1. Python Data Type

Python has the following data types built-in by default, in these categories:

- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview

In [2]:

```
x = "Hello World" #str
print(x)

x = 20 #int
print(x)

x = 20.5 #float
print(x)

x = ["apple", "banana", "cherry"] #list
print(x)

x = ("apple", "banana", "cherry") #tuple
print(x)

x = range(6) #range
print(x)

x = {"name" : "John", "age" : 36} #dict
print(x)

x = True #bool
print(x)
```

```
Hello World
20
20.5
['apple', 'banana', 'cherry']
('apple', 'banana', 'cherry')
range(0, 6)
{'name': 'John', 'age': 36}
True
```

2. Python Array

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

2.1. List

In [3]:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
print(thislist[1]) # access items
print(thislist[-1]) # negative indexing

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
```

```
print(thislist[2:5]) # range of indexes
print(thislist[-4:-1])
```

```
['apple', 'banana', 'cherry']
banana
cherry
['cherry', 'orange', 'kiwi']
['orange', 'kiwi', 'melon']
```

In [4]:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant" # change item value
print(thislist)
```

```
['apple', 'blackcurrant', 'cherry']
```

In [5]:

```
thislist = ["apple", "banana", "cherry"]
for x in thislist: # loop through a list
  print(x)
```

```
apple
banana
cherry
```

In [6]:

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist: # check if item exist
  print("Yes, 'apple' is in the fruits list")
```

```
Yes, 'apple' is in the fruits list
```

In [7]:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist)) # list length
```

```
3
```

In [8]:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange") # add items
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange") # add item at the specified index
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.remove("banana") # remove item
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.pop() # removes the specified index, (or the last item if index is not specified)
print(thislist)

thislist = ["apple", "banana", "cherry"]
del thislist[0] # removes the specified index
print(thislist)

thislist = ["apple", "banana", "cherry"]
del thislist # delete the list completely

thislist = ["apple", "banana", "cherry"]
thislist.clear() # empties the list
```

```
thislist.clear() # empties the list
print(thislist)
```

```
['apple', 'banana', 'cherry', 'orange']
['apple', 'orange', 'banana', 'cherry']
['apple', 'cherry']
['apple', 'banana']
['banana', 'cherry']
[]
```

In [9]:

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy() # copy of a list
print(mylist)

thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
```

```
['apple', 'banana', 'cherry']
['apple', 'banana', 'cherry']
```

In [10]:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list3 = list1 + list2 # join two lists
print(list3)

list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

for x in list2:
  list1.append(x)

print(list1)

list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list1.extend(list2)
print(list1)
```

```
['a', 'b', 'c', 1, 2, 3]
['a', 'b', 'c', 1, 2, 3]
['a', 'b', 'c', 1, 2, 3]
```

2.2. Tuple

In [11]:

```
# You can convert the tuple into a list, change the list, and convert the list back into a tuple.

x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

```
('apple', 'kiwi', 'cherry')
```

In [12]:

```
# To create a tuple with only one item, you have add a comma after the item,
# unless Python will not recognize the variable as a tuple.
```

```
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

```
<class 'tuple'>
<class 'str'>
```

In [13]:

```
# Tuples are unchangeable, so you cannot remove items from it, but you can delete the tuple comple
tely
thistuple = ("apple", "banana", "cherry")
del thistuple
```

2.3. Set

In [14]:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

```
{'cherry', 'banana', 'apple'}
```

2.4 Dictionary

In [15]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

In [16]:

```
x = thisdict["model"] # accessing items
print(x)

x = thisdict.get("model")
print(x)
```

```
Mustang
Mustang
```

In [17]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2019 # change values
```

In [18]:

```
for x in thisdict:
  print(x)

print("\n")
```

```
for x in thisdict:
  print(thisdict[x])

print("\n")

for x in thisdict.values():
  print(x)

print("\n")

for x, y in thisdict.items():
  print(x, y)
```

```
brand
model
year


Ford
Mustang
2019


Ford
Mustang
2019


brand Ford
model Mustang
year 2019
```

In [19]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem() # removes the last inserted item
print(thisdict)

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

```
{'brand': 'Ford', 'year': 1964}
{'brand': 'Ford', 'model': 'Mustang'}
{'brand': 'Ford', 'year': 1964}
```

3. Python If ... Else

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b

- Greater than or equal to: a >= b

In [20]:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
b is greater than a
```

In [21]:

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a and b are equal
```

In [22]:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

4. Python Loops

4.1. For Loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

In [23]:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
apple
banana
```

In [24]:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

```
apple
cherry
```

In [25]:

```
for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

In [26]:

```
for x in range(6):
  print(x)
```

```
0
1
2
3
4
5
```

In [27]:

```
for x in range(2, 6):
  print(x)
```

```
2
3
4
5
```

In [28]:

```
for x in range(2, 30, 3):
  print(x)
```

```
2
5
8
11
14
17
20
23
26
29
```

In [29]:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
3
4
5
Finally finished!
```

In [30]:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
```

```
  for y in fruits:
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

4.2. While Loop

In [31]:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

In [32]:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
1
2
3
```

In [33]:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
1
2
4
5
6
```

In [34]:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

```
1
2
3
```

```
4
5
i is no longer less than 6
```

5. Python Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

In [35]:

```
def my_function():
  print("Hello from a function")

my_function()
```

```
Hello from a function
```

In [36]:

```
# Parameters

def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

```
Emil Refsnes
Tobias Refsnes
Linus Refsnes
```

In [37]:

```
# Default Parameter Value

def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

```
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

In [38]:

```
# Passing a List as a Parameter

def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

```
apple
banana
cherry
```

In [39]:

```
# Return Values

def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

```
15
25
45
```

In [40]:

```
# Keyword Arguments

def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")

def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

```
The youngest child is Linus
The youngest child is Linus
```

6. Python Command Input

In [41]:

```
x = input("Type your name: ")
print("Name: ", x)
```

```
Type your name: Nizar
Name:  Nizar
```