Variables and data types

```
my_string = "Hello world !"
my_int = 5
my_float = 20.5
my_complex = 1j
my_list = ["apple", "banana", "cherry"]
my_tuple = ("apple", "banana", "cherry")
my_range = range(6)
my_dict = {"name" : "John", "age" : 36}
my_set = {"apple", "banana", "cherry"}
my_frozenset = frozenset({"apple", "banana", "cherry"})
my_bool = True # or False
my_bytes = b"Hello world !"
my_bytearrey = bytearray(5)
my_memoryview = memoryview(bytes(5))
my_nonetype = None
print(type(my_string)) # output the type of variable 'my_string'
```

Some functions and methods

```
# Example with string methods
my_string = "   Hello world   "
print(my_string[-1]) # Returns last character
print(my_string[1:-1]) # Returns the string without the first and last character
print(my_string.upper())  # Converts the string to uppercase
print(my_string.lower())  # Converts the string to lowercase
print(my_string.strip())  # Removes leading and trailing whitespace
print(my_string.split())  # Splits the string into a list of substrings based on whitespace
words = ["apple", "banana", "cherry"]
print(" ".join(words))  # Joins the elements of the list into a string using space as a separator

# Example with int and float methods
my_int = -5
my_float = 20.75
print(abs(my_int))  # Returns the absolute value of my_int
print(round(my_float, 1))  # Rounds my_float to one decimal place

# Example with list methods
my_list = [1, 2, 3]
my_list.append(4)  # Adds 4 to the end of the list
my_list.extend([5, 6])  # Extends the list by appending elements from another list
print(my_list.pop())  # Removes and returns the last element of the list
my_list.remove(2)  # Removes the first occurrence of 2 from the list
print(my_list.index(3))  # Returns the index of the first occurrence of 3 in the list
```

```
# Example with tuple methods (Note: Tuples are immutable and do not have methods that modify them)
my_tuple = (1, 2, 2, 3)
print(my_tuple.count(2))  # Returns the number of occurrences of 2 in the tuple
print(my_tuple.index(3))  # Returns the index of the first occurrence of 3 in the tuple

# Example with dict methods
my_dict = {"name": "John", "age": 30}
print(my_dict.keys())  # Returns a view of all keys in the dictionary
print(my_dict.values())  # Returns a view of all values in the dictionary
print(my_dict.items())  # Returns a view of all key-value pairs as tuples
print(my_dict.get("name"))  # Returns the value associated with the key "name"
my_dict.update({"city": "New York"})  # Updates the dictionary with key-value pairs from another dictionary
print(my_dict)

# Example with set methods
my_set = {"apple", "banana", "cherry"}
my_set.add("orange")  # Adds "orange" to the set
my_set.remove("banana")  # Removes "banana" from the set
other_set = {"banana", "grape"}
print(my_set.union(other_set))  # Returns a new set containing all elements from both sets
print(my_set.intersection(other_set))  # Returns a new set containing common elements from both sets
print(my_set.difference(other_set))  # Returns a new set containing elements in my_set but not in other_set

# Example with bytes and bytearray methods
my_bytes = b"Hello world"
print(my_bytes.decode())  # Decodes bytes into a string
print(my_bytes.hex())  # Returns a hexadecimal representation of the bytes
hex_string = "48656c6c6f20776f726c64"
print(bytes.fromhex(hex_string))  # Creates a bytes object from a hexadecimal string
```

Casting variables types

```
# String to Integer
str_to_int = int("123")

# Integer to String
int_to_str = str(123)

# String to Float
str_to_float = float("123.45")

# Float to String
float_to_str = str(123.45)

# String to Complex
str_to_complex = complex("1+2j")

# Complex to String
complex_to_str = str(1 + 2j)

# List to Tuple
list_to_tuple = tuple([1, 2, 3])

# Tuple to List
tuple_to_list = list((1, 2, 3))

# List of tuples to Dictionary
list_of_tuples_to_dict = dict([('a', 1), ('b', 2)])

# Set to List
set_to_list = list({1, 2, 3})

# List to Set
list_to_set = set([1, 2, 3])

# Integer to Boolean
int_to_bool = bool(0)

# Bytes to Bytearray
bytes_to_bytearray = bytearray(b"hello")

# Bytearray to Bytes
bytearray_to_bytes = bytes(bytearray(b"hello"))

# No casting needed for NoneType
none_type = None
```

Functions

Functions are independent blocks of code that can be called from anywhere.

```
def function_name(arguments):
    “ “ “
    functions logics
    ” ” ”
    return values
```

Arithmetic operations

```
# Arithmetic Operations
x = 10
y = 5
addition = x + y  # Addition
subtraction = x - y  # Subtraction
multiplication = x * y  # Multiplication
division = x / y  # Division
integer_division = x // y  # Integer Division (floor division)
modulo = x % y  # Modulo (remainder)
exponentiation = x ** y  # Exponentiation

print("Arithmetic Operations:")
print("Addition:", addition)
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
print("Division:", division)
print("Integer Division:", integer_division)
print("Modulo:", modulo)
print("Exponentiation:", exponentiation)

# Increment and Decrement
a = 5
a += 1  # Increment
b = 10
b -= 1  # Decrement

print("\\nIncrement and Decrement:")
print("Increment a:", a)
print("Decrement b:", b)

# Swap
a = 5
b = 10
a, b = b, a  # Swap values of a and b
```

```
print("\nSwap a and b:")
print("a after swap:", a)
print("b after swap:", b)
```

String operations

```
# String Operations
string1 = "Hello"
string2 = "World"
concatenation = string1 + " " + string2  # String Concatenation
repetition = string1 * 3  # String Repetition

print("\nString Operations:")
print("String Concatenation:", concatenation)
print("String Repetition:", repetition)
```