Variables and Simple Data Types (Chapter 2)

In [11]:
```
# Examples of variables in Python
x=4
student="Alice"
my_favorite_prime_number = 14
print(my_favorite_prime_number)



# TODO declare three more variables below:
newVar="Erin"
classPeriod="programming"
year="2023"
print(newVar)
print(classPeriod)
print(year)
```

```
14
Erin
programming
2023
```

Rules for naming variables:

- Can contain only letters, numbers, and underscores. May not start with a number
- No spaces, use underscores instead
- Avoid built-in functions and keywords like `input`, `print`, `for`, `else`, etc
- Names should be short but descriptive. Avoid single letter names (easy to forget what they mean).

In [14]:
```
# TODO create variables to hold a student's first and last names and current grade. No
studentFirst="Erin"
studentLast="Kincade"
curGrade="3/C"
```

Strings

In [19]:
```
# strings are lists of characters enclosed by either " or '
full_name = "Alice Aardvark"
favorite_team = 'Antartic Antz'

# use " when you need a ' inside the string
```

```
favorite_jam = "Mom's Marmalade"

# several functions can be used to change the capitalization of a string
print(favorite_jam.lower())

print(favorite_jam.upper())

student = 'tiny teddy'
print(student.title())

# TODO create three more strings and modify them using one of the capitalization funct
tinyHello = 'hello'
print(tinyHello.lower())
print(tinyHello.upper())
print(tinyHello.title())
diffNotation="Erin's computer"
print(diffNotation.title())
lastOne = 'goodbye'
print(lastOne.upper())
```

```
mom's marmalade
MOM'S MARMALADE
Tiny Teddy
hello
HELLO
Hello
Erin'S Computer
GOODBYE
```

Strings can be composed from variables. These are called *f-strings* or formatted strings

In [22]:
```
name = 'Alice'
# greeting is an f-string that uses name
greeting = f"Hello, my name is {name}"

friend = 'Bob'
message = f"Hello {friend}, my name is {name}"

# TODO compose a message to Bob telling him your favorite number
favorite_number = 10
message = f"Hey, {friend} my favorite number is {favorite_number}!"
print(message)
```

```
Hey, Bob my favorite number is 10!
```

Adding Whitespace Strings can contain whitespace like tabs and newlines. These non-printing characters are written by using `\` followed by a special character.

In [27]:
```
multiline_message = "This is line 1\nThis is line 2" # what if you add a space around
print(multiline_message)
```

```
This is line 1
This is line 2
```

In [28]:
```
indented_message = "Alice\tAardvark"
print(indented_message)

print("\n\n") # two empty lines
```

```
student_roster = "Alice\tAardvark\nBill\tBobcat"
print(student_roster)
```

```
Alice	Aardvark


Alice	Aardvark
Bill	Bobcat
```

Stripping Whitespace Whitespace can be removed from the beginning and/or end of a string using `strip` functions

In [37]:
```
favorite_class = '   ew200   '
print(f"I love {favorite_class}!")

print(f"I love {favorite_class.lstrip()}!") # remove leading whitespace
print(f"I love {favorite_class.rstrip()}!") # remove trailing whitespace
print(f"I love {favorite_class.strip()}!")  # remove both leading and trailing whitesp

# create a string with leading and trailing white space including \n and \t characters
# then print it using each of the strip functions

whiteSpace = "    hello\tWorld    "
print(f"here it is {whiteSpace}!")
print(f"here it is {whiteSpace.lstrip()}!")
print(f"here it is {whiteSpace.rstrip()}!")
print(f"here it is {whiteSpace.strip()}!")
```

```
I love    ew200   !
I love ew200   !
I love    ew200!
I love ew200!
here it is     hello    World    !
here it is hello        World    !
here it is     hello    World!
here it is hello        World!
```

Numbers

In [38]:
```
# Python supports basic math operations
print(2+2)
print(3-2)
print(8*5)
print(3/2)
```

```
4
1
40
1.5
```

In [39]:
```
# Exponents use **
print(3**2)
```

```
9
```

WARNING! Decimals are approximate- they may not give the exact answer, but this is not usually a problem

In [40]:
```
print(0.2+0.1)
```

```
0.30000000000000004
```

Math Functions

In [49]:
```
from math import sin, cos, pi, floor # add functions from the math package

# math has constants like pi
print(f"pi={pi}")

# compute trig functions (should be 0)
print(f"\nsin(pi)={sin(pi)}")
print(f"cos(pi)={cos(pi)}")


# other useful functions
print(f"\nfloor(2.16)={floor(2.16)}")

# TODO import ceil, what do you think it does?
from math import ceil
print(f"\nceil(3.05)={ceil(3.05)}")

# TODO what is the arctangent of 1.0 in degrees?
from math import atan
print(f"\natan(1.0)={atan(1.0)}")
```

```
pi=3.141592653589793

sin(pi)=1.2246467991473532e-16
cos(pi)=-1.0

floor(2.16)=2

ceil(3.05)=4

atan(1.0)=0.7853981633974483
```

Helpful Hints

In [50]:
```
# use _ to group digits in large numbers
one_billion = 1_000_000_000
print(one_billion)

# multiple assignment can be used to initialize several variables at once
a, b, c = 1, 2, 3
print(c)

# if a variable shouldn't change its value during your program use all capital letters
GRAVITATIONAL_CONSTANT = 9.81
print(GRAVITATIONAL_CONSTANT)
```

```
# Use comments in your code to document your work, include citations, and make it easi
print("Don't forget to use comments in your code!")
```

```
1000000000
3
9.81
Don't forget to use comments in your code!
```

Homework Problems

2-1. Simple Message: Assign a message to a variable and then print that message

In [51]:
```
simpleMessage = 'super simple message'
print(simpleMessage)
```

```
super simple message
```

2-2 Simple Messages: Assign a message to a variable and then print that message. Then change the value of the variable to a new message, and print the new message

In [52]:
```
oldMessage = 'this message is old'
print(oldMessage)
newMessage = 'this message is new!'
print(newMessage)
```

```
this message is old
this message is new!
```

2-3. Personal Message: Use a variable to represent a person's name, and print a message to that person. Your message should be simple, such as, "Hello Eric, would you like to learn some Python today?"

In [54]:
```
roommate = 'Leah'
sentence = 'super stinky'
print(f"My roommate {roommate} is {sentence}.")
```

```
My roommate Leah is super stinky.
```

2-4. Name Cases: Use a variable to represent a person's name and then print that peron's name in lowercase, uppercase, and title case.

In [57]:
```
name = "jOsH"
print(name.lower())
print(name.upper())
print(name.title())
```

```
josh
JOSH
Josh
```

2-5. Famous Quote: Find a quote from a famous person you admire. Print the quote and the name of its author. Your output should look somethink like the following, including quotation marks:

```
    Albert Einstein once said, "A person who never made a mistake
never tried anything new"
```

In [61]:
```
quote = 'Henry Ford once said, "Whether you think you can or you think you can not, yo
print(quote)
```

```
Henry Ford once said, "Whether you think you can or you think you can not, you are ri
ght"
```

2-6. Famous Quote 2: Repeat Exercise 2-5, but this time, represent the famous person's name using a variable called `famous_person`. Then compose your message and represent it with a new variable called `message`. Print your message

In [64]:
```
## create variables for author and quote
famous_person = 'Henry Ford'
message = '"Whether you think you can or think you can not, you are right"'
## print using f
print(f"{famous_person} once said,{message}")
```

```
Henry Ford once said,"Whether you think you can or think you can not, you are right"
```

2-7. Stripping Names: Use a variable to represent a person's name, and include some whitespace characters at the beginning and end of the name. Make sure you use each character combination, "\t" and "\n", at least once. Print the name using each of the three stripping functions, `lstrip()`, `rstrip()`, and `strip()`.

In [68]:
```
nameVar ="    Erin \nKincade\t   "
print(nameVar)
print(nameVar.lstrip())
print(nameVar.rstrip())
print(nameVar.strip())
```

```
    Erin
Kincade
Erin
Kincade
    Erin
Kincade
Erin
Kincade
```

2-8. Number Eight: Write addition, subtraction, multiplication, and division operations that each result in the number 8. Be sure to enclose your operations in `print()` calls to see the results. You should create four lines that look like this:

```
print(5+3)
```

Your output should simply be four lines with the number 8 appearing once on each line.

In [69]:
```
## create four expression that equal 8
print(6+2)
print(4*2)
print(10-2)
print(16/2)
```

```
8
8
8
8.0
```

2-9. Favorite Number: Use a variable to represent your favorite number. Then, using that variable, create a message that reveals your favorite number. Print that message.

In [71]:
```
favNum = '5'
print(f"My favorite number is {favNum}.")
```

```
My favorite number is 5.
```

2-10. Adding Comments: Choose two of the previous problems and at at least one comment to each.

2-11. Zen of Python: Enter `import this` and skim through the additional principles.

In [72]:
```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

In [ ]:
```
```