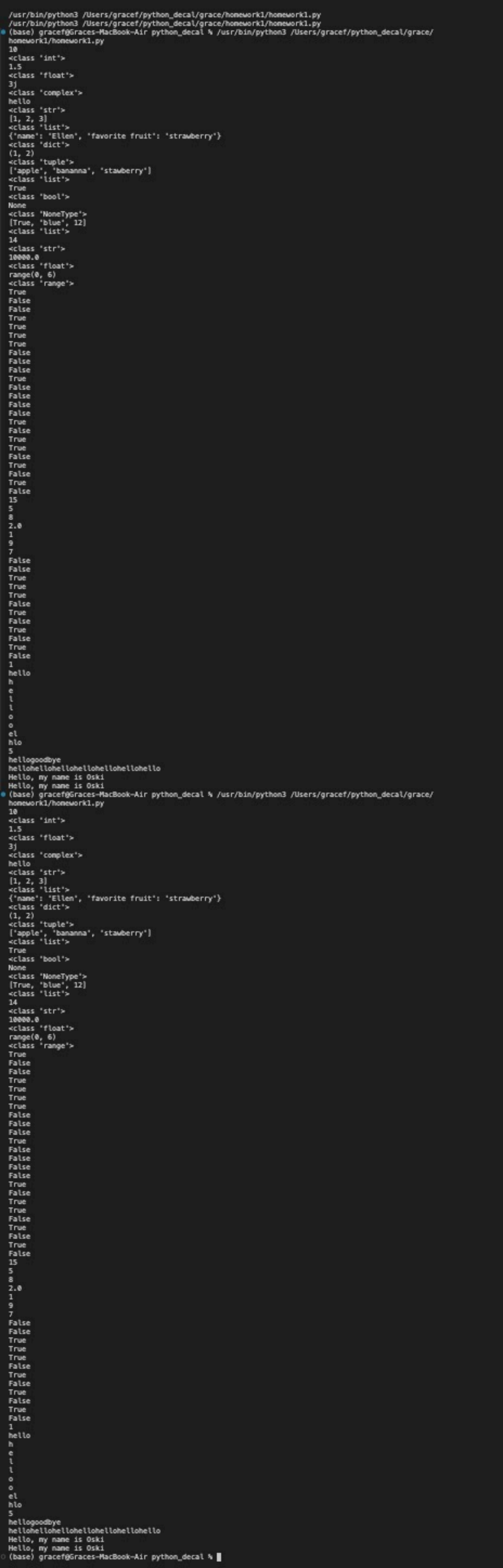

```
/usr/bin/python3 /Users/gracef/python_decal/grace/homework1/homework1.py
/usr/bin/python3 /Users/gracef/python_decal/grace/homework1/homework1.py
(base) gracef@Graces-MacBook-Air python_decal % /usr/bin/python3 /Users/gracef/python_decal/grace/
homework1/homework1.py
10
<class 'int'>
1.5
<class 'float'>
3j
<class 'complex'>
hello
<class 'str'>
[1, 2, 3]
<class 'list'>
{'name': 'Ellen', 'favorite fruit': 'strawberry'}
<class 'dict'>
(1, 2)
<class 'tuple'>
['apple', 'bananna', 'stawberry']
<class 'list'>
True
<class 'bool'>
None
<class 'NoneType'>
[True, 'blue', 12]
<class 'list'>
14
<class 'str'>
10000.0
<class 'float'>
range(0, 6)
<class 'range'>
True
False
False
True
True
True
True
False
False
False
True
False
False
False
False
True
False
True
True
False
True
False
True
False
15
5
8
2.0
1
9
7
False
False
True
True
True
False
True
False
True
False
True
False
1
hello
h
e
l
l
o
o
el
hlo
5
hellogoodbye
hellohellohellohellohellohellohello
Hello, my name is Oski
Hello, my name is Oski
(base) gracef@Graces-MacBook-Air python_decal % /usr/bin/python3 /Users/gracef/python_decal/grace/
homework1/homework1.py
10
<class 'int'>
1.5
<class 'float'>
3j
<class 'complex'>
hello
<class 'str'>
[1, 2, 3]
<class 'list'>
{'name': 'Ellen', 'favorite fruit': 'strawberry'}
<class 'dict'>
(1, 2)
<class 'tuple'>
['apple', 'bananna', 'stawberry']
<class 'list'>
True
<class 'bool'>
None
<class 'NoneType'>
[True, 'blue', 12]
<class 'list'>
14
<class 'str'>
10000.0
<class 'float'>
range(0, 6)
<class 'range'>
True
False
False
True
True
True
True
False
False
False
True
False
False
False
False
True
False
True
True
False
True
False
True
False
15
5
8
2.0
1
9
7
False
False
True
True
True
False
True
False
True
False
True
False
1
hello
h
e
l
l
o
o
el
hlo
5
hellogoodbye
hellohellohellohellohellohellohello
Hello, my name is Oski
Hello, my name is Oski
(base) gracef@Graces-MacBook-Air python_decal %
```

```python
# --- Variables and Data Types ---
a = 10
print(a)
print(type(a)) #a is an integer
b = 1.5
print(b)
print(type(b)) #b is a float
c = 3j
print(c)
print(type(c)) #c is complex
d = "hello"
print(d)
print(type(d)) #d is a string
e = [1,2,3]
print(e)
print(type(e)) #e is a list
f = {"name": "Ellen", "favorite fruit": "strawberry"}
print(f)
print(type(f)) #f is a dict
g = (1,2)
print(g)
print(type(g)) #g is a tuple
h = ["apple", "bananna", "stawberry"]
print(h)
print(type(h)) #h is a list
i = True
print(i)
print(type(i)) #i is a boolean
j = None
print(j)
print(type(j)) #j is No type
k = [True, "blue", 12]
print(k)
print(type(k)) #k is a list
l = str(14)
print(l)
print(type(l)) # l is a string
m = 1e4
print(m)
print(type(m)) #m is a float (in scientific notation)

# questions:
# number of data types: 8
# data types: integer, float, string, complex, dict, tuple, boolean, none
# variables with the same data type: float: b and m; string: d and l; list: e, h and k
# the data type of l is a string and not an integer because the command str() converts the value to a string
# another data type is range
n = range(6)
print(n)
print(type(n)) # n is a range


# ---- Booleans ---
print(10 > 9) #true
print(10 == 9) #false, 10 and 9 are not equivalent
print(10 <= 9) #false, 9 is not less than or equal to 9
print(bool ("abc")) #true, strings are true unless they are empty
print(bool (123)) #true, has content
print(bool(["apple", "cherry", "banana"])) #true, all nonempty lists are true
print(bool(True)) # true, input value
print(bool(False)) #false input value
print(bool(0)) #false, 0 is false
print(bool("")) #false, empty stting
print(bool(" ")) #true, nonempty string
print(bool(())) #false, empty
print(bool([])) #false, empty
print(bool({})) #false, empty
print(bool(True and False)) #false, true and false is false
print(bool(True and True)) #true, both sides are true
print(bool(False and False)) #false, both statements false
print(bool(True or False)) #True, in or statements if there is a true then it is true
print(bool(True or True)) #true, both are true
print(bool(False or False)) #False, both are false
print(bool(not(False))) #true, not false is true
print(bool(not(True))) #false, not true is false

#I notice that the pattern seems to be anything empty or 0 is false but most nonempty structures are true
# 123 suprised me as true because there is no statement and it is not a string
#expression that will return true:
print(bool(4*3-4==8)) #returns true because 4*3 = 12, and 12-4 = 8
#expression that will return false
print(bool("grace"=="lame")) #will return false because i'm not lame and those are not the same string

# ---Arithmitic Operators---
print(10+5) #15, + performs addition
print(10-5) #5, - is subtraction
print(2*4) #8 , * is multiplication
print(6 / 3) # 2.0, / is division
print(5 % 2) # 1, the remainder of 5 divided by 2
print(3 ** 2) #9, 3 to the power of 2
print(15 // 2) #7, how many times 2 can fit into 15 without remainders

# ---Compairson operators---
print(5 == 2) #false, becasue 5 isnt equal to 2
print(10 != 10 ) #false, =! is for not equal, and 10 is equal to itself
print(2<5) #true because 5 is greater than 2
print(12>5) #true because 12 is greater than 5
print(5 <= 6) #true because 6 is greater than 5
print(1 >= 10) #false because one is not greater or equal to 10

# --- assignment operators ---
x = 5
x += 5
x -= 4
x *= 3


# --- Logical operators ---
# 1. the operator and determines if both statements are true or not
print(True and True) # expression that results true
print(False and True) #is and expression that returns false
# 2. the or operator determines if one of the values is true
print(True or False) #true expression
print(False or False) #false expression
# 3. The operator not negates the statement
print(not False) # true expression
print(not True) #false expression

# more questions
# 1. / is float division and // is the amount of times a number can go into another, not with remainders
# 2. % gives the remainder of the two numbers divides, while // gives the result without a remainder
# 3. I would use % to find the remainder, an example is as follows:
print(4 % 3) # remainder 1

# 4. Assignment operators take the variable on the left and use the opperator on the right of the equals sign
# in combination with the value on the right to perform an operation

# --- strings ---
my_string = "hello"
print(my_string) # prints hello
print(my_string[0]) # prints h
print(my_string[1]) # prints e
print(my_string[2]) # prints l
print(my_string[3]) # prints l
print(my_string[4]) # prints o
print(my_string[-1]) # prints o
print(my_string[1:3]) # prints el
print(my_string[0:5:2]) # prints hlo
print(len(my_string)) # prints 5
print( my_string + "goodbye") #prints hello goodbye
print(my_string*7) #it prints 7 times in a row

# 1. slicing is taking a part of a string and splitting it, as shown in my_string[0:5:2]
name = "Oski"
print("Hello, my name is", name) #prints Hello, my name is Oski
name = "Oski"
print(f"Hello, my name is {name}") # prints the same thing, but instead makes it a function so then the vatiable is able to be inserted into the string
#4. the difference is that the first statement was a string in addition to a variable whole the second statement created s function allowing for a variable to be in the string


# --- terminal Commands ---
# cd
# changes directory, use to move from one folder to another
# example: cd python_decal
#ls
# list contents
# lists the folders and file in the folder you are in
# example : ls
# ls -a
# lists the folders/contents in the path directory
# ex ls -a
# mkdir
# creates a new folder
# ex mikdir New_File_name
# cat
# concatenate, print out the entire contents of the file (including comments)
# cat datatypes.py
# pwd
# present working directory - where you are in your files
# ex pwd
# cd ..
# moves up a parent direcotry
# ex cd ..
# cd ~
# return to home directory
# ex cd ~
# cp
# copy - creates a copy of the contents of the file
# ex cp og_file destination_file
# mv
#move and rename files
# ex mv random_screenshot Helpful_name
#rm
# deletes files
# git rm
# clear
# clears the terminal
#ex clear
#grep
# searches for a type of file
#ex grep "python" notes.txt

#questions
#1. 3 other commands :
# a) rmdir, deletes empty direcorties, ex: rimdi directory_name
# b) locate, tells you the direcroty path of a specific file, ex: locate lecture_notes
# c) wget, downloads files from their url, ex wget hwanswers.com (but an actual full url)
#2. ls does not show hidden files like the .git files like ls -a does
#3. a file that is not listed at first and is hidden from view but still exists
#4. 3 other flags
# a) cp --help, gives directions regarding how to use the command
# b) ls -l, gives detailed information about the file
# c) rm -rf, recustivley removes files
```