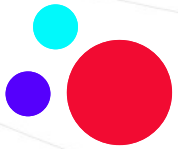
infoShare ACADEMY

Python Podstawy – Intel

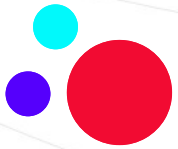
infoShare Academy

infoShareAcademy.com



Funkcje

```
def funkcja():
    # kod funkcji który zostanie wykonany przy jej wywołaniu
    # wiele linijek...
    # tak dużo dopóki jest wcięcie

    def do_nothing():
        pass

do_nothing()                # nic nie zrobiła, czyli tak jak chcieliśmy :)
wynik = do_nothing          # funkcja jest obiektem (jak wszystko w pythonie)
wynik()                     # więc możemy przypisać ją do zmiennej i ją wykonać
```



Funkcje – dokumentacja (docstring PEP257)

```
def do_nothing(x, y):
        """Does absolutely nothig"""
        pass


def give_square(x):
        """Return square of given number

        (number) → number
        """
        return x**2


give_square.__doc__
```

Funkcje - kolejność

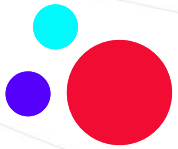
```
1  give_square(35)
2
3  def give_square(x):
4      print(x**2)
5
```

NIE

```
1  def give_square(x):
2      print(x**2)
3
4  give_square(35)
5
```

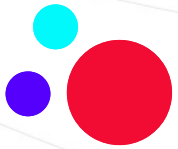
TAK

```
    give_square(35)
NameError: name 'give_square' is not defined
```



Funkcje – argumenty

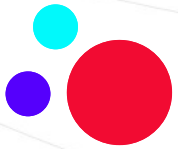
```
def do_nothing():                          # brak argumentów
    pass

def do_nothing(x):                         # jeden argument
    pass

def do_nothing(x, y, z):                   # 3 argumenty
    pass

def do_nothing(*args, **kwargs):           # dowolna ilość argumentów
    pass
```



Funkcje – argumenty domyślne

```
def do_nothing(x, y=10):
        pass

def do_nothing(x, y, name="Ola", age="18"):
        pass

def do_nothing(x, y, name="Ola", age="18", city):
        pass

def do_nothing(y=10):
        pass
```

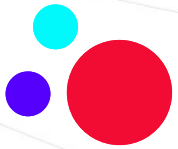
- argumenty domyślne **muszą** być po argumentach wymaganych
- argument domyślny jest sprawdzany **tylko przy pierwszym** wywołaniu funkcji – uwaga na typy referencyjne



Funkcje – typowanie (type hinting)

```
def do_nothing(x, y: int=10):
        pass

def do_nothing(x, y, name: str=”Ola”, age: int=”18”):
        pass

def simple_return(x: int) -> str:
        return x
```

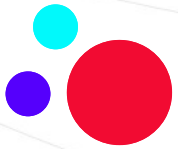
- Typy są sprawdzane w trakcie działania programu (runtime), a nie przed kompilacją (compile time).
- Adnotacje typów (: int, -> str) są tylko metadanymi. Interpreter Pythona ignoruje te adnotacje. Używa ich tylko w celach dokumentacji i nie wymusza ich przestrzegania.



Funkcje – typy referencyjne w argumentach domyślnych

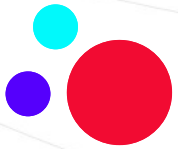
```
def dodaj_imie(imie, imiona=[]):
        imiona.append(imie)
        return imiona

>>> print(dodaj_imie("Iza"))
["Iza"]
>>> print(dodaj_imie("Kasia"))
["Iza", "Kasia"]
>>> print(dodaj_imie("Gosia"))
["Iza", "Kasia", "Gosia"]
```

```
def dodaj_imie(imie, imiona=None):
        if(imiona is None):
                imiona = []
        imiona.append(imie)
        return imiona

>>> print(dodaj_imie("Iza"))
["Iza"]
>>> print(dodaj_imie("Kasia"))
["Kasia"]
>>> print(dodaj_imie("Gosia"))
["Gosia"]
```



Funkcje – wywołanie

```
def do_nothing(x, y, name="Ola", age: int=18):
        pass


do_nothing(1)
do_nothing(1, 2, "Iza")
do_nothing(1, 2, "Iza", 22)
do_nothing(1, name="Iza", 2)
do_nothing(1, 2, name="Iza", age=22)
do_nothing(1, 2, name="Iza", age="22")
do_nothing(1, 2, age=22, name="Iza")
```



Funkcje – zwracanie wartości

Funkcja może wykonywać coś w sobie:

```
def show(x):
        print(x)
```

lub zwracać dowolne wartości:

```
def add(x, y):
        sum = x + y
        return sum

wynik = add(2, 3)
```

Funkcje – zakres zmiennych (scope)

Zmienne lokalne funkcji są do wykorzystania tylko w tej funkcji (i głębiej).
Zmiany za pomocą **global** i **nonlocal**

```
name = ”Jola”
def change_name():
        # print(name) # *
        name = ”Teresa”


print(name)
???
print(change_name())
???
print(name)
???
```

* Zmienna zdefiniowana globalnie jest domyślnie dostępna do odczytu wewnątrz dowolnej funkcji.

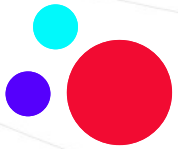
Wyjątki

Syntax Error *= błąd w składni polecenia*

Exception, ***Error*** *= wyjątki – to inaczej błędy powstałe w trakcie wykonywania programu np.*

- *błąd dzielenia przez zero,*
- *brak zdefiniowanej zmiennej,*
- *odwołanie się do nie istniejącego indeksu*
- *itp*

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



Wyjątki – typy

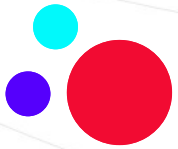
Staramy się wyłapywać konkretne typy wyjątków, np.
***ValueError**, **NameError** zamiast ogólnego wyjątku Exception.*

Bloki except deklarujemy od szczegółu do ogółu

*W przypadku kilku błędów, **tylko** jeden blok except zostanie wywołany, ten najwyżej.*

Wyjątek można samemu wywołać celowo!

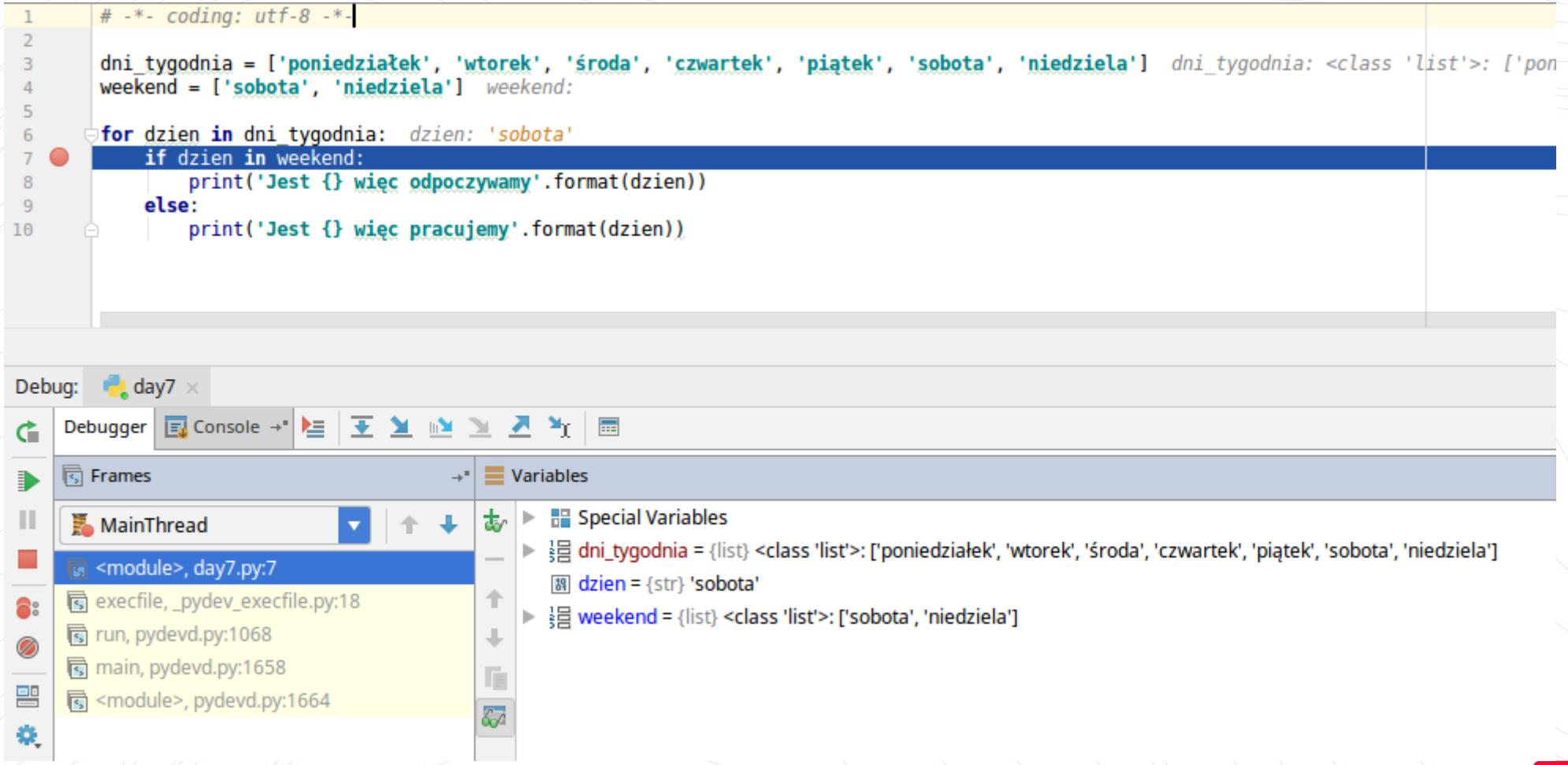
```
raise ValueError(„Nasz komunikat”)
raise DziwnyWyjatek
```



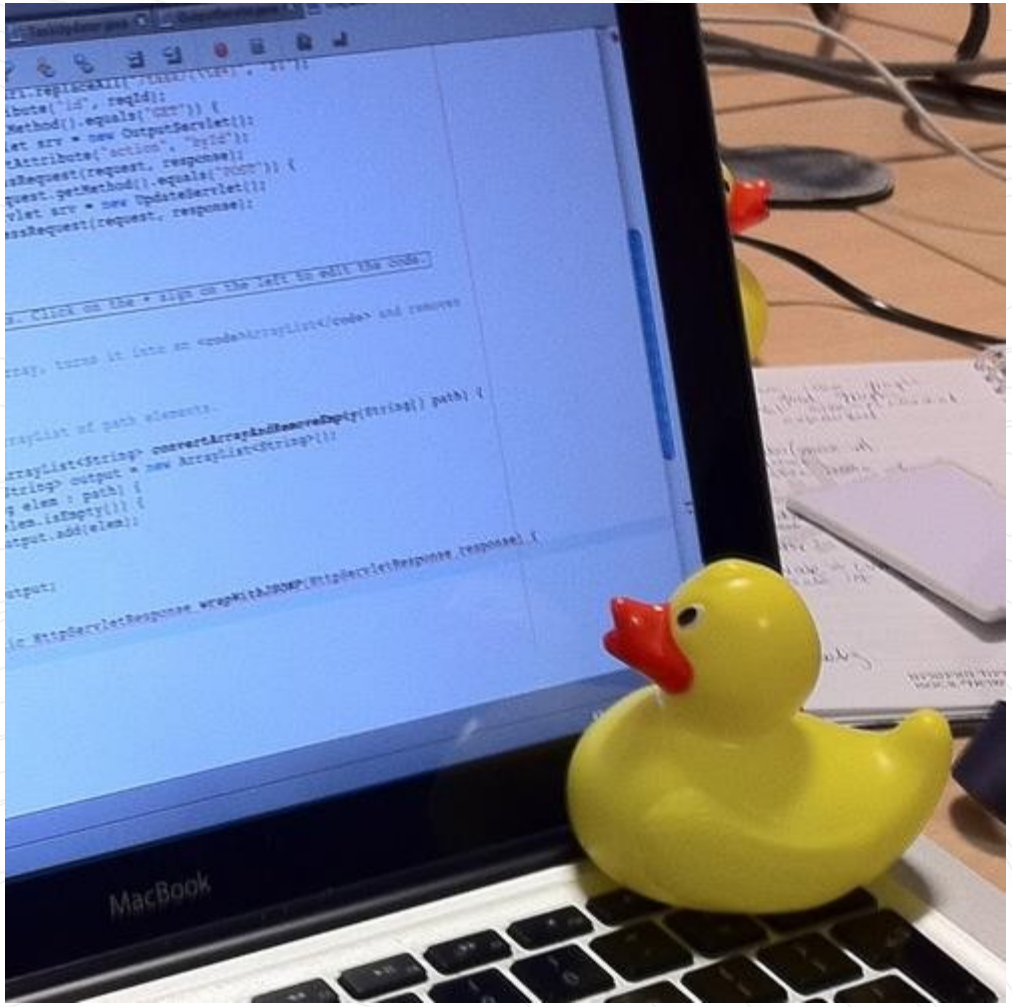
Wyjątki – łapanie

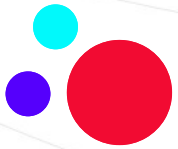
```
try:
        raise RuntimeError("Oops!")
        #raise DziwnyWyjatek
        #pass
except RuntimeError: # (RuntimeError, Exception)
        print("Wystąpił wyjątek Runtime Error")
except:
        print("Wystąpił nieokreślony wyjątek")
else:
        print("Wykonam się tylko jeśli nie było żadnego wyjątku")
finally:
        print("Zawsze się wykonam!")
```

Debuggowanie



Debuggowanie – metoda kaczuszki

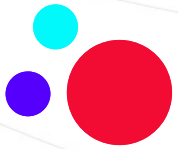




Refaktoryzacja

Proces poprawiania struktury kodu, **bez** zmiany jego funkcjonalności.

infoShare ACADEMY



Moduły

UWAGA!

PyCharm tworzy własne środowisko uruchomieniowe – dodaje do folderów wyszukiwania (sys.path) folder główny projektu, dlatego wskazujemy relatywną do gł. folderu ścieżkę

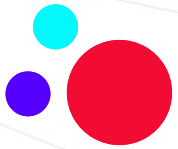
```
# -*- coding: utf-8 -*-
import sys
import os

print("Ścieżki wyszukiwania Python: ", sys.path)
print("Aktualny folder roboczy: ", os.getcwd())
```

Moduły

UWAGA!

Aby rozwiązać problem z importami (jeśli są) możemy:

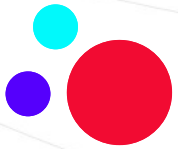
- *umieszczać importowany moduł w tym samym folderze co plik, do którego importujemy*
- *rozszerzyć sys.path –* ***sys.path.append****(”moja_sciezka”)*
- *moduły wrzucać do folderu, którego ścieżkę dodajemy w zmiennej środowiskowej*
  - *PYTHONPATH (na poziomie systemu!)*
- *moduł umieścić w folderze bibliotek standardowych Python*
  - *(folder lib\site-packages\) w instalacji Pythona*



Moduły - importowanie

Sposoby importowania modułów:

```
import modul

import podkatalog.modul

import podkatalog.modul as nowa_nazwa_modulu

from podkatalog.modul import funkcja as nowa_nazwa_funkcji

from podkatalog.modul import *
```

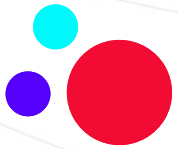


Moduły - PyPI (Python Package Index)

Ktoś już wykonał za nas robotę :)

pypi.org

infoShare ACADEMY

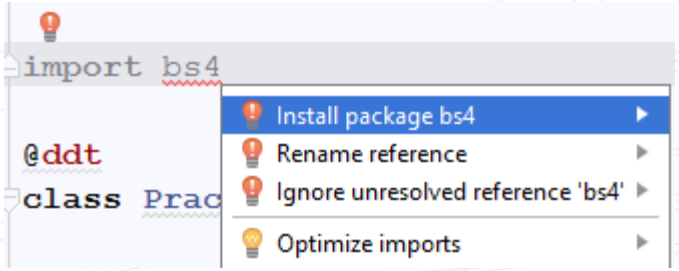


Moduły – pip

Menadżer pakietów instalowany razem z Python. Komendy w wierszu poleceń:

- ***pip help*** *– ogólna pomoc*
- ***pip help install*** *– pomoc dot. polecenia*
- ***pip list*** *– lista zainstalowanych pakietów*
- ***pip search*** *– szuka pakietów w repozytorium online*
- ***pip install pakiet*** *– instalowanie modułu*
- ***pip uninstall pakiet -*** *odinstalowanie*
- ***pip list –o -*** *sprawdzenie nieaktualnych pakietów*
- ***pip install –U pakiet -*** *update pakietu*
- ***pip freeze > plik.txt*** *– zapisanie informacji do pliku o pakietach*
- ***pip install –r plik.txt*** *– zainstaluje wszystkie wymagane pakiety*

Moduły – Install via PyCharm



- *Alt + Enter*

Moduły – przykłady

*Moduł **os** służy do pracy z plikami,*

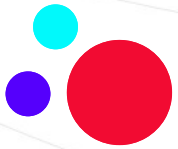
ścieżkami, zmiennymi systemowymi, np.:

- *mkdir*
- *chdir*
- *getcwd*
- *unlink*
- *rmdir*
- *listdir*
- *walk*

Moduł **csv** praca na plikach CSV

Moduł **openpyxl** praca na plikach Excel

Moduł **send2trash** przenosi dane do kosza
(zgodny ze wszystkimi systemami operacyjnymi)



Pliki – tworzenie

plik = open(”ścieżka_do_pliku”, tryb)

tryby tekstowe:

r *tylko do odczytu*

w *zapisywanie pliku (stary plik o tej samej nazwie zostanie usunięty)*

r+ *do odczytu i zapisu*

a *dopisywanie do pliku (dopisywane do końca istniejącego pliku)*

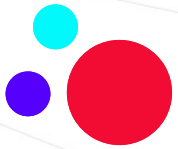
tryby binarne:

rb *tylko do odczytu*

wb *zapisywanie pliku (stary plik o tej samej nazwie zostanie usunięty)*

rb+ *do odczytu i zapisu*

ab *dopisywanie do pliku (dopisywane do końca istniejącego pliku)*



Pliki – odczytywanie

***plik*.*read*(*[int]*)**

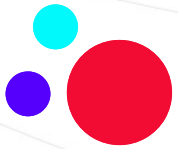
```
# odczytanie całego pliku, zwracany jest string zawierający cały tekst pliku
  włącznie ze znakami \n;
# int - określająca ilość bajtów do wczytania
```

***plik*.*readline*()**

```
# odczytanie jednej linii z pliku, zwracany jest string z linijką testu,
  włącznie ze znakiem \n
```

***plik*.*readlines*()**

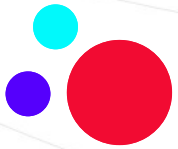
```
# odczytuje cały tekst – zwraca listę stringów - linijek
```



Pliki – uwaga na białe znaki

```
for line in plik:
    print(line)
```

```
for line in plik:
    print(line, end='')
```



Pliki – zapisywanie

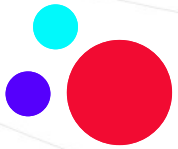
plik*.write(string*)

```
# zapisuje string do pliku w obecnej pozycji kursora
# zwraca liczbę zapisanych znaków
# należy pamiętać o znaku \n
```

plik*.writelines(iterable*)

```
# zapisuje elementy z typu iteracyjnego jako poszczególne linie w pliku
# należy pamiętać o znaku \n
```

Plik musi być otworzony w trybie do zapisu aby móc go zmieniać!



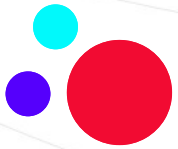
Pliki – zamykanie pliku

Pliki należy zamykać po użyciu.

```
plik = open(”ścieżka_do_pliku”)
# kod
plik.close()
```

Otwarcie pliku za pomocą **with** pozwala na automatyczne zamykanie pliku przez Pythona

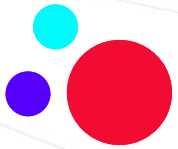
```
with open(”ścieżka_do_pliku”) as plik:
    # kod
```



Pliki – CSV

```
Imie,Nazwisko,Adres,Telefon
Joanna,Kowalska,Gdansk Przytulna,64 654-65-45
Adam,Nowak,Gdynia Swietojanska,0700325487
```

```
import csv

with open('adresy.csv', newline='') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
                print(row)
    writer = csv.writer(csvfile)
    writer.writerow(['Jan', 'Kowalski', 'Sopot', '123-432-111'])
```



Pliki – pickle

moduł służący do zapisywania obiektów do plików.

zapisać i odczytać możemy każdy obiekt Python'a:

listy z danymi, słowniki, klasy, instancje klasy itd.

zapis w **trybie binarnym**

```
import pickle

dane = ["Bartosz", "Mojo", 33]

with open("ogorek.pickle", "wb") as plik:
    pickle.dump(dane, plik)

# odczytanie
with open("ogorek.pickle", "rb") as plik:
    dane_wczytane = pickle.load(plik)

print(dane_wczytane)
```

Wysyłka maili

Przydatne moduły:

- **smtplib** – Simple Mail Transfer Protocole
- **imaplib** – obsługa poczty IMAP
- **email.mime.MimeText** – format przesyłania informacji MIME

https://docs.python.org/3/library/smtplib.html#module-smtplib

https://docs.python.org/3.1/library/email-examples.html

Wysyłka maili – serwer pocztowy

login: pythonintel@int.pl

hasło: isapython;2025

smtp: poczta.int.pl

port: 465

zabezpieczenie: SSL

Wysyłka maili – algorytm

1)importuje biblioteki

2)przygotowuję maila z tematem treścią i odbiorcą

3)tworze obiekt mailera

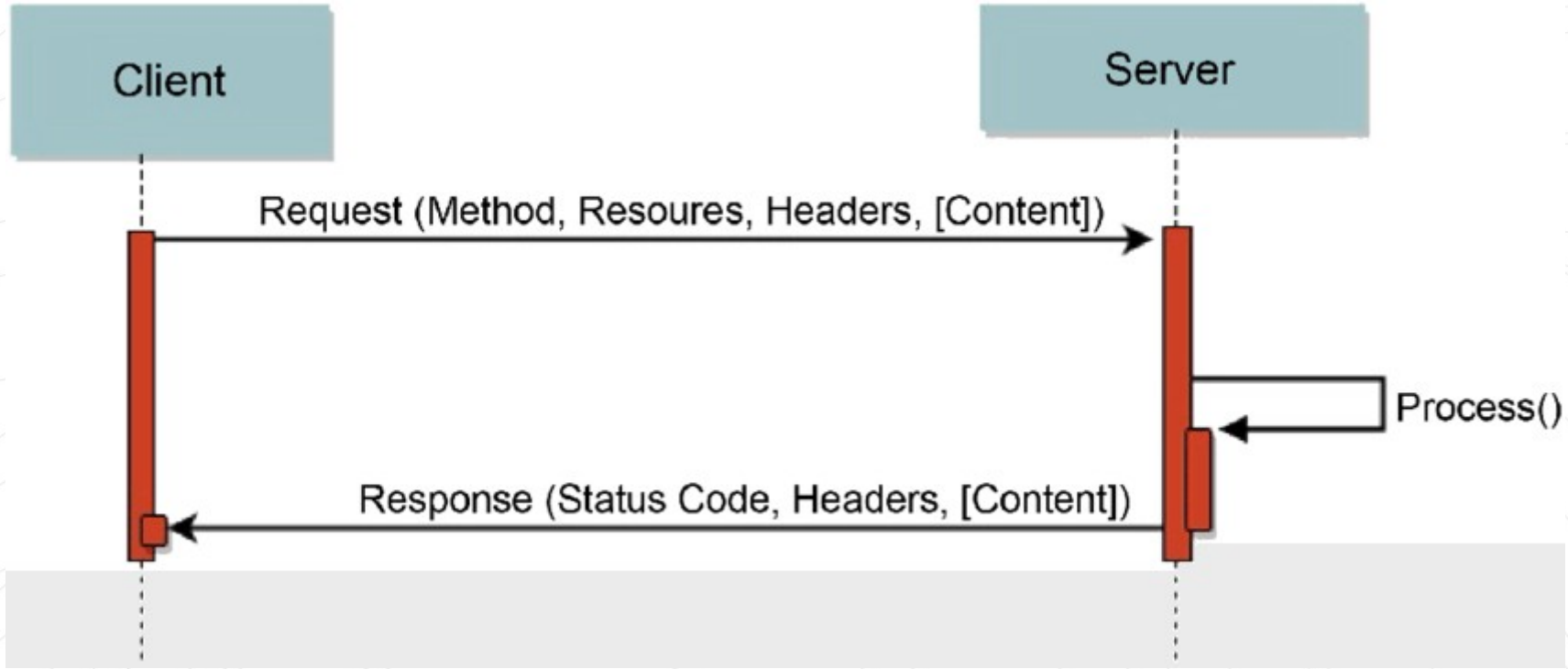
   1)witam się z serwerem smtp – tworzę połączenie

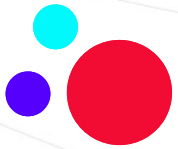
   2)loguję się (podając login i hasło)

   3)wysyłam maila

4)kończę połączenie z serwerem

HTTP





HTTP – moduł requests

Moduł do pracy z żądaniami HTTP(S)

```
import requests

site = requests.get('https://wp.pl')
```

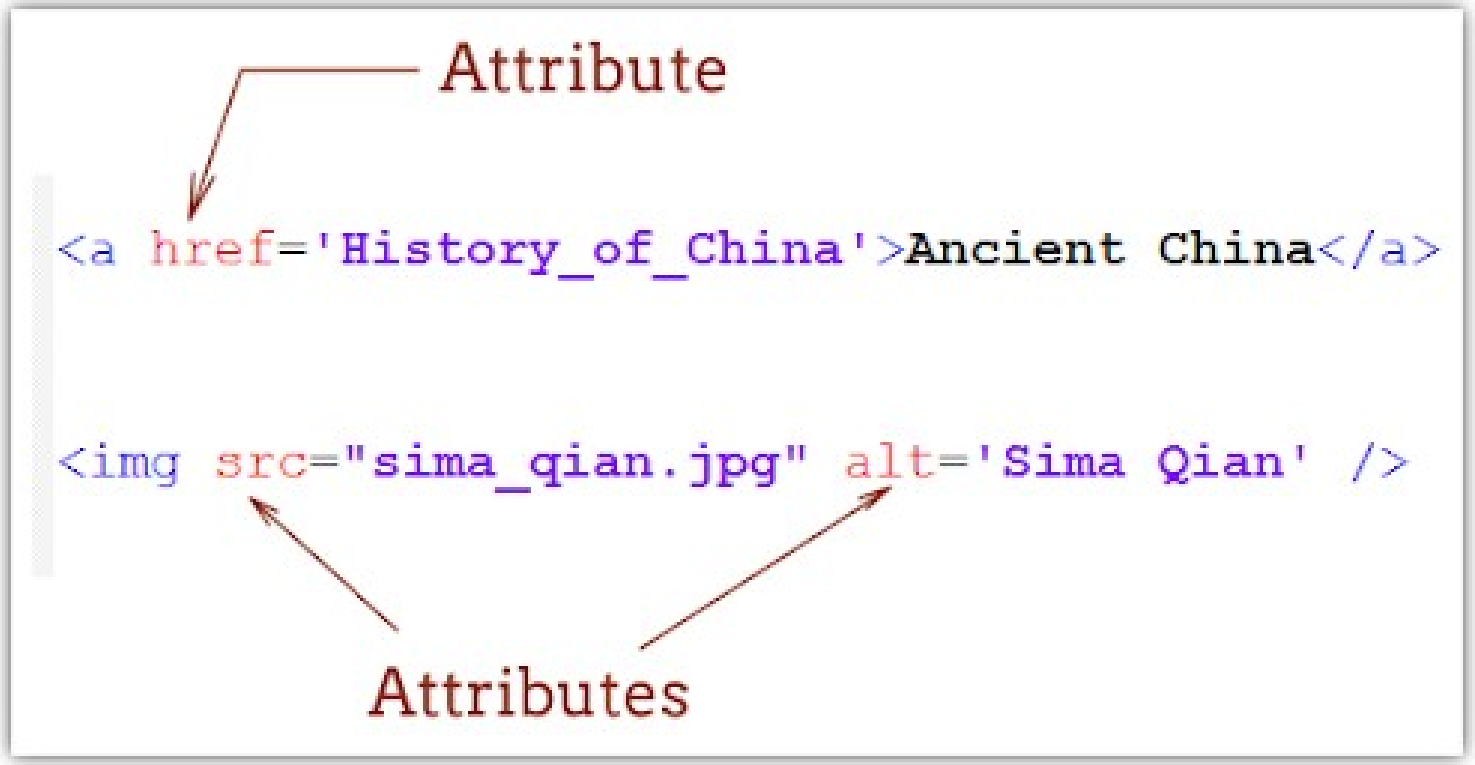
Niektóre z metod i właściwości:

site**.json()**

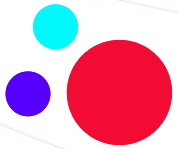
site**.text**

site**.content**

site**.status_code**


Attribute
Ancient China

Attributes



HTML – moduł BeautifulSoup

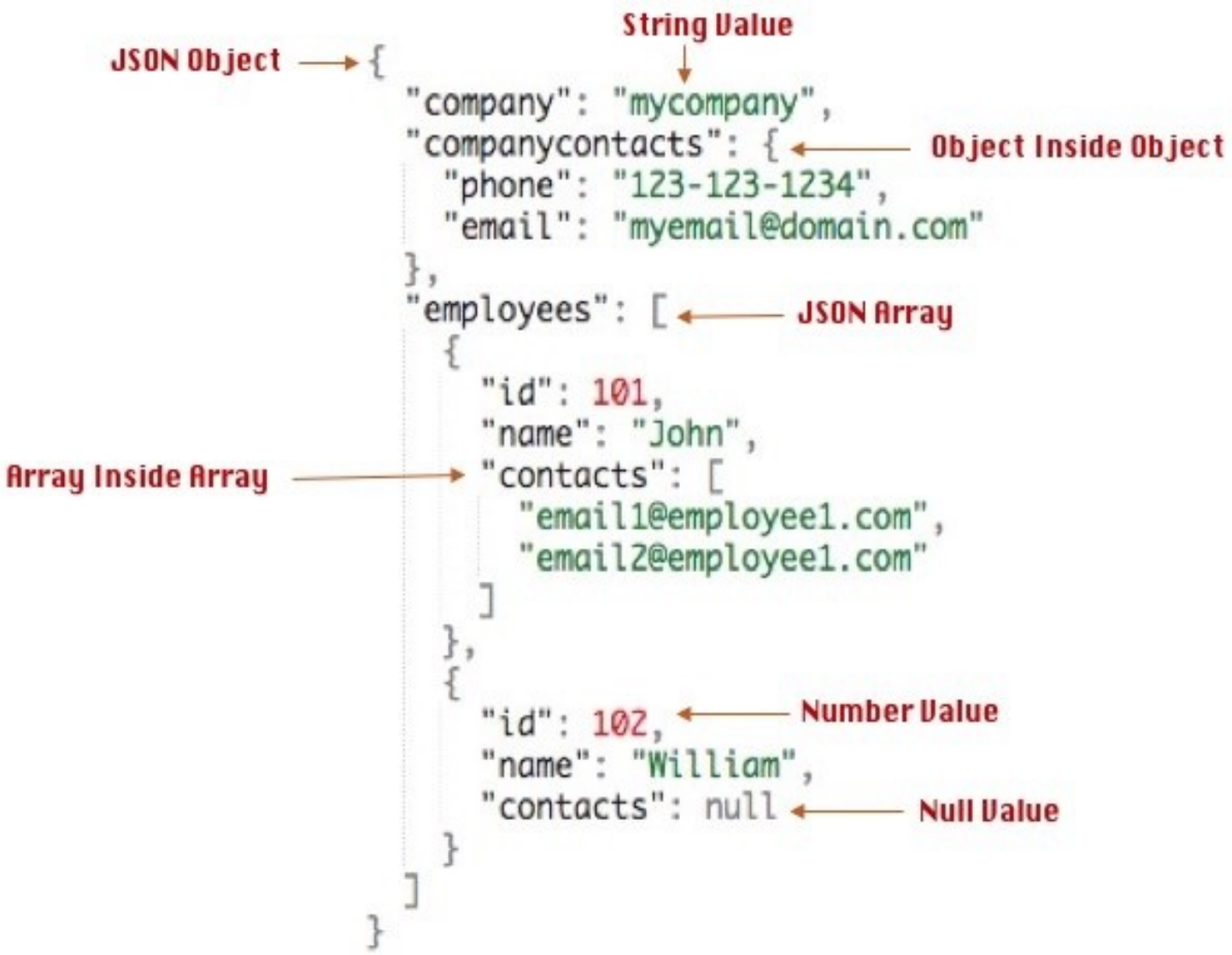
Moduł do pracy z plikami HTML, XML

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(html_doc, 'html.parser')
```

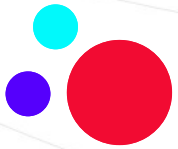
Niektóre z metod:

get(), find(), find_all()

JSON



infoShare ACADEMY



JSON – moduł json

Moduł do pracy z danymi typu JSON

```
import json
```

Niektóre z metod:

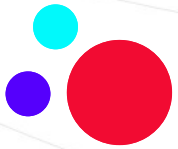
json.loads()

json.load()

json.dumps()

json.dump()

infoShare ACADEMY



Pliki – Excel

```
# moduł służący do operacji na plikach excel
# aby operować na danych należy wybrać konkretny arkusz i komórkę

import openpyxl

excel = openpyxl.load_workbook('test.xlsx') # .workbook.Workbook()

# arkusz = excel.active

# arkusze = excel.sheetnames

arkusz = excel['Arkusz1']

komorka = arkusz.cell(row = 1, column = 1)

print(komorka.value)

komorka.value = 'Nowa wartość'

excel.save('plik.xlsx')
```

Praca z grafiką

Moduł do pracy ze zdjęciami

```
from PIL import Image

foto = Image.open(plik)
```

Niektóre z metod:

copy(), crop(), filter(), paste(), resize(), rotate(), save()

https://pillow.readthedocs.io/en/4.0.x/
https://pillow.readthedocs.io/en/4.0.x/reference/Image.html

infoShare ACADEMY

DZIĘKUJĘ NA DZIŚ

Python Podstawy - Intel

infoShareAcademy.com