integers are returned as themselves

```
ex. hash("hello")
```

isinstance()

Returns whether or not the given object is of a certain type or class

```
ex. isinstance("hello", str)
ex. isinstance(42, int)
```

bin(), hex()

Returns a binary or hex representation of the given integer

```
ex. bin(52)
ex. hex(255)
```

---STRINGS---

"Hello World!"

Basic text/string! Can be surrounded by single or double quotes

Escape Characters

\t Tab
\n Newline
\" \' Quotes
\\ Backslash

str + str

String addition

```
ex. "cat" + "dog"
```

*str * x*

String multiplication

*ex. "Wow" * 3*

str[x], str[-x]

Referencing single characters inside strings, putting a negative sign in front of the number refers to the index from the back

```
ex. "Dog"[0]
ex. "cat"[-1]
```

str[:]

Referencing sections of strings not including the number after the colon

```
ex. "Large dog"[:5]
ex. "wow great"[4:6]
```

str.lower() str.upper()

Returns lowercase and uppercase versions of the string respectively

```
ex. "BIG".lower()
ex. "small".upper()
```

str.replace()

Replaces the given characters with other characters

```
ex. "this string without spaces".replace(" ", "")
ex. "a cdef".replace("def", "at")
```

str.split()

Splits the string into a list of strings around the given character

```
ex. "Thats a lot".split()
ex. "abcabcabc".split("a")
```

---LISTS---

[1, 2, 3, 4]

Default list, refers to individual objects in the list the same as strings

Useful List Commands

list.append()
list.insert()
list.remove()
list.pop()
list.count()
list.sort()

[["X", "X", "O"], ["O", "O", "X"], ["X", "X", "O"]]

Two dimensional list! Instead of referring to `list[0]`, refer to `list[0][0]`

```
ex. for list in lists:
```

```
for num in list:
    print(num)
```

---OPERATORS----

`and`

Boolean `and` operator, both statements must be true for `and` to be true

```
ex. True and True
ex. True and False
```

`or`

Boolean `or` operator, either statement can be true for or to be true

```
ex. False or True
```

`not`

Boolean not operator, returns the inverse of the boolean

```
ex. not False
```

`in`

Boolean operator, returns true if object is in the given list

```
ex. 1 in [1, 2, 3]
```

`==, <, <=, >, >=`

Comparison operators with equals, less than, greater than etc.

```
ex. 5 > 12
ex. 5 == 5
```

`+, +=, -, -=, *, *=, /, /=`

Basic math operators, operators with assignments as well

```
ex. Incr += 1
ex. 5 * 5
```

`//, //=`

Floor division and floor division assignment

```
ex. 5//2
ex. 8//10
```

`%`

Mod operator, returns remainder after division, useful for determining even or odd

```
ex. 5 % 2
ex. 8 % 10
```

`&` - Bitwise AND

`|` - Bitwise OR

`<< >>`

Bitwise bit shifting, shifts an integer a certain number of bits right or left

```
ex. 4 << 2
ex. 16 >> 3
```