Python CheetSheet

Naga Ganesh B J
bjnaga@gmail.com

A Quick Guide to Python

1 Collection

List	Touple	Set	Dictionary
mutable	imutable	mutable and unique	mutable key not mutable
append()		add()	update()
pop()		pop()-removes random item	pop()specified item removed
sort()			sorted() sorts key

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, unchangeable*, and unindexed. No duplicate members.
Dictionary is a collection which is ordered** and changeable. No duplicate members.

2 Comments

Single - line comment
"""Some Text""" - Multi-line comment

3 Assignment

x, y, z = "Orange", "Banana", "Cherry"
x, y, z = ["apple", "banana", "cherry"] both statements are same
x = y = z = "Orange"
print(x, y, z) = print(x+y+z)

4 Built-in Data type

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
None Type:	NoneType

bool - True/False x=10j - is an example of complex number initialization
a="""multiline string including newline""" or "'x"'

5 String

b = "Hello, World!"
print(b[2:5]) - llo
print(b[:5]) - Hello
print(b[2:]) - llo, World!
print(b[-5:-2]) - orl
print(b.upper() - HELLO, WORLD!
b.lower()
b.strip() - remove whitespace from beginning and end.
b.replace('H','E')
b.split(',')
eg:
age = 36
txt = "My name is John, and I am "
print(txt.format(age))

quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay 2 dollars for 0 pieces of item 1."
print(myorder.format(quantity, itemno, price))
I want to pay 49.95 dollars for 3 pieces of item 567 txt = "We are the so-called \"Vikings\" from the north."

\' Single Quote

\\ Backslash

\n NewLine

\ Carriage Return

\b Backspace

\f Form feed

\ooo Octal value

\xhh Hex value

String Functions

- Method Description
- capitalize() Converts the first character to upper case
- casefold() Converts string into lower case
- center() Returns a centered string
- count() Returns the number of times a specified value occurs in a string
- encode() Returns an encoded version of the string
- endswith() Returns true if the string ends with the specified value
- expandtabs() Sets the tab size of the string
- find() Searches the string for a specified value and returns the position of where it was found
- format() Formats specified values in a string
- format_map() Formats specified values in a string
- index() Searches the string for a specified value and returns the position of where it was found
- isalnum() Returns True if all characters in the string are alphanumeric
- isalpha() Returns True if all characters in the string are in the alphabet
- isdecimal() Returns True if all characters in the string are decimals
- isdigit() Returns True if all characters in the string are digits
- isidentifier() Returns True if the string is an identifier
- islower() Returns True if all characters in the string are lower case

- isnumeric() Returns True if all characters in the string are numeric
- isprintable() Returns True if all characters in the string are printable
- isspace() Returns True if all characters in the string are whitespaces
- istitle() Returns True if the string follows the rules of a title
- isupper() Returns True if all characters in the string are upper case
- join() Joins the elements of an iterable to the end of the string
- ljust() Returns a left justified version of the string
- lower() Converts a string into lower case
- lstrip() Returns a left trim version of the string
- maketrans() Returns a translation table to be used in translations
- partition() Returns a tuple where the string is parted into three parts
- replace() Returns a string where a specified value is replaced with a specified value
- rfind() Searches the string for a specified value and returns the last position of where it was found
- rindex() Searches the string for a specified value and returns the last position of where it was found
- rjust() Returns a right justified version of the string
- rpartition() Returns a tuple where the string is parted into three parts
- rsplit() Splits the string at the specified separator, and returns a list
- rstrip() Returns a right trim version of the string
- split() Splits the string at the specified separator, and returns a list
- splitlines() Splits the string at line breaks and returns a list
- startswith() Returns true if the string starts with the specified value
- strip() Returns a trimmed version of the string
- swapcase() Swaps cases, lower case becomes upper case and vice versa
- title() Converts the first character of each word to upper case
- translate() Returns a translated string
- upper() Converts a string into upper case
- zfill() Fills the string with a specified number of 0 values at the beginning

6 Variable Name

Camel Case
myVariable

Pascal Case
MyVariableName

Snake Case my_variable_name

7 Function Definition

Listing 1: Insert code directly in your document

```
def myfunc():
        global x
        x="fantastic"
myfunc()
```

8 While

Listing 2: Insert code directly in your document

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

9 For

Listing 3: Insert code directly in your document

```
for x in range(6):
  if x == 3: break
  pass
else:
  print("Finally finished!")
```

10 Function Arguments

Listing 4: Insert code directly in your document

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

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

Listing 5: Insert code directly in your document

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

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

11 Lambda

A lambda function is a small anonymous function.
A lambda function can take any number of arguments, but can only have one expression.

Listing 6: Insert code directly in your document

```
x = lambda a, b : a * b
print(x(5, 6))
```

12 Class and Objects

Class is like bule-print to create objects.

Listing 7: Insert code directly in your document

```
class MyClass:
  x = 5
p1 = MyClass()
print(p1.x)
```

12.1 The init function

This is equivalent to constructor in cpp or Java

Listing 8: Insert code directly in your document

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

12.2 str function

This is equivalent to to_string() in java

Listing 9: Insert code directly in your document

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1)
```

The above code will return object reference for p1
where as it can be easily changed to our requirement

Listing 10: Insert code directly in your document

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def __str__(self):
    return f"{self.name}({self.age})"

p1 = Person("John", 36)

print(p1)
```

self is equivalent to this in java. It can be any variable as first parameter of the function.

Listing 11: Insert code directly in your document

```
class Person:
  def __init__(mysillyobject, name, age):
    mysillyobject.name = name
    mysillyobject.age = age

  def myfunc(abc):
    print("Hello my name is " + abc.name)

p1 = Person("John", 36)
p1.myfunc()
```

o/p : Hello my name is John

modity object parameter obj.z=10

Delete object parameter del obj.z

Deleting object del obj

13 Inheritance

Listing 12: Insert code directly in your document

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  pass

x = Student("Mike", "Olsen")
x.printname()
```

Child can't inherit the properties of the parent init function. In order to do so use the code below instead.

Listing 13: Insert code directly in your document

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname):
    Person.__init__(self, fname, lname)

x = Student("Mike", "Olsen")
x.printname()
```

13.1 Super keyword

The above code can be used with the same modification.

Listing 14: Insert code directly in your document

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname):
    super().__init__(fname, lname)

x = Student("Mike", "Olsen")
x.printname()
```

13.2 Adding properties to object

Listing 15: Insert code directly in your document

```
class Student(Person):
  def __init__(self, fname, lname, year):
    super().__init__(fname, lname)
    self.graduationyear = year

x = Student("Mike", "Olsen", 2019)
```

13.3 Adding Methods

Listing 16: Insert code directly in your document

```
class Student(Person):
  def __init__(self, fname, lname, year):
    super().__init__(fname, lname)
    self.graduationyear = year

  def welcome(self):
    print("Welcome", self.firstname,
          self.lastname,"to the class of",
          self.graduationyear)
```

13.4 Iterator

The for loop actually creates an iterator object and executes the next() method for each loop. To create an object/class as an iterator you have to implement the methods iter() and next() to your object. As you have learned in the Python Classes/Objects chapter, all classes have a function called init(), which allows you to do some initializing when the object is being created.The iter() method acts similar, you can do operations (initializing etc.), but must always return the iterator object itself.The next() method also allows you to do operations, and must return the next item in the sequence.

Listing 17: Insert code directly in your document

```
class MyNumbers:
  def __iter__(self):
    self.a = 1
    return self

  def __next__(self):
    x = self.a
    self.a += 1
    return x

myclass = MyNumbers()
myiter = iter(myclass)

print(next(myiter))
print(next(myiter))
print(next(myiter))
```

```
print(next(myiter))
print(next(myiter))
```

Another variation of iterator method is given below

Listing 18: Insert code directly in your document

```
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))
```

The example above would continue forever if you had enough next() statements, or if it was used in a for loop.

13.4.1 StopIteration

The below code stops after 20 iteration.

Listing 19: Insert code directly in your document

```
class MyNumbers:
  def __iter__(self):
    self.a = 1
    return self

  def __next__(self):
    if self.a <= 20:
      x = self.a
      self.a += 1
      return x
    else:
      raise StopIteration

myclass = MyNumbers()
myiter = iter(myclass)

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

14 Module

Save this code in a file named mymodule.py

Listing 20: Insert code directly in your document

```
def greeting(name):
  print("Hello, " + name)
```

Listing 21: Insert code directly in your document

```
import mymodule

mymodule.greeting("Jonathan")
```

14.1 Naming a Module

import mymodule as mx

14.2 dir()

There is a built-in function to list all the function names (or variable names) in a module. The dir() function:

Listing 22: Insert code directly in your document

```
import platform

x = dir(platform)
print(x)
```

14.3 from keyword

Let us consider below things to be the content in mymodule

Listing 23: Insert code directly in your document

```
def greeting(name):
  print("Hello, " + name)

person1 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

content of the module can be easily accessed

Listing 24: Insert code directly in your document

```
from mymodule import person1

print (person1["age"])
```

15 Type Casting

int()
float()
str()

16 Operators

16.1 Arithmetic Operators

+ Addition x + y

- Subtraction x - y

* Multiplication x * y

/ Division x / y

% Modulus x % y

** Exponentiation x ** y

// Floor division x // y

16.2 Assignment Operator

=

16.3 Comparison Operator

16.4 Logical Operator

and, or, not

16.5 Identity Operator

is, is not - Identify if the variables are the same objects

16.6 Membership Operator

in, not in - Sequence with the specified value is present in an object

16.7 Bitwise Operator

bitwise and or not using tilda, leftshift and rightshift

17 JSON

JSON = JavaScript Object Notation

17.1 json.loads()

If you have a JSON string, you can parse it by using the json.loads() method.

Listing 25: Insert code directly in your document

```
import json

# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'
# parse x:
y = json.loads(x)
# the result is a Python dictionary:
print(y["age"])
```

If you have a Python object, you can convert it into a JSON string by using the json.dumps() method.

Listing 26: Insert code directly in your document

```
import json

# a Python object (dict):
x = {
  "name": "John",
  "age": 30,
  "city": "New York"
}
# convert into JSON:
y = json.dumps(x)
# the result is a JSON string:
print(y)
```

Listing 27: Insert code directly in your document

```
import json

print(json.dumps({"name": "John", "age": 30}))
print(json.dumps(["apple", "bananas"]))
print(json.dumps(("apple", "bananas")))
print(json.dumps("hello"))
print(json.dumps(42))
print(json.dumps(31.76))
print(json.dumps(True))
print(json.dumps(False))
print(json.dumps(None))
```

Python	JSON
dict	Object
list	Array
tuple	Array
str	String
int	Number
float	Number
True	true
False	false
None	null

Listing 28: Insert code directly in your document

```
import json

x = {
  "name": "John",
  "age": 30,
  "married": True,
  "divorced": False,
  "children": ("Ann","Billy"),
  "pets": None,
  "cars": [
    {"model": "BMW 230", "mpg": 27.5},
    {"model": "Ford Edge", "mpg": 24.1}
  ]
}
print(json.dumps(x))
```

17.2 Format the Result

json.dumps(x, indent=4)
Use the separators parameter to change the default separator: use "." instead of "," and "=" instead of ":"
json.dumps(x, indent=4, separators=(".", " = "))
Use the sort_keys parameter to specify if the result should be sorted or not -**Order the Result**
json.dumps(x, indent=4, sort_keys=True)

18 Exception Handling

The try block lets you test a block of code for errors. The except block lets you handle the error. The else block lets you execute code when there is no error. The finally block lets you execute code, regardless of the result of the try- and except blocks.
Print one message if the try block raises a NameError and another for other errors:

Listing 29: Insert code directly in your document

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("An exception occurred")
finally:
  print("The 'try except' is finished")
```

You can use the **else** keyword to define a block of code to be executed if no errors were raised.
The **finally** block, if specified, will be executed regardless if the try block raises an error or not.This can be useful to close objects and clean up resources.

18.1 Raise an exception

As a Python developer you can choose to throw an exception if a condition occurs. To throw (or raise) an exception, use the raise keyword.

Listing 30: Insert code directly in your document

```
x = -1

if x < 0:
  raise Exception("Sorry, no numbers below zero")
```

The raise keyword is used to raise an exception. You can define what kind of error to raise, and the text to print to the user.

Listing 31: Insert code directly in your document

```
x = "hello"

if not type(x) is int:
  raise TypeError("Only integers are allowed")
```

19 User Input

Listing 32: Insert code directly in your document

```
username = input("Enter username:")
print("Username is:" + username)
```

20 String Formatting

Listing 33: Insert code directly in your document

```
price = 49.999
txt = "The price is {} dollars"
print(txt.format(price))
txt2 = "The price is {:.2f} dollars"

quantity = 3
itemno = 567
price = 49
myorder = "I want {0} pieces of item number {1} for
print(myorder.format(quantity, itemno, price))
```

20.1 Named Indexes

You can also use named indexes by entering a name inside the curly brackets carname, but then you must use names when you pass the parameter values txt.format(carname = "Ford"):

Listing 34: Insert code directly in your document

```
myorder = "I have a {carname}, it is a {model}."
print(myorder.format(carname = "Ford", model = "Musta
```

21 File Open

Creating, reading, updating, and deleting files. **open()** function takes two parameters; filename, and mode.

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

"r" for read, and "t" for text are the **default values**, you do not need to specify them. Note: Make sure the **file exists**, or else you will get an error.

21.1 Read File

Listing 35: Insert code directly in your document

```
f = open("D:\\myfiles\welcome.txt", "r")
print(f.read())
# Used to read first 5 characters of the file
print(f.read(5))
print(f.readline()) # Read one line
print(f.readline()) # Reads next line

#loops through the file contents
f = open("demofile.txt", "r")
for x in f:
  print(x)
f.close() # closing a file
```

21.2 Write/Create File

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

"x" - Create - will create a file, returns an error if the file exist

Note: the "w" method will overwrite the entire file.

Listing 36: Insert code directly in your document

```
# Create a new file if it does not exist:

f = open("myfile.txt", "w")
```

21.3 Delete File

must **import the OS module**, and run its os.remove() function:

Listing 37: Insert code directly in your document

```
import os
os.remove("demofile.txt")

# Check if the file exists
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")

# you can remove an empty directory through
os.rmdir()
```

22 Miscellaneous

(x) - type x will be printed
range(6) - 0,1,2,3,4,5.

\hfill - This will horizontally fill the line with white space Global keyword - Creating global variable inside inside local scope.

Listing 38: Insert code directly in your document

```
def myfunc():
  global x
  x = 300

myfunc()
print(x)
```

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

Listing 39: Insert code directly in your document

```
x = 300

def myfunc():
  global x
  x = 200

myfunc()
print(x)
```

pip list - List all the package installed in the system.

23 datetime

import a module named datetime to work with dates as date objects.To create a date, we can use the datetime() class (constructor) of the datetime module. The datetime() class requires three parameters to create a date: year, month, day.The datetime() class also takes parameters for time and timezone (hour, minute, second, microsecond, tzone), but they are optional, and has a default value of 0, (None for timezone).

```
import datetime

x = datetime.datetime.now()
print(x)
```

o/p - 2023-02-24 23:18:47.491683 The date contains year, month, day, hour, minute, second, and microsecond.

23.1 strftime()

method for formatting date objects into readable strings. It takes one parameter, format, to specify the format of the returned string.

```
import datetime

x = datetime.datetime(2018, 6, 1)
print(x.strftime("%B"))
```

o/p - June

[htbp] %a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%C	Century	20
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%
%G	ISO 8601 year	2018
%u	ISO 8601 weekday (1-7)	1
%V	ISO 8601 weeknumber (01-53)	01

24 Math Module

Math Methods

```
import math

x = math.sqrt(64)
x = math.ceil(1.4)
y = math.floor(1.4)
x = math.pi
# Built in functions
x = min(5, 10, 25)
y = max(5, 10, 25)
x = abs(-7.25)
x = pow(4, 3)
```

[htbp] Method	Description
math.acos()	Returns the arc cosine of a number
math.acosh()	Returns the inverse hyperbolic cosine of a number
math.asin()	Returns the arc sine of a number
math.asinh()	Returns the inverse hyperbolic sine of a number
math.atan()	Returns the arc tangent of a number in radians
math.atan2()	Returns the arc tangent of y/x in radians
math.atanh()	Returns the inverse hyperbolic tangent of a number
math.ceil()	Rounds a number up to the nearest integer
math.comb()	Returns the number of ways to choose k items from n items without repetition and order
math.copysign()	Returns a float consisting of the value of the first parameter and the sign of the second parameter
math.cos()	Returns the cosine of a number
math.cosh()	Returns the hyperbolic cosine of a number
math.degrees()	Converts an angle from radians to degrees
math.dist()	Returns the Euclidean distance between two points (p and q), where p and q are the coordinates of that point
math.erf()	Returns the error function of a number
math.erfc()	Returns the complementary error function of a number
math.exp()	Returns E raised to the power of x
math.expm1()	Returns Ex - 1
math.fabs()	Returns the absolute value of a number
math.factorial()	Returns the factorial of a number
math.floor()	Rounds a number down to the nearest integer
math.fmod()	Returns the remainder of x/y
math.frexp()	Returns the mantissa and the exponent, of a specified number
math.fsum()	Returns the sum of all items in any iterable (tuples, arrays, lists, etc.)
math.gamma()	Returns the gamma function at x
math.gcd()	Returns the greatest common divisor of two integers
math.hypot()	Returns the Euclidean norm
math.isclose()	Checks whether two values are close to each other, or not
math.isfinite()	Checks whether a number is finite or not
math.isinf()	Checks whether a number is infinite or not
math.isnan()	Checks whether a value is NaN (not a number) or not
math.isqrt()	Rounds a square root number downwards to the nearest integer
math.ldexp()	Returns the inverse of math.frexp() which is x * (2**i) of the given numbers x and i
math.lgamma()	Returns the log gamma value of x
math.log()	Returns the natural logarithm of a number, or the logarithm of number to base
math.log10()	Returns the base-10 logarithm of x
math.log1p()	Returns the natural logarithm of 1+x
math.log2()	Returns the base-2 logarithm of x
math.perm()	Returns the number of ways to choose k items from n items with order and without repetition
math.pow()	Returns the value of x to the power of y
math.prod()	Returns the product of all the elements in an iterable
math.radians()	Converts a degree value into radians
math.remainder()	Returns the closest value that can make numerator completely divisible by the denominator
math.sin()	Returns the sine of a number
math.sinh()	Returns the hyperbolic sine of a number
math.sqrt()	Returns the square root of a number
math.tan()	Returns the tangent of a number
math.tanh()	Returns the hyperbolic tangent of a number
math.trunc()	Returns the truncated integer parts of a number

Math Constants

Constant	Description
math.e	Returns Euler's number (2.7182...)
math.inf	Returns a floating-point positive infinity
math.nan	Returns a floating-point NaN (Not a Number) value
math.pi	Returns PI (3.1415...)
math.tau	Returns tau (6.2831...)

25 Regular Expression

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern. RegEx can be used to check if a string contains the specified search pattern.

```
# Search the string to see if it starts with "The" and ends with "Spain":
import re

txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)
```

25.1 RegEx Functions

Function	Description
findall	Returns a list containing all matches
search	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string

25.2 Metacharacters

Character	Description	Example
[]	A set of characters	"[a–m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"planet$"
*	Zero or more occurrences	"he.*o"
+	One or more occurrences	"he.+o"
?	Zero or one occurrences	"he.?o"
{}	Exactly the specified number of occurrences	"he.{2}o"
|	Either or	"falls|stays"
()	Capture and group	

25.3 Special Sequences

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string "\AThe"	
\b	Returns a match where the specified characters are at the beginning or at the end of a word(the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word(the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"ain\B" r"\Bain"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

25.4 Sets

Set	Description
[arn]	Returns a match where one of the specified characters (a, r, or n) is present
[a–n]	Returns a match for any lower case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a, r, and n

```
[0123]          Returns a match where any of the specified digits (0, 1, 2, or 3) are present
[0-9]           Returns a match for any digit between 0 and 9
[0-5][0-9]      Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]                Returns a match for any character alphabetically between a and z, lower case OR upper case
[+]                     In sets, +, *, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string
```

25.5 findall()

The findall() function returns a list containing all matches.

```
import re

txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
```

25.6 search()

The search() function searches the string for a match, and returns a Match object if there is a match.If no matches are found, the value None is returned. If there is more than one match, only the first occurrence of the match will be returned:
Search for the first white-space character in the string:

```
import re

txt = "The rain in Spain"
x = re.search("\s", txt)

print("The first white-space character is located in position:", x.start())
```

25.7 split() Function

The split() function returns a list where the string has been split at each match:

```
import re

txt = "The rain in Spain"
x = re.split("\s", txt)
print(x)
```

You can control the number of occurrences by specifying the maxsplit parameter:

```
import re

txt = "The rain in Spain"
x = re.split("\s", txt, 1)
print(x)
```

25.8 sub() Function

The sub() function replaces the matches with the text of your choice:Replace every white-space character with the number 9

```
import re

txt = "The rain in Spain"
x = re.sub("\s", "9", txt)
print(x)
```

You can control the number of replacements by specifying the count parameter:Replace the first 2 occurrences.

```
import re

txt = "The rain in Spain"
x = re.sub("\s", "9", txt, 2)
print(x)
```

25.9 Match Object

A Match Object is an object containing information about the search and the result. Note: If there is no match, the value None will be returned, instead of the Match Object.

```
import re

#The search() function returns a Match object:

txt = "The rain in Spain"
x = re.search("ai", txt)
print(x)
print(x.span())
print(x.string)
print(x.group())
```

.span() returns a tuple containing the start-, and end positions of the match.
.string returns the string passed into the function.
.group() returns the part of the string where there was a match.
Print the position (start- and end-position) of the first match occurrence. The regular expression looks for any words that starts with an upper case "S":

```
import re

txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.span())
```

Note: If there is no match, the value None will be returned, instead of the Match Object.