PYTHON3

[HOME PAGE](https://www.python.org/)

[Built-in Functions and others](https://docs.python.org/3/library/index.html)

[w3schools](https://www.w3schools.com/python/)

https://docs.python.org/3/library/index.html

Evreone command begin from first colomn,

in cicle while if from secon third ...

if not return „IndentationError: unexpected indent“

Errors

1./usr/lib/python3/site-packages/requests/\_\_init\_\_.py:91: RequestsDependencyWarning: urllib3 (1.25.2) or chardet (3.0.4) doesn't match a supported version! RequestsDependencyWarning)

[RequestsDependencyWarning: urllib3 (1.25.2) or chardet (3.0.4) doesn't match a supported version!](https://stackoverflow.com/questions/56155627/requestsdependencywarning-urllib3-1-25-2-or-chardet-3-0-4-doesnt-match-a-s)

Solved :

- pip3 install --upgrade requests

- pip3 install requests

- pip install --upgrade requests==2.20.1

2. from PIL import ImageTk, Image

ImportError: cannot import name 'ImageTk' from 'PIL' (/usr/lib/python3/dist-packages/PIL/\_\_init\_\_.py

Solved : sudo apt-get install python3-pil python3-pil.imagetk

3.

4.

5.

1. Python 3 Tutorial - https://www.programiz.com/python-programming/tutorial

2. Top 10 Python Project Ideas for Beginners - https://hackr.io/blog/python-projects

3. Online Python Compiler  **-** [**https://www.tutorialspoint.com/execute\_python\_online.php**](https://www.tutorialspoint.com/execute_python_online.php)

4. Compile file .PY to file.PYC

- run ~/work/library/python/compile.sh -> create /*\_\_*pycache*\_\_*/calc.cpython-38.pyc

5. Decompile file .PYC to file.PY

- unzip -> ~/IDISK/LINUX/32-64/64/uncompyle6-3.8.0.tar.gz

- install -> run uncompyle6-3.8.0/setup.py

- check install ->

- run ~/work/library/python/*\_\_*pycache*\_\_*/decompile.sh -> create calc..py

6. Decompile all .pyc files in dir -> decompile-pyc-dir-recursively.py

7. **Cicle While loop break**

i = 1

while True:

print(i)

i = i + 1

if(i > 3):

break

8. **Func**

def f():

i=0

while True:

print(i)

i = i + 1

if(i > 3):

break

f()

9. python3 -m pip install pandas

10. **for**

import time

nums = []

for n in range(10):

nums.append(str(n))

print("".join(nums)) # much more efficient

print("|".join(nums)) # much more efficient

print((nums)) # much more efficient

print(nums) # much more efficient

time.sleep(2)

11. **Open file**

**'w' Open a text file for writing. If the file exists, the function will truncate all the contents as soon as you open it. If the file doesn’t exist, the function creates a new file.**

**'a' Open a text file for appending text. If the file exists, the function append contents at the end of the file.**

**‘+’ Open a text file for updating (both reading & writing).**

**- open**

f = open("aaa.txt", "r+")

with open('aaa.txt', "r+") as f: or

**- read**

f = open("aaa.txt", "r")

print(f.readline().rstrip())

print(f.readline().rstrip())

print(f.readline().rstrip())

print(f.readline().rstrip())

#print (f.close()) # NOT this work

with open('aaa.txt', "r") as f:

print(f.readlines())

with open('aaa.txt', "r") as f:

for line in f:

print(line, end='')

with open("aaa.txt") as f:

for i in f:

print(i.rstrip('\n'))

for line in open("aaa.txt"):

print(line, end="")

with open("myfile.txt") as f:

for line in f:

print(line, end="")

print(f.closed)

for line in open("modules.txt"):

print(line, end='')

**- write**

# f = open('aaa.txt', 'w')

with open('aaa.txt', 'a') as f:

f.write('\ncccccc\n')

f.write('dddddd')

f = open("aaa.txt", "r")

print(f.readline().rstrip())

print(f.readline().rstrip())

print(f.readline().rstrip())

print(f.readline().rstrip())

print(f.closed)

**12. Compare string**

print ((1, 2, 3) < (1, 2, 4))

print ([1, 2, 3] < [1, 2, 4])

print ('ABC' < 'C' < 'Pascal' < 'Python')

print ((1, 2, 3, 4) < (1, 2, 4))

print ((1, 2) < (1, 2, -1))

print ((1, 2, 3) == (1.0, 2.0, 3.0))

print ((1, 2, ('aa', 'ab')) < (1, 2, ('abc', 'a'), 4))

13.  **fibo1.py call functions from ibo.py**

**- fibo.py -> functions**

def fib(n): # write Fibonacci series up to n

a, b = 0, 1

while a < n:

print(a, end=' ')

a, b = b, a+b

print()

def fib2(n): # return Fibonacci series up to n

#result = []

result = []

# a, b = 0, 1

a, b = 0, 1

while a < n:

result.append(a)

a, b = b, a+b

return result

**- fibo1.py -> call functions fibo.py file**

import fibo

print(fibo.fib(1000)) ; print(fibo.fib2(1000))

#from fibo import fib, fib ; print(fib(500))

#from fibo import \* ; print(fib(500))

#import fibo as fib ; fib.fib(500)

#from fibo import fib as fibonacci ; fibonacci(500)

#exec(open('fibo.py').read())

# import fibo.py ; pibo.py

**13. Return** statement

|  |  |
| --- | --- |
| A return statement is used to end the execution of the function call and “returns” the result (value of the expression following the return keyword) to the caller. The statements after the return statements are not executed. If the return statement is without any expression, then the special value None is returned. A return statement is overall used to invoke a function so that the passed statements can be executed.  Note: Return statement can not be used outside the function. | Инструкцията за връщане се използва за прекратяване на изпълнението на извикването на функцията и „връща“ резултата (стойността на израза след ключовата дума return) на извикващия. Операторите след операторите за връщане не се изпълняват. Ако операторът return е без израз, тогава се връща специалната стойност None. Операторът за връщане се използва като цяло за извикване на функция, така че предадените оператори да могат да бъдат изпълнени.  Забележка: Инструкцията за връщане не може да се използва извън функцията. |

- example

demonstrate return statement

def add(a, b):

# returning sum of a and b

return a + b # if not a+b

def is\_true(a):

# returning boolean of a

return bool(a)

# calling function

res = add(2, 3)

print("Result of add function is {}".format(res))

res = is\_true(2<5)

print("\nResult of is\_true function is {}".format(res))

# **Not code** if not a+b -> print("Result of add function is {}".format(res)) ->

because python not know what doe a and b?

**14.** Print print(dir(sys)) -> names a module – Nothing do ????

**15. Packages and moduls**  -> Pillow, [Django](https://www.djangoproject.com/), The requests, BeautifulSoup4, [Pygame](http://pygame.org/hifi.html), Selenium,

[PyMongo](https://api.mongodb.org/python/current/)

**16.** Substr Subs print(f.seek(1)) ; print(f.read(3)) ; print(f.readline().rstrip())

**17.** <\_io.TextIOWrapper name='aaa.txt' mode='r' encoding='UTF-8'>

f = open('aaa.txt') ; s = f.readline()

**18.** Class

class MyClass:

i = 12345

def f():

a = "abcde"

print(a)

return 'hello world'

print(i)

f()

**- Class \_\_init\_\_(**

class Complex:

def \_\_init\_\_(self, realpart, imagpart):

self.r = realpart

self.i = imagpart

x = Complex(3.0, -4.5)

print(x.r, x.i)

**- other**

class Dog:

kind = 'canine' # class variable shared by all instances

def \_\_init\_\_(self, name):

self.name = name # instance variable unique to each instance

d = Dog('Fido')

e = Dog('Buddy')

print(d.kind) # shared by all dogs

print(e.kind) # shared by all dogs

print(d.name) # unique to d

print(e.name) # unique to e

|  |  |
| --- | --- |
| WHAT NOW?  Reading this tutorial has probably reinforced your interest in using Python — you should be eager to apply  Python to solving your real-world problems. Where should you go to learn more?  This tutorial is part of Python’s documentation set. Some other documents in the set are:  • library-index:  You should browse through this manual, which gives complete (though terse) reference material about  types, functions, and the modules in the standard library. The standard Python distribution includes  a lot of additional code. There are modules to read Unix mailboxes, retrieve documents via HTTP,  generate random numbers, parse command-line options, write CGI programs, compress data, and many  other tasks. Skimming through the Library Reference will give you an idea of what’s available.  • installing-index explains how to install additional modules written by other Python users.  • reference-index: A detailed explanation of Python’s syntax and semantics. It’s heavy reading, but is  useful as a complete guide to the language itself.  More Python resources:  • https://www.python.org: The major Python Web site. It contains code, documentation, and pointers  to Python-related pages around the Web. This Web site is mirrored in various places around the world,  such as Europe, Japan, and Australia; a mirror may be faster than the main site, depending on your  geographical location.  • https://docs.python.org: Fast access to Python’s documentation.  • https://pypi.org: The Python Package Index, previously also nicknamed the Cheese Shop, is an index  of user-created Python modules that are available for download. Once you begin releasing code, you  can register it here so that others can find it.  • https://code.activestate.com/recipes/langs/python/: The Python Cookbook is a sizable collection of  code examples, larger modules, and useful scripts. Particularly notable contributions are collected in  a book also titled Python Cookbook (O’Reilly & Associates, ISBN 0-596-00797-3.)  • http://www.pyvideo.org collects links to Python-related videos from conferences and user-group meet-  ings.  • https://scipy.org: The Scientific Python project includes modules for fast array computations and  manipulations plus a host of packages for such things as linear algebra, Fourier transforms, non-linear  solvers, random number distributions, statistical analysis and the like.  For Python-related questions and problem reports, you can post to the newsgroup comp.lang.python, or  send them to the mailing list at python-list@python.org. The newsgroup and mailing list are gatewayed,  so messages posted to one will automatically be forwarded to the other. There are hundreds of postings a  day, asking (and answering) questions, suggesting new features, and announcing new modules. Mailing list  archives are available at https://mail.python.org/pipermail/. | СЕГА КАКВО?  Четенето на този урок вероятно е засилило интереса ви към използването на Python — трябва да сте нетърпеливи да кандидатствате  Python за решаване на вашите проблеми от реалния свят. Къде трябва да отидете, за да научите повече?  Този урок е част от комплекта документация на Python. Някои други документи в комплекта са:  • библиотечен индекс:  Трябва да прегледате това ръководство, което дава пълен (макар и кратък) справочен материал за  типове, функции и модули в стандартната библиотека. Стандартната дистрибуция на Python включва  много допълнителен код. Има модули за четене на Unix пощенски кутии, извличане на документи чрез HTTP,  генериране на произволни числа, анализиране на опции на командния ред, писане на CGI програми, компресиране на данни и много други  други задачи. Прегледът на Библиотечния справочник ще ви даде представа какво е налично.  • installing-index обяснява как да инсталирате допълнителни модули, написани от други потребители на Python.  • reference-index: подробно обяснение на синтаксиса и семантиката на Python. Това е тежко четиво, но е  полезно като пълно ръководство за самия език.  Още ресурси за Python:  • https://www.python.org: Основният уеб сайт на Python. Той съдържа код, документация и указатели  към страници, свързани с Python в мрежата. Този уеб сайт е огледален на различни места по света,  като Европа, Япония и Австралия; огледало може да е по-бързо от основния сайт, в зависимост от вашия  географско местоположение.  • https://docs.python.org: Бърз достъп до документацията на Python.  • https://pypi.org: Индексът на пакетите на Python, по-рано също наричан Cheese Shop, е индекс  създадени от потребители Python модули, които са достъпни за изтегляне. След като започнете да пускате код, вие  можете да го регистрирате тук, за да могат други да го намерят.  • https://code.activestate.com/recipes/langs/python/: Готварската книга на Python е значителна колекция от  примери за кодове, по-големи модули и полезни скриптове. Особено забележителни приноси са събрани в  книга, също озаглавена Python Cookbook (O’Reilly & Associates, ISBN 0-596-00797-3.)  • http://www.pyvideo.org събира връзки към видеоклипове, свързани с Python, от конференции и срещи на потребителски групи  ings.  • https://scipy.org: Проектът Scientific Python включва модули за бързи изчисления на масиви и  манипулации плюс множество пакети за неща като линейна алгебра, трансформации на Фурие, нелинейни  решаващи програми, разпределения на случайни числа, статистически анализ и други подобни.  За въпроси и доклади за проблеми, свързани с Python, можете да публикувате в дискусионната група comp.lang.python или  изпратете ги в пощенския списък на python-list@python.org. Дискусионната група и пощенският списък са шлюзирани,  така че съобщенията, публикувани до единия, автоматично ще бъдат препращани към другия. Има стотици публикации a  ден, задавайки (и отговаряйки) на въпроси, предлагайки нови функции и обявявайки нови модули. Пощенски списък  архивите са достъпни на https://mail.python.org/pipermail/. |

----------------------------------------------------

**Python basics Modules**

Python basics

First steps

Variables and simple data types

Booleans and control structures with while iteration

Creating a simple console version

List, tuples, and sets

Dictionaries and for loops

Modules and standard library

Functions

Comprehensions and Exceptions

Files I/O and Context Managers, JSON

Retrospection

Advanced Python with OOP

Functions as first-class objects

Decorators

Classes, scope, and namespaces

Inheritance

Polymorphism

Property decorators and recap

Iterators and Generators

Basics of testing

Context managers extended

Retrospection

Classical computer science, algorithms and data structures

Big O notation and python built-in datastructures

Recursion

Data structures (Stack, Queue, Deque)

Data structures (Lists)

Searching algorithms and hashing

Graphs and graph algorithms

Databases

SQL basics. Part 1

SQL basics. Part 2

Concurrent programming

Basics of networking programming

Introduction to HTTP, making HTTP requests

Threads

Multiprocessing

Asyncio

Practice: Creating a chat

Advanced Python with OOP

Functions as first-class objects

Decorators

Classes, scope, and namespaces

Inheritance

Polymorphism

Property decorators and recap

Iterators and Generators

Basics of testing

Context managers extended

Retrospection

Web frameworks, ORMand Final project

Django. First steps

HTML, CSS, and template language basics

Django ORM and working with databases

Django views. Working with queries using Django

Unit testing for an application

Advanced functionality: HTML, CSS, and Django Templates

Authentication, authorization, and admin page in Django

Async with Django

Facebook API

Working on the final project. Part 1

Working on the final project. Part 2

Working on the final project. Part 3

Working on the final project. Part 4

Project presentation

**Python basics Modules\_END**

PAVEL

Инсталация и синтаксис

IDE & код редактори, Jupyter

Променливи

Стрингове и методи

Списъци и действия с тях

If и While функции и логични оператори

Цикли, break, continue, pass, range

Итератори и генератори

Функции

Рекурсия, безименни функции

Сетове и тъпъли

Речници и действия с тях

Input & output - вход и изход

Практически задачи

OOP, Класове, Наследяване на класове

Организиране на библиотеки, модули

Math, Random, Calendar, OS, Sys

Работа с файлове, терминал и библиотека argparse

Функционално програмиране с Python

Многонишково програмиране

Бази данни SQL, NoSQL, SQLAlchemy

Практически задачи

Графичен интерфейси Tkinter, PyQT

Unit tests

Machine Learning с Pandas

Декоратори, Криптография, Регулярни изрази

Web програмиране, Requests, Scraping

Django

Задание за финален проект