2. Variables and simple data types

This is a variable! message is the variable

Rules for variables:

- only letters
- no spaces allowed
- avoid pythin keywords
- should be short and descriptive
- be careful with l and 1

In [ ]:

```
message = "Hello python"
print(message)

message = "Hello puta"
print(message)
```

Strings

"this is a string" 'this is also a sting' 'this "is" a string too!'

In [10]:

```
print("this is a string")
print('this is also a sting')
print('this "is" a string too!')
```

```
this is a string
this is also a sting
this "is" a string too!
```

change the case of the string

- method title() performs this action here
- the . (dot) after "name" makes the method "ttitle" act on "name"
- after every method are parantheses, because methods can have additional information to do their work
- in this case there is not such a need and it is empty

In [6]:

```
name = "little one"
print(name.title())
```

```
Little One
```

- other useful methods are "upper" and "lower"
- they can change each letter of the string in uppercase or lowercase

In [7]:

```
name = "other ones"
print(name.upper())
print(name.lower())
```

```
OTHER ONES
other ones
```

combining or *concatenating* string you often want to combine strings

In [12]:

```
first_name =  "ugur"
last_name = "tigu"
full_name = first_name + " " + last_name
print(full_name)
```

```
ugur tigu
```

In [13]:

```
first_name =  "ugur"
last_name = "tigu"
full_name = first_name + " " + last_name
print("Hello, " + full_name.title() + "!")
```

```
Hello, Ugur Tigu!
```

In [14]:

```
first_name =  "ugur"
last_name = "tigu"
full_name = first_name + " " + last_name
message = "Hello, " + full_name.title() + "!"
print(message)
```

```
Hello, Ugur Tigu!
```

adding whitespace to strings with tabs or newlines

- to add "tab" add "\t"
- to add "new line" add "\n"
- you can also combine both

In [15]:

```
print("python")
print("\tpython")
```

```
python
        python
```

In [19]:

```
print("languages:\nenglish\ngerman\nturkish")
```

```
languages:
english
german
turkish
```

In [23]:

```
print("languages:\n\tenglish\n\tgerman\n\tturkish")
```

```
languages:
	english
	german
	turkish
```

stripping whitespace

- 'python' and 'python ' look pretty much the same
- To ensure that no whitespace exists at the right end of a string, use the *rstrip()* method

we store the fav_lang in a new variable so we can print it directly out without doing the operation in the print method)

btw! you can do the same operation with left also! the method is called *lstrip()* method and works pretty much the same way!!

In [90]:

```
fav_lang = 'python '
print(fav_lang+'a')
print(fav_lang.rstrip()+'a')
fav_lang = fav_lang.rstrip()
print(fav_lang+'a')
```

```
python a
pythona
pythona
```

Tasks

- 2-3. Personal Message: Store a person’s name in a variable, and print a message to that person . Your message should be simple, such as, “Hello Eric, would you like to learn some Python today?”
- 2-4. Name Cases: Store a person’s name in a variable, and then print that person’s name in lowercase, uppercase, and titlecase .
- 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 something like the following, including the quotation marks: Albert Einstein once said, “A person who never made a mistake never tried anything new.”
- 2-6. Famous Quote 2: Repeat Exercise 2-5, but this time store the famous per- son’s name in a variable called famous_person . Then compose your message and store it in a new variable called message . Print your message .
- 2-7. Stripping Names: Store 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 once, so the whitespace around the name is displayed . Then print the name using each of the three stripping functions, lstrip(), rstrip(), and strip().

In [93]:

```
#2.3
name = "ugur"
print("Hello " + name.title() + " would you like to learn some Python today?")
```

```
Hello Ugur would you like to learn some Python today?
```

In [96]:

```
#2.4
name = "ugur"
print(name.lower())
print(name.upper())
print(name.title())
```

```
ugur
UGUR
Ugur
```

In [108]:

```
#2.5
quote = "As is a tale, so is life: not how long it is, but how good it is, is wh
at matters."
name = 'seneca'
print(name.title() + " once said: " + '"' + quote + '"' )
```

```
Seneca once said: "As is a tale, so is life: not how long it is, but
how good it is, is what matters."
```

In [109]:

```
#2.6
quote = "As is a tale, so is life: not how long it is, but how good it is, is wh
at matters."
name = 'seneca'
message = name.title() + " once said: " + '"' + quote + '"'
print(message)
```

```
Seneca once said: "As is a tale, so is life: not how long it is, but
how good it is, is what matters."
```

In [116]:

```
#2.7
name = " ugur "
print("a"+name+"a")
print("a"+name.lstrip()+"a")
print("a"+name.rstrip()+"a")
print("a"+name.strip()+"a")
```

```
a ugur a
augur a
a ugura
augura
```

Integers

you can add (+) substract (-) multiply (*) and devide (/) integers

In [40]:

```
2+3
```

Out[40]:

```
5
```

In [41]:

```
2-3
```

Out[41]:

```
-1
```

In [42]:

```
2*3
```

Out[42]:

```
6
```

In [43]:

```
2/3
```

Out[43]:

```
0.6666666666666666
```

for exponents use **

In [47]:

```
2**3
```

Out[47]:

```
8
```

Floats

any number with a decimal is a float

avoiding type errors with the *str()* function

when use integers and strings together you have to tell that you want the integer as a string

In [50]:

```
age = 30
message = "Happy " + str(age) + "rd Birthday!"
print(message)
```

```
Happy 30rd Birthday!
```

Comments

the hash mark (#) you can use as comments

In [52]:

```
#say hallo
print("this and ignore the thing above")
```

```
this and ignore the thing above
```

In [53]:

```
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 i
t.
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!
```

3. Lists

- Lists allow you to store sets of information in one place
- A list is a collection of items in a particular order
- You can make a list that includes the letters of the alphabet
- the digits from 0–9
- or the names of all the people in your family
- it’s a good idea to make the name of your list plural, such as letters, digits, or names
- In Python, square brackets ([ ]) indicate a list
- and individual elements in the list are separated by commas

In [54]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes)
```

```
['trek', 'connan', 'redline']
```

acceccing elements in a list

- lists are ordered collections
  1. you have to tell the position (or the index) of the desired item
  2. to access an element in a list, write the name of the list followed by the index of the item in square brackets

In [55]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes[0])
```

```
trek
```

In [57]:

```
#using the string method title() to format the element to have it capitalized
bikes = ['trek', 'connan', 'redline']
print(bikes[0].title())
```

```
Trek
```

index positions start at 0 not 1!!

- the first item is always 0 and not 1
- if you want to have the 4th item you have to say 3 as index

In [58]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes[2])
```

```
redline
```

for the last element in a list just ask for the "-1" item in the list!

In [59]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes[-1])
```

```
redline
```

you can go backwards by saying "-2" and "-3" ...

In [60]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes[-2])
```

```
connan
```

you can use individual values from a list just like any other variable

In [62]:

```
bikes = ['trek', 'connan', 'redline']
message = "my first bike was a " + bikes[0].title() + "."
print(message)
```

```
my first bike was a Trek.
```

changing elements from a list

1. use the name of the list
2. followed by the index of the element you want to change
3. and provide the new value

In [63]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes)
```

```
['trek', 'connan', 'redline']
```

In [65]:

```
bikes = ['trek', 'connan', 'redline']
bikes[0] = 'ducati'
print(bikes)
```

```
['ducati', 'connan', 'redline']
```

adding elements to a list

1. append the element to the list by using the append method
2. the element will be added to the end of the list

In [66]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes)

bikes.append('ducati')
print(bikes)
```

```
['trek', 'connan', 'redline']
['trek', 'connan', 'redline', 'ducati']
```

- you can start with a empty list
- and add something into the list with the append() method

this is a common way to build lists, because you don't know what the data will be in the list

In [67]:

```
bikes = []

bikes.append('bike_a')
bikes.append('bike_b')
bikes.append('bike_c')

print(bikes)
```

```
['bike_a', 'bike_b', 'bike_c']
```

inserting elements to the list

- with the insert() method
- you can specify the index of the new element
- parameters (index number, element string)

In [69]:

```
bikes = ['trek', 'connan', 'redline']

bikes.insert(0, 'ducati')
print(bikes)
```

```
['ducati', 'trek', 'connan', 'redline']
```

In [70]:

```
bikes = ['trek', 'connan', 'redline']

bikes.insert(1, 'ducati')
print(bikes)
```

```
['trek', 'ducati', 'connan', 'redline']
```

removing elements from a list with the *del statement*

- with the "del" statement and the name of the list and the index of the element
- you can remove an element according to its position in the list
- or according to its value

In [71]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes)

del bikes[0]
print(bikes)
```

```
['trek', 'connan', 'redline']
['connan', 'redline']
```

removing elements from a list with the *pop() method*

- you want to use the removed element of a list
- the *pop() method* removes the last item in a list
- but it lets you work with that item after removing it
- pop from a stack (from the top) here from the end of the list

In [73]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes)

popped_bikes = bikes.pop()
print(bikes)
print(popped_bikes)
```

```
['trek', 'connan', 'redline']
['trek', 'connan']
redline
```

In [75]:

```
bikes = ['trek', 'connan', 'redline']
last_owned_bike = bikes.pop()
print("The last owned bike is a " + last_owned_bike.title() + ".")
```

```
The last owned bike is a Redline.
```

- after using the *pop() method* the element is no longer in the list!
- if you want to delete the item and not want to use it any more use **del statement**
- if you want to delete the item and want to use it again use **pop() method**

removing an item by value

- if you only know the value (not the position)
- use the **remove() method**

In [76]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes)

bikes.remove('connan')
print(bikes)
```

```
['trek', 'connan', 'redline']
['trek', 'redline']
```

you can also work with removed item removed via the **remove() method**

In [81]:

```
bikes = ['trek', 'connan', 'redline']
print(bikes)

too_expensive = 'connan'
bikes.remove(too_expensive)
print(bikes)
print("\A " + too_expensive.title() + " is too expensive for me.")
```

```
['trek', 'connan', 'redline']
['trek', 'redline']

A Connan is too expensive for me.
```

organizing lists

- you can't always control the order of
- sometimes you want the original order
- sometimes you want to sort them **alphabetically**
- the **sort() method** changes the order of the list *permanently*

In [82]:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort()
print(cars)
```

```
['audi', 'bmw', 'subaru', 'toyota']
```

- sort the list reverse alphabetically by passing the argument **reverse = True** to the **sort() method**
- the order is permanently changed here, too!

In [84]:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort(reverse=True)
print(cars)
```

```
['toyota', 'subaru', 'bmw', 'audi']
```

- **sorting a list temporarily with the *sorted() function***
- the **sorted() function** can also accep **reverse = True**

In [85]:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print("Here is the original list:")
print(cars)

print("\nHere is the sorted list:")
print(sorted(cars))

print("\nHere is the original list again:")
print(cars)
```

```
Here is the original list:
['bmw', 'audi', 'toyota', 'subaru']

Here is the sorted list:
['audi', 'bmw', 'subaru', 'toyota']

Here is the original list again:
['bmw', 'audi', 'toyota', 'subaru']
```

- **printing in reverse order (chronologically - not alphabetically)**
- with the **reverse() method**
- you can "undo" by applying the same function a second time

In [86]:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)

cars.reverse()
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']
['subaru', 'toyota', 'audi', 'bmw']
```

In [87]:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)

cars.reverse()
print(cars)

cars.reverse()
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']
['subaru', 'toyota', 'audi', 'bmw']
['bmw', 'audi', 'toyota', 'subaru']
```

- **finding the lenght of a list**
- with the **len() function**

In [117]:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
len(cars)
```

Out[117]:

```
4
```

Tasks

- 3-1. Names: Store the names of a few of your friends in a list called names . Print each person's name by accessing each element in the list, one at a time.
- 3-2. Greetings: Start with the list you used in Exercise 3-1, but instead of just printing each person's name, print a message to them. The text of each message should be the same, but each message should be personalized with the person's name.
- 3-3. Your Own List: Think of your favorite mode of transportation, such as a motorcycle or a car, and make a list that stores several examples . Use your list to print a series of statements about these items, such as "I would like to own a Honda motorcycle."
- 3-4. Guest List: If you could invite anyone, living or deceased, to dinner, who would you invite? Make a list that includes at least three people you'd like to invite to dinner . Then use your list to print a message to each person, inviting them to dinner.
- 3-5. Changing Guest List: You just heard that one of your guests can't make the dinner, so you need to send out a new set of invitations . You'll have to think of someone else to invite.
  - Start with your program from Exercise 3-4 . Add a print statement at the end of your program stating the name of the guest who can't make it.
  - Modify your list, replacing the name of the guest who can't make it with the name of the new person you are inviting.
  - Print a second set of invitation messages, one for each person who is still in your list.
- 3-6. More Guests: You just found a bigger dinner table, so now more space is available. Think of three more guests to invite to dinner.
  - Start with your program from Exercise 3-4 or Exercise 3-5 . Add a print statement to the end of your program informing people that you found a bigger dinner table.
  - Use insert() to add one new guest to the beginning of your list.
  - Use insert() to add one new guest to the middle of your list.
  - Use append() to add one new guest to the end of your list.
  - Print a new set of invitation messages, one for each person in your list.

In [119]:

```
#3.1
friends = ['jeff', 'ricardo', 'nina', 'sven']
print(friends[0])
print(friends[1])
print(friends[2])
print(friends[3])
```

```
jeff
ricardo
nina
sven
```

In [121]:

```
#3.2
friends = ['jeff', 'ricardo', 'nina', 'sven']
print("Hello my friend " + friends[0].title() + "!")
print("Hello my friend " + friends[1].title() + "!")
print("Hello my friend " + friends[2].title() + "!")
print("Hello my friend " + friends[3].title() + "!")
```

```
Hello my friend Jeff!
Hello my friend Ricardo!
Hello my friend Nina!
Hello my friend Sven!
```

In [123]:

```
#3.3
cars = ['audi', 'benz', 'bentley', 'smart']
print("I would like to drive a " + cars[1].title() + "!")
```

```
I would like to drive a Benz!
```

In [126]:

```
#3.4
persons = ['hitler', 'saddam', 'hitlers wife']
print("I would like to invite " + persons[0].title() + " to dinner!")
print("I would like to invite " + persons[1].title() + " to dinner!")
print("I would like to invite " + persons[2].title() + " to dinner!")
```

```
I would like to invite Hitler to dinner!
I would like to invite Saddam to dinner!
I would like to invite Hitlers Wife to dinner!
```

In [146]:

```
#3.5
persons = ['hitler', 'saddam', 'hitlers wife']
not_coming = 'hitler'
persons.remove(not_coming)
print(persons)
print("\nMy friend " + not_coming.title() + " is not coming tonight!")
print(persons)
persons.insert(1, 'merkel')
print(persons)
print("I would like to invite " + persons[0].title() + " to dinner!")
print("I would like to invite " + persons[1].title() + " to dinner!")
print("I would like to invite " + persons[2].title() + " to dinner!")
```

```
['saddam', 'hitlers wife']

My friend Hitler is not coming tonight!
['saddam', 'hitlers wife']
['saddam', 'merkel', 'hitlers wife']
I would like to invite Saddam to dinner!
I would like to invite Merkel to dinner!
I would like to invite Hitlers Wife to dinner!
```

In [155]:

```python
#3.6
persons = ['hitler', 'saddam', 'hitlers wife']
print("Hey " + persons[0].title() + ", I found a bigger table!")
print("Hey " + persons[1].title() + ", I found a bigger table!")
print("Hey " + persons[2].title() + ", I found a bigger table!")
persons.insert(0, 'merkel')
persons.insert(4, 'atatürk')
print(persons)

print("\nHey " + persons[0].title() + ", I found a bigger table!")
print("Hey " + persons[1].title() + ", I found a bigger table!")
print("Hey " + persons[2].title() + ", I found a bigger table!")
print("Hey " + persons[3].title() + ", I found a bigger table!")
print("Hey " + persons[4].title() + ", I found a bigger table!")
```

```
Hey Hitler, I found a bigger table!
Hey Saddam, I found a bigger table!
Hey Hitlers Wife, I found a bigger table!
['merkel', 'hitler', 'saddam', 'hitlers wife', 'atatürk']

Hey Merkel, I found a bigger table!
Hey Hitler, I found a bigger table!
Hey Saddam, I found a bigger table!
Hey Hitlers Wife, I found a bigger table!
Hey Atatürk, I found a bigger table!
```

Tasks

- 3-7. Shrinking Guest List: You just found out that your new dinner table won’t arrive in time for the dinner, and you have space for only two guests .
  - Start with your program from Exercise 3-6 . Add a new line that prints a message saying that you can invite only two people for dinner .
  - Use pop() to remove guests from your list one at a time until only two names remain in your list . Each time you pop a name from your list, print a message to that person letting them know you’re sorry you can’t invite them to dinner .
  - Print a message to each of the two people still on your list, letting them know they’re still invited .
  - Use del to remove the last two names from your list, so you have an empty list . Print your list to make sure you actually have an empty list at the end of your program .

In [177]:

```
#3.7
persons = ['hitler', 'saddam', 'hitlers wife']
print("\nHey, I can invite just 2 people!")

person_goes_out = persons.pop()
print("The person who goes out is: " + person_goes_out.title() + "!")
person_goes_out = persons.pop()
print("The person who goes out is: " + person_goes_out.title() + "!")
last_name = persons[-1]
print("\nMy friend " + last_name.title() + " is invited!")
print(persons)
del persons[-1]
print(persons)
```

```
Hey, I can invite just 2 people!
The person who goes out is: Hitlers Wife!
The person who goes out is: Saddam!

My friend Hitler is invited!
['hitler']
[]
```

Tasks

- 3-8. Seeing the World: Think of at least five places in the world you'd like to visit.
    - Store the locations in a list . Make sure the list is not in alphabetical order.
    - Print your list in its original order . Don't worry about printing the list neatly, just print it as a raw Python list.
    - Use sorted() to print your list in alphabetical order without modifying the actual list. Show that your list is still in its original order by printing it.
    - Use sorted() to print your list in reverse alphabetical order without changing the order of the original list. Show that your list is still in its original order by printing it again.
    - Use reverse() to change the order of your list . Print the list to show that its order has changed.
    - Use reverse() to change the order of your list again . Print the list to show it's back to its original order.
    - Use sort() to change your list so it's stored in alphabetical order. Print the list to show that its order has been changed.
    - Use sort() to change your list so it's stored in reverse alphabetical order . Print the list to show that its order has changed.
- 3-9. Dinner Guests: Working with one of the programs from Exercises 3-4 through 3-7 (page 46), use len() to print a message indicating the number of people you are inviting to dinner.
- 3-10. Every Function: Think of something you could store in a list. For example, you could make a list of mountains, rivers, countries, cities, languages, or anything else you'd like. Write a program that creates a list containing these items and then uses each function introduced in this chapter at least once.

In [178]:

```
#3.8
locations = ['rio', 'kabul', 'tokio', 'new york', 'san fransisco']
print(locations)
```

```
['rio', 'kabul', 'tokio', 'new york', 'san fransisco']
```

In [179]:

```
print(sorted(locations))
```

```
['kabul', 'new york', 'rio', 'san fransisco', 'tokio']
```

In [180]:

```
print(sorted(locations, reverse=True))
```

```
['tokio', 'san fransisco', 'rio', 'new york', 'kabul']
```

In [181]:

```
print(sorted(locations))
```

```
['kabul', 'new york', 'rio', 'san fransisco', 'tokio']
```

In [186]:

```
print(locations)
locations.reverse()
print(locations)
locations.reverse()
print(locations)
```

```
['san fransisco', 'new york', 'tokio', 'kabul', 'rio']
['rio', 'kabul', 'tokio', 'new york', 'san fransisco']
['san fransisco', 'new york', 'tokio', 'kabul', 'rio']
```

In [188]:

```
print(locations)
locations.sort()
print(locations)
locations.sort(reverse=True)
print(locations)
```

```
['kabul', 'new york', 'rio', 'san fransisco', 'tokio']
['kabul', 'new york', 'rio', 'san fransisco', 'tokio']
['tokio', 'san fransisco', 'rio', 'new york', 'kabul']
```

In [189]:

```
#3.9
persons = ['hitler', 'saddam', 'hitlers wife']
num_persons = len(persons)
print("We have " + str(num_persons) + " people!")
```

```
We have 3 people!
```

In [ ]:

```
#3.10
```