APSSDC

Andhra Pradesh State Skill Development Corporation

(https://apssdc.in)

Day 13 Packages and Modules in Python

Recap

Files in Python

- Open
- Do some Operations
- Close

Day13 Objectives

- Positional Arguments.Reguires
- Keyword Arguments
- Default Arguments
- Variable Length arguments -> Arbitrary Functoins
- Call by Value
- Call by reference
- Examples in Files
- Modules and Packages

In [2]:
```
def addition(a, b):
    return a + b
```

In [3]:
```
print(addition(5, 5))
```

```
10
```

Variable Length arguments

In [4]:
```
def addition(a, *b): # *args
    print(a)
    print(b)
    return
```

In [5]:
```
print(addition(5, 5))
```

```
5
(5,)
None
```

In [6]:
```
addition(5,8,6,7,9,3,5,55,7,5,5)
```

```
5
(8, 6, 7, 9, 3, 5, 55, 7, 5, 5)
```

In [7]:
```
def addition(*a, *b): # *args
    print(a)
    print(b)
    return
```

```
  File "<ipython-input-7-d3fa43dceabc>", line 1
    def addition(*a, *b): # *args
                      ^
SyntaxError: invalid syntax
```

In [8]:
```
def addition(*a, b): # *args
    print(a)
    print(b)
    return
```

In [9]:
```
addition(5,5,10,324,64,534,35)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-6d7e01db062d> in <module>
----> 1 addition(5,5,10,324,64,534,35)

TypeError: addition() missing 1 required keyword-only argument: 'b'
```

In [10]:
```
addition(5,5,10,324,64,534,35, b = 55)
```

```
(5, 5, 10, 324, 64, 534, 35)
55
```

In [12]:
```
def addition(*a, b): # *args
    s = b
    for num in a:
        s += num
    return s
```

In [13]:
```
addition(5,5,10,324,64,534,35, b = 55)
```

Out[13]: 1032

In [14]:
```
def addition(*a, b): # *args
    return sum(a) + b
```

In [15]:
```
addition(5,5,10,324,64,534,35, b = 55)
```

Out[15]: 1032

In [17]:
```
def addition(**var): # *kwargs
    return var
```

In [18]:
```
print(addition(d))
```

```
{'a': 1, 'b': 10, 'c': 25, 'd': 35, 'abc': 55}
```

In [19]:
```
def addition(*var): # *kwargs
    return var
```

In [20]:
```
print(addition(a=1, b=10,c=25,d=35,abc=55))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-42c4d7724927> in <module>
----> 1 print(addition(a=1, b=10,c=25,d=35,abc=55))

TypeError: addition() got an unexpected keyword argument 'a'
```

In [23]:
```
print(1,5,79,6,8,68,6,22,96,3,59,6,332, sep = '\t')
```

```
1	5	79	6	8	68	6	22	96	3
59	6	332
```

In [25]:

```
import pandas as pd


print(pd.read_csv.__doc__)
```

```
Read a comma-separated values (csv) file into DataFrame.

Also supports optionally iterating or breaking of the file
into chunks.

Additional help can be found in the online docs for
`IO Tools <https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html>`_.

Parameters
----------
filepath_or_buffer : str, path object or file-like object
    Any valid string path is acceptable. The string could be a URL. Valid
    URL schemes include http, ftp, s3, and file. For file URLs, a host is
    expected. A local file could be: file://localhost/path/to/table.csv.

    If you want to pass in a path object, pandas accepts any ``os.PathLike``.
```

In [42]:

```
def read_csv(x, **args):
    print(args)
    print(args['a'] * args['b'])
    print(args['c'] + args['d'])
    print(args['d'] + args['e'])
```

In [35]:

```
def read_csv2(a,b,c,d,e):
    print(a * b)
    print(c + d)
    print(d + e)
```

In [43]:

```
read_csv(55, a = 5, b = 25, c = 55, d = 99, e = 365, f = 255, m = 758)
```

```
{'a': 5, 'b': 25, 'c': 55, 'd': 99, 'e': 365, 'f': 255, 'm': 758}
125
154
464
```

In [37]:

```
read_csv2(5,6,8,5,5)
```

```
30
13
10
```

In [46]:

```
def open_file(fileName):
    s = open(fileName, 'r')
    data = s.read()
    s.close()
    return data


def open_file2(fileName):
    with open(fileName, 'r') as f:
        data = f.read()
    return data
```

In [47]:

```
data = open_file('data_file.txt')

print(data)
```

```
Python consistently ranks as one of the most popular programming language
s.[34][35][36][37][38]


Contents
1       History
2       Design philosophy and features
3       Syntax and semantics
3.1     Indentation
3.2     Statements and control flow
3.3     Expressions
3.4     Methods
3.5     Typing
3.6     Arithmetic operations
4       Programming examples
5       Libraries
6       Development environments
7       Implementations
```

In [51]:

```
def freq_words(data):
    data = data.split()
    print(len(data))

    uniq = set(data)
    print(len(uniq))

    freq = {}
    for word in uniq:
        freq[word] = data.count(word)
    return freq
```

In [52]:
```
1 freq_words(data)
```

```
'body': 1,
'It': 4,
'programming': 12,
'9': 1,
'postponed': 1,
'15': 1,
'library': 2,
'3.[48]': 1,
"Rossum's": 1,
'20),': 1,
'Monty': 2,
'large': 2,
'Warsaw,': 1,
'1': 1,
'style.': 1,
'API': 2,
'7.5': 1,
'made': 1,
'working': 1,
```

Variable scope and lifetime of variables

- Global Variable -> It can be used anywhere in the program -> Lifetime will be untill the execution of the program
- Local Variables -> It can be used only inside the functions -> Lifetime will be untill the execution of the funtion

In [53]:
```
1 print(data)
```

```
Python is an interpreted high-level general-purpose programming language.
Python's design philosophy emphasizes code readability with its notable u
se of significant indentation. Its language constructs as well as its obj
ect-oriented approach aim to help programmers write clear, logical code f
or small and large-scale projects.[30]

Python is dynamically-typed and garbage-collected. It supports multiple p
rogramming paradigms, including structured (particularly, procedural), ob
ject-oriented and functional programming. Python is often described as a
"batteries included" language due to its comprehensive standard library.
[31]

Guido van Rossum began working on Python in the late 1980s, as a successo
r to the ABC programming language, and first released it in 1991 as Pytho
n 0.9.0.[32] Python 2.0 was released in 2000 and introduced new features,
such as list comprehensions and a garbage collection system using referen
ce counting. Python 3.0 was released in 2008 and was a major revision of
the language that is not completely backward-compatible and much Python 2
code does not run unmodified on Python 3. Python 2 was discontinued with
```

In [55]:

```
1 print(uniq)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-55-0cdb91e3e4ae> in <module>
----> 1 print(uniq)

NameError: name 'uniq' is not defined
```

In [56]:

```
1 a = 55
2
3 def fun(b):
4     print(a)
5     print(b)
6
7 print(a)
8 fun(123)
9 print(b)
```

```
55
55
123

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-56-aecc8cfd32d9> in <module>
      7 print(a)
      8 fun(123)
----> 9 print(b)

NameError: name 'b' is not defined
```

In [58]:

```python
a = 55

def fun(b):
    print(a) #-> globalVar
    print(b) #-> local
    def fun2(c):
        print(a) #-> globalVar
        print(b) #-> LocalVar
        print(c)
    fun2(365)
    print(c) #-> nonLocal

print(a)
fun(123)
print(b)
```

```
55
55
123
55
123
365

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-58-21665f3478b6> in <module>
     12
     13 print(a)
---> 14 fun(123)
     15 print(b)

<ipython-input-58-21665f3478b6> in fun(b)
      9         print(c)
     10     fun2(365)
---> 11     print(c) #-> nonLocal
     12
     13 print(a)

NameError: name 'c' is not defined
```

Call by Value

If i'm pass the immutable data type variables to the function

Call by reference

If i'm pass the mutable data type variables to the function

In [54]:
```
print(data)
```

```
Python is an interpreted high-level general-purpose programming language.
Python's design philosophy emphasizes code readability with its notable u
se of significant indentation. Its language constructs as well as its obj
ect-oriented approach aim to help programmers write clear, logical code f
or small and large-scale projects.[30]

Python is dynamically-typed and garbage-collected. It supports multiple p
rogramming paradigms, including structured (particularly, procedural), ob
ject-oriented and functional programming. Python is often described as a
"batteries included" language due to its comprehensive standard library.
[31]

Guido van Rossum began working on Python in the late 1980s, as a successo
r to the ABC programming language, and first released it in 1991 as Pytho
n 0.9.0.[32] Python 2.0 was released in 2000 and introduced new features,
such as list comprehensions and a garbage collection system using referen
ce counting. Python 3.0 was released in 2008 and was a major revision of
the language that is not completely backward-compatible and much Python 2
code does not run unmodified on Python 3. Python 2 was discontinued with
```

In [59]:
```
def call_value(val):
    return val
```

In [60]:
```
a = 55
call_value(a)
```

Out[60]:
```
55
```

In [61]:
```
def call_value(val):
    print(val, id(val))
    val = 625
    return val, id(val)
```

In [62]:
```
a = 55
call_value(a)
```

```
55 140710091439584
```

Out[62]:
```
(625, 2967215743952)
```

In [64]:
```
def call_ref(val):
    print(val, id(val))
    val.append(625)
    return val, id(val)
```

In [66]:
```
li = [1,2,3,4,5]
print(id(li))

print(call_ref(li))

print(li)
```

```
2967204985536
[1, 2, 3, 4, 5] 2967204985536
([1, 2, 3, 4, 5, 625], 2967204985536)
[1, 2, 3, 4, 5, 625]
```

In [67]:
```
li = [1,2,3,4,5]
print(id(li))

print(call_ref(li.copy())) # Call ref

print(li)
```

```
2967204829376
[1, 2, 3, 4, 5] 2967205274816
([1, 2, 3, 4, 5, 625], 2967205274816)
[1, 2, 3, 4, 5]
```

In [68]:
```
print(id(li[:]))
```

```
2967206625920
```

recursive Fun

Modules and Packages

- Pre-defined/ builtIn
- user defined/ 3rd party packages

Modules

single py file contains functions/ attributes/ calsses constructed for some purpose

packages

group of py files/ modules contains functions/ attributes/ calsses constructed for some purpose

adavantage

- reducing the complexity of application
- user readibility
- reusability

Predifined/ builtin

In [69]:
```
1 import math
```

In [72]:
```
1 print(math.__doc__)
```

```
This module provides access to the mathematical functions
defined by the C standard.
```

In [71]:
```
1 dir(math)
```

Out[71]:
```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'comb',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'dist',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'gcd',
 'hypot',
 'inf',
 'isclose',
 'isfinite',
 'isinf',
 'isnan',
 'isqrt',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log1p',
 'log2',
 'modf',
 'nan',
 'perm',
 'pi',
 'pow',
 'prod',
 'radians',
 'remainder',
 'sin',
```

```
 'sinh',
 'sqrt',
 'tan',
 'tanh',
 'tau',
 'trunc']
```

In [73]:
```
1 print(math.pi) #moduleName/PackageName.fun/attributeName
```

```
3.141592653589793
```

In [74]:
```
1 print(math.sin(90))
```

```
0.8939966636005579
```

In [75]:
```
1 print(math.factorial(5))
```

```
120
```

In [77]:
```
1 print(math.dist((1,3), (2,4)))
```

```
1.4142135623730951
```

In [80]:
```
1 import sys
2
3 sys.version
```

Out[80]:
```
'3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]'
```

In [ ]:
```
1
```