TAMU ENGR-102 Cheat Sheet

Special Numerical Operators:

`x % y  ->` Modulo: Remainder of $\frac{x}{y}$

`x // y ->` Integer Division: $\lfloor\frac{x}{y}\rfloor$

Conditional (if-elif-else) Statement:

```
if condition1:
    Do this
elif condition2: # Optional
    Do this instead
else: # Optional
    Otherwise, do this
```

Lists:

```
Accessing Elements (x[index]):
x = [[9, 6], 4]
x[0][1] -> 6
x[-2]   -> [9, 6]

Slicing (x[start:end:indexJump]):
x = ['H', 'o', 'w', 'd', 'y']
x[2:]    -> ['w', 'd', 'y']
x[:3]    -> ['H', 'o', 'w']
x[2:4]   -> ['w', 'd']
x[1:-1]  -> ['o', 'w', 'd']
x[-5:-1] -> ['H', 'o', 'w', 'd']
x[::-1]  -> ['y', 'd', 'w', 'o', 'H']

Instance Methods (In-Place):
x.append(y) # Adds y to end of x
del x[i]    # Deletes x[i]
x.remove(y) # Deletes y
x.pop()     # Deletes & returns x[-1]
x.sort()    # Lexicographically sorts x
x.index(y)  # Returns index of y

Static Methods:
len(x) # Number of elements in x
min(x) # Lowest number in x
max(x) # Highest number in x
sum(x) # Summation of numbers in x
```

Dictionaries:

```
Accessing Elements:
d = {'x':9, 'y':"Howdy", 'z':[3, 6, 1]}
d['y']     -> "Howdy"
d['z'][-2] -> 6
d.get('x') -> 9
for key in d:
    Do this for each key in d

for key, value in d.items():
    Do this for each key:value pair in d

Modification:
d[k] = v # Adds key:value pair to d
del d[k] # Deletes key:value pair
```

Functions:

```
def func1(): # Declaration
  '''0 args; implicit return (None)'''
  Do this # Definition

def func2(a, b=25): # b has default val
  '''Positional args; explicit return'''
  Do this
  return finalVal

if __name__ == "__main__":
  func1()      -> None # Function Call
  func2(x)     -> finalVal # a = x, b = 25
  func2(x, y)  -> finalVal # a = x, b = y
```

Try-Except Block:

```
try:
    Try running this exception-prone code
except Exception: # Optional
    Run if specified exception raised
except:
    Run if any exception raised
else: # Optional
    Run only if no exception raised
finally: # Optional
    Always run this
```

Loops:

```
Keywords:
break    # Exists loop
continue # Skips to next iteration

While:
Repeats indented code until condition is False
while condition:
    Do this

For:
Iterates over container (list, tuple, dict)
range(start, end, jump) -> list[int][::jump]
for var in range:
    Do this
```

File IO:

```
Useful File Methods:
str.strip()          -> str # Trim edge ' ', '\n'
str.split(x)         -> list[str] # Split at x
str.join(list[str]) -> str # Joins strs in list

Open/Close File:
  File Modes:
    'r': Read starting at 0th line
    'w': Write starting at 0th line
    'a': Append starting at last line

with open(filename, mode) as file:
    Do this # Auto-closes file after indent

file = open(filename, mode)
file.close()

Read Methods:
file.read()      -> str       # All file data
file.readline()  -> str       # 1 line
file.readlines() -> list[str] # All file data

Write Methods:
file.write(str)             # Writes str
file.writelines(list[str]) # Writes list[str]
```