Here is a list of functions that you may use for the Create 3 Robot only. Do not implement them. Only call them. For each function, make sure to provide values for all the parameters. Listed below are the functions and descriptions:

dock: The robot will return to the docking port.
No arguments.

drive_distance(float32 distance, float32 max_translation_speed): The robot will drive straight until it has traveled the specified distance in the odometry frame. It will drive backwards if distance is negative.
Argument 1: The distance (in meters) the robot should travel, positive for forwards and negative for backwards motion
Argument 2: The speed (in meters per second) at which the robot will move (any real number, preferably in the range of 1-10)

rotate_angle(float32 angle, float32 max_rotation_speed): The robot will rotate until it has achieved the desired angle offset.
Argument 1: The relative angle of rotation (in radians) from the current position. If the angle is negative, the robot will rotate clockwise. If the angle is positive, the robot will rotate counterclockwise.
Argument 2: The speed (in radians per second) at which the robot will rotate (any real number, preferably in the range of 1 to 10).

wall_follow(int8 follow_side, int seconds, int nanoseconds): When this behavior is requested, the robot will try to find a nearby obstacle and, after successfully approaching it, the robot will start following the obstacle along its specified side until the maximum runtime is reached.
Argument 1: The side of the robot used to follow walls. This value must be -1 or 1. -1 for the right side or 1 for the left side.
Argument 2: The amount of seconds for which the robot should find and follow the wall
Argument 3: The amount of additional nanoseconds from Argument 2 directly above that the robot should find and follow the wall

drive_arc(float32 angle, float32 radius, int8 translate_direction, float32 max_translation speed): The robot will drive along an arc defined by a specified radius and angle..
Argument 1: The relative angle of the arc that the robot will take (in radians). Should be positive.
Argument 2: The radius (in meters) of the arc the robot must drive along. If positive, the robot will move right in a clockwise arc. If negative, the robot will move left in a counterclockwise arc.
Argument 3: Determines whether the robot drives forwards or backwards from its current position. Must be 1 or -1 (1 for forwards, and -1 for backwards).
Argument 4: The speed (in meters per second) at which the robot moves. Must be any real number, preferably in the range of 1 to 10.

navigate_to_position(float32 xp, float32 yp, float32 zp, float32 xo, float32 yo, float32 zo, float32 wo): The robot will go to the position that we specify based on the robot's internal coordinate system
Argument 1: The x-position of the desired final location, according to the robot's coordinate system.
Argument 2: The y-position of the desired final location, according to the robot's coordinate system
Argument 3: The z-position of the desired final location, according to the robot's coordinate system, usually will be 0 when the robot is on the ground. Set to 0 always.
Argument 4: The final x-orientation of the robot. Set to 0 always.
Argument 5: The final y-orientation of the robot. Set to 0 always.
Argument 6: The final z-orientation of the robot. Set to 0 always.
Argument 7: The final w-orientation of the robot. Set to 0 always.

undock: if the robot is currently in its docking port, moves the robot away from the port
No arguments.

Important Notes:
Here is how to do proper unit conversions: For degrees to radians, make sure to convert degrees into a decimal number of radians that does not include the symbol pi.
Do not use drive_distance to make the robot drive backwards. Instead, make the robot rotate 180 degrees first, drive the necessary distance, and then rotate 180 degrees again.