



O prowadzącym

Kamil Musiał

Tester w Tieto (wcześniej 7 lat tester / integrator /
verification specialist w Nokia)



certyfikowany tester ISTQB
(full advanced)



od 5 lat wykładowca WSB Wrocław
(testowanie, telekomunikacja, sieci, IoT, Python)



doktorant
Politechnika Wrocławska



fan morsowania
zanim to było modne



kamil.musial@wsb.wroclaw.pl

kamil.musial@chorzow.wsb.pl

O prowadzącej

Natalia Kniat

CI Lead / testerka / integratorka
w Nokia



certyfikowana testerka
ISTQB



trenerka Nokia Academy
(testowanie, telekomunikacja, Python)



zorganizowana mama
z zamiłowaniem do wyrobów własnych



natalia.kniat@wsb.wroclaw.pl

Tester jako developer narzędzi z pomocą Python-a - warsztaty (8h)



01

Biblioteka
os oraz time

02

Biblioteka
datetime

03

Obsługa plików
csv bez bibliotek

04

Zadanie: znaleźć
nazwę hosta i IP

05

Powtórka:
operacje
na stringach

06

Obsługa plików
txt bez bibliotek

Tester jako developer narzędzi z pomocą Python-a - warsztaty (8h)



07

Zadanie:
Tracert: jak
daleko do googla?

08

Zadanie:
Search and destroy

09

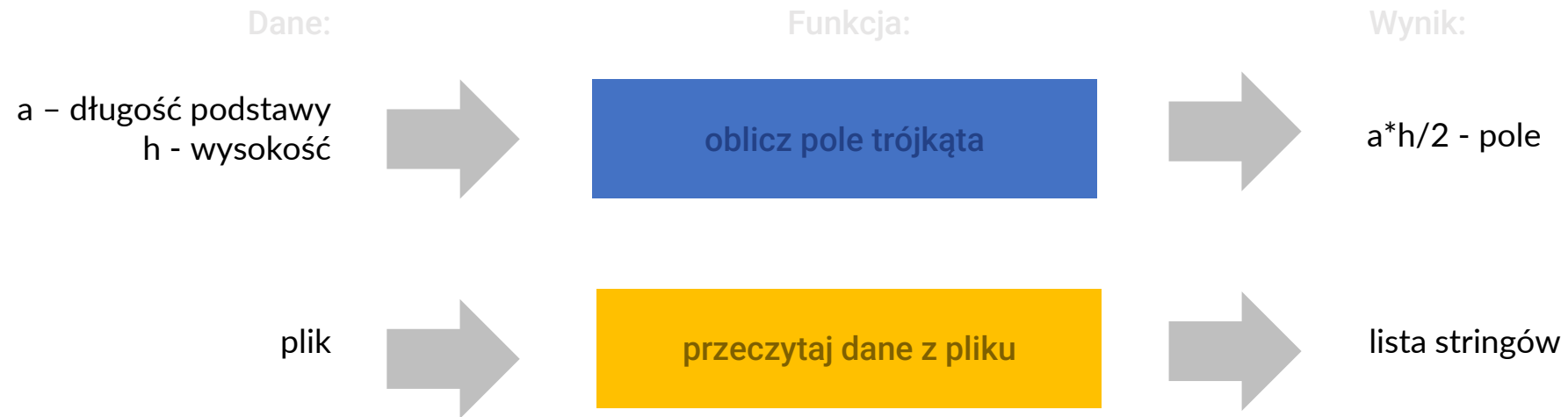
Zadanie:
Podmiana danych

10

Biblioteka Pandas:
obsługa plików csv


Python: funkcje

Funkcja – część programu, zgrupowany kod programu odpowiadający za pewną funkcjonalność
(możliwość wielokrotnego użycia)



Biblioteka os oraz time

Przykłady użycia biblioteki OS



```
import os
import time
```

`print(os.getcwd())` sprawdzenie aktualnej lokalizacji

`os.chdir('C:\\Users\\kamil\\Desktop')`
zmiana lokalizacji

`os.mkdir('new_folder')` utworzenie nowego folderu

`os.rename('new_folder', 'new_folder2')`
zmiana nazwy folderu

`os.rmdir('new_folder2')` usunięcie folderu

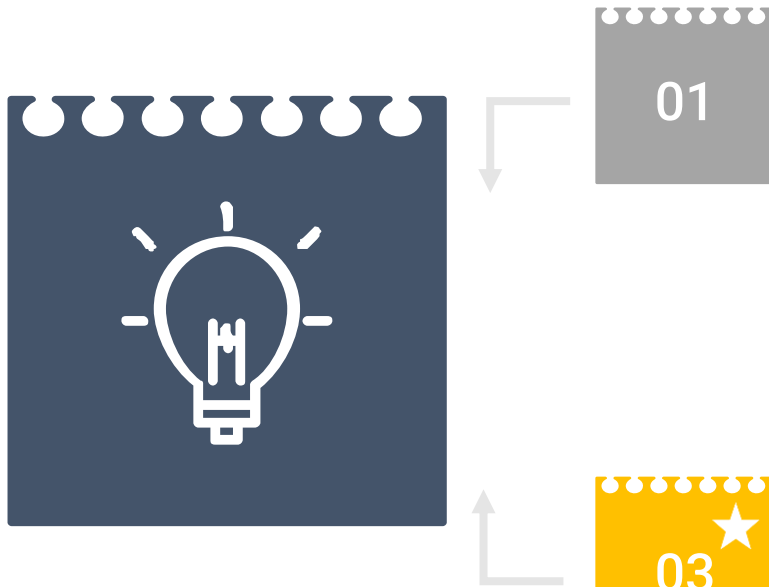
`os.system('cmd /c "cd C://Users//kamil//Desktop")`
użycie konsoli systemowej

`time.sleep(5)` odczekanie 5 sekund

`&&` łączenie dwóch komend w terminalu

Biblioteka os oraz time

Zadania Python *



- przejdź na pulpit
- utwórz folder i poczekaj kilka sekund
- zmień nazwę folderu i poczekaj kilka sekund
- usuń folder

Wyszukaj na pulpicie pliki `new.txt` i zwróć rezultat wyszukiwań w pliku `result.txt`

Rezultat:

Po uruchomieniu programu na pulpicie pojawia się folder, za kilka sekund folder zmienia nazwę i za kolejne kilka sekund folder ten znika.

Na pulpicie pojawia się plik `results.txt` z wynikami wyszukiwania pliku `new.txt`

* PyCharm - pliki z kodem dostępne osobno

Operacje na stringach

Powtórka



Łączenie
stringów

```
my_string1 = 'mom'
```

```
my_string2 = 'dad'
```

```
sum1 = my_string1 + my_string2
```

```
print(sum1)      -> momdad
```

```
sum2 = 'my' + my_string1 + ' and my ' + my_string2
```


```
print(sum2)      -> my mom and my dad
```

Operacje na poszczególnych elementach:

```
print(sum2[3:6]) -> mom
```

```
print(sum2[-4:-1]) -> dad
```

Biblioteka datetime



Jaki dzisiaj
dzień?

```
import datetime
```

wyciągnięcie daty:

```
today = datetime.date.today()
```

%d dzień

%m miesiąc

%Y / **%y** rok (pełny, skrócony)


%B / **%b** miesiąc słownie (pełny, skrócony)

Różne sposoby przedstawienia daty:

```
d1 = today.strftime('%d/%m/%Y') dd/mm/YY
```

```
d2 = today.strftime('%B %d, %Y') tekstowo miesiąc, potem dzień i rok
```

Biblioteka datetime



Jaki dzisiaj
dzień?

wyciągnięcie godziny i daty:

```
now = datetime.datetime.now()
```

%d dzień

%m miesiąc

%Y / %y rok (pełny, skrócony)

%B / %b miesiąc słownie (pełny, skrócony)

%H godzina

%M minuta

%S sekunda

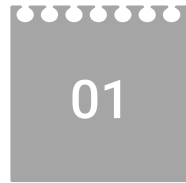
Tworzenie stringa z datą bądź godziną:

```
timer = now.strftime('%H:%M:%S')
```

```
date = now.strftime('year = %Y, month = %m, day = %d')
```

Biblioteka datetime

Zadania Python *



Wypisać 10 nazw (plików)
w odstępach 2 sekundowych o nazwach:
report<godzina>.txt

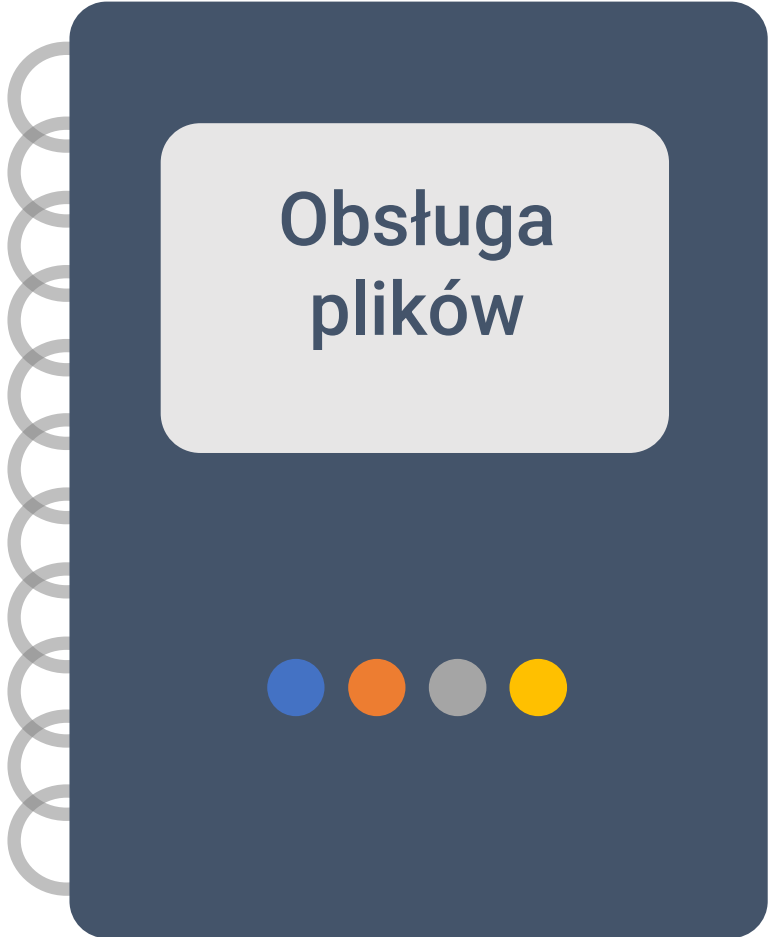
Wynik:

```
report012512.txt  
report012514.txt  
report012516.txt  
report012518.txt
```

Każdy plik stworzony o godzinie 01:25 i co 2 sekundy.

* PyCharm - pliki z kodem dostępne osobno

Obsługa plików txt bez bibliotek



Obsługa plików

Metody:

`read()` odczyt całego pliku jako string

`read(5)` odczyt 5 znaków (pierwszych bądź kolejnych)

`readline()` odczyt linii (pierwszej bądź kolejnej)

`readlines()` odczyt całego pliku jako tablicy (jedna linia, jeden element)

Przykład:

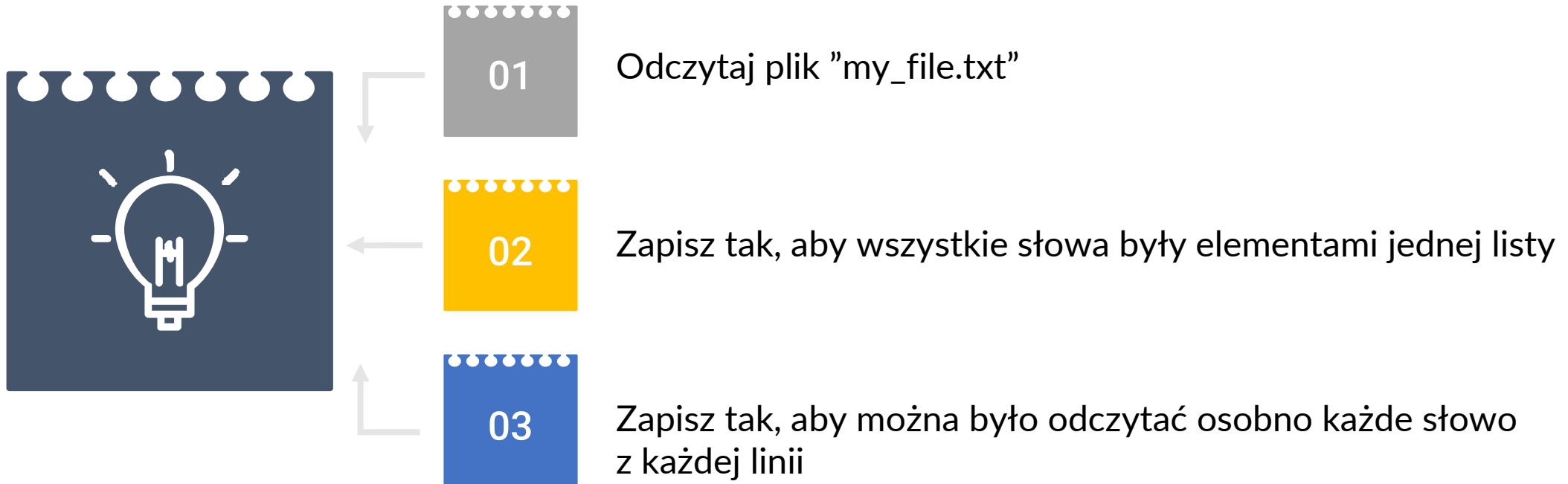
```
with open ("C://Users//kamil/Desktop//my_file.txt",'r') as plik1:  
    content = plik1.readlines()
```

gdzie:

`r` - ,read' tryb do odczytu (inne tryby: `w` - write, `a` - append)

Obsługa plików txt bez bibliotek

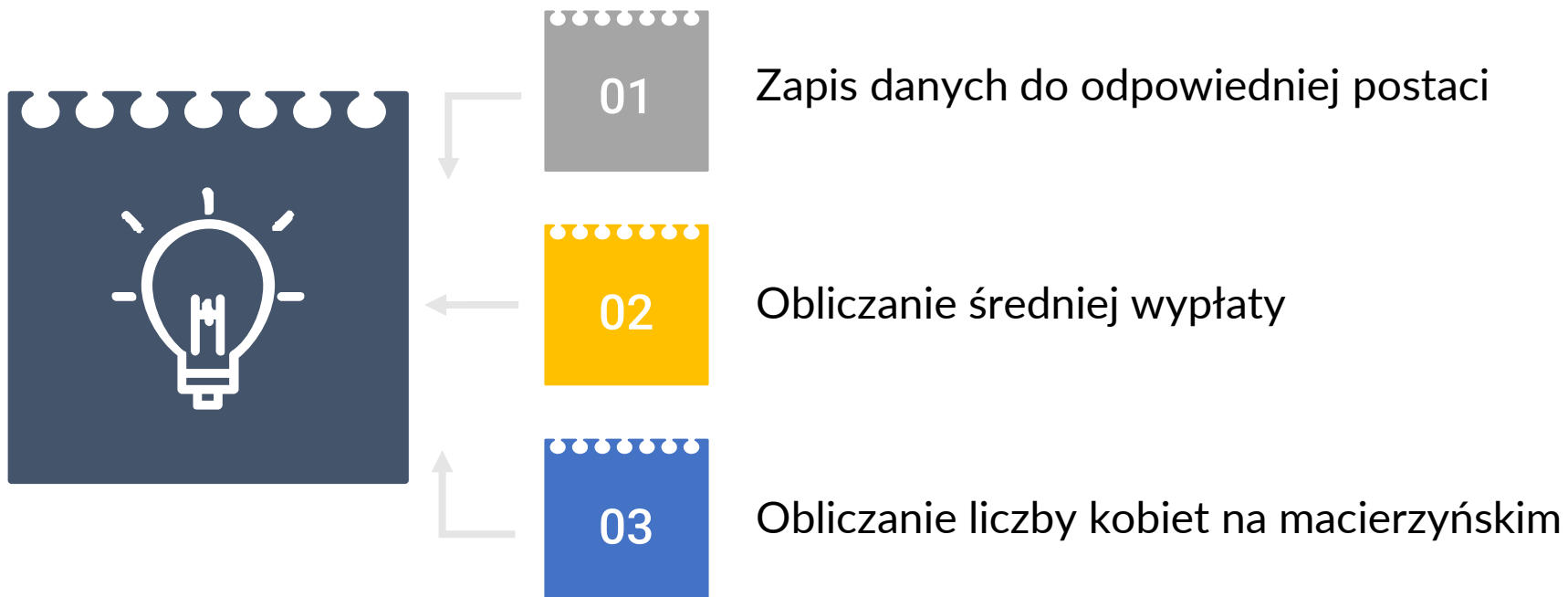
Zadania Python *



* PyCharm - pliki z kodem dostępne osobno

Obsługa plików csv bez bibliotek

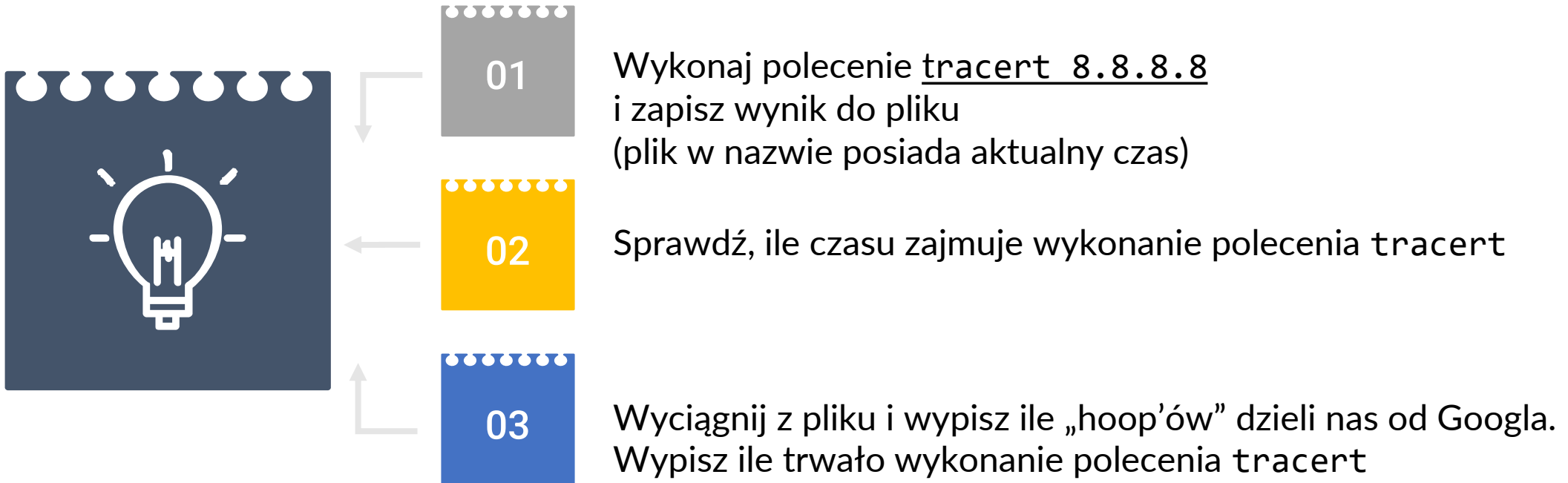
Zadania Python *



* PyCharm - pliki z kodem dostępne osobno

Zadanie: Tracert – jak daleko od Googla?

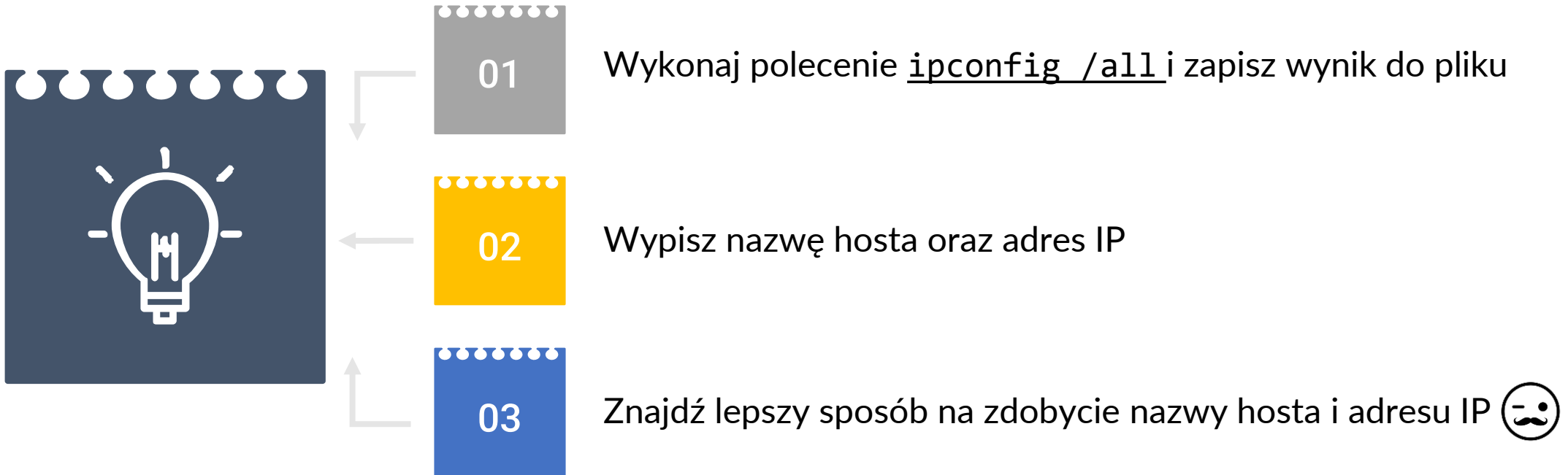
Zadania Python *



* PyCharm - pliki z kodem dostępne osobno

Zadanie: znaleźć nazwę hosta i IP

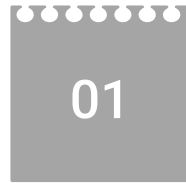
Zadania Python *



* PyCharm - pliki z kodem dostępne osobno

Zadanie: Search and destroy

Zadania Python *

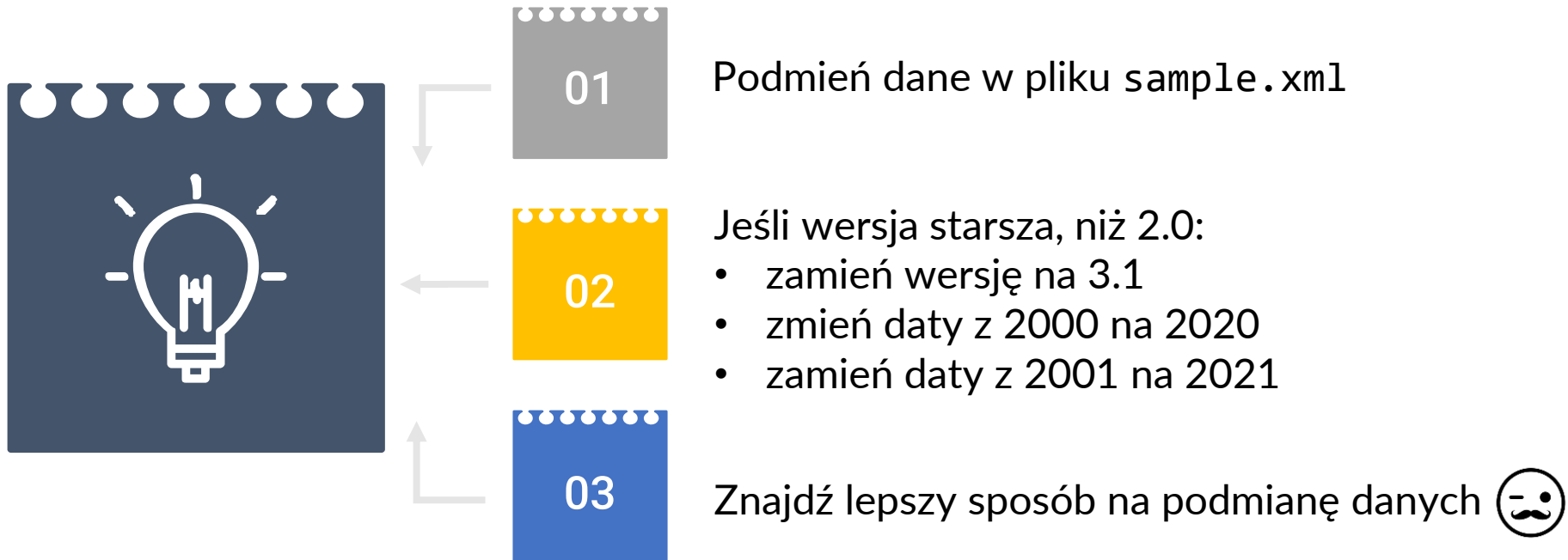


Usuń z pulpitu wszystkie pliki z rozszerzeniem "pcap"

* PyCharm - pliki z kodem dostępne osobno

Zadanie: Podmiana danych

Zadania Python *



* PyCharm - pliki z kodem dostępne osobno

Obsługa plików biblioteka Pandas

Środowisko – doinstalowanie biblioteki Pandas oraz ffspec



Biblioteka
Pandas

Import biblioteki i nadanie jej nazwy roboczej pd:

```
import pandas as pd
```

Przykład:

Zapis pliku csv z przecinkiem jako separatorem pod nazwą data:

```
data = pd.read_csv(path, delimiter=',')
```

Plik

pokemons.csv

Obsługa plików biblioteka Pandas

Przykłady użycia biblioteki Pandas



Biblioteka
Pandas

`print(data.head(5))` wyświetlenie 5 pierwszych wierszy

`print(data.columns)` wyświetlenie kolumn

`print(data[['Name', 'Type 1', 'HP']][0:5])`
wyświetlenie pięciu pierwszych
wierszy i kolumn Name, Type 1 i HP

`print(data.iloc[1:4])` wyświetlenie wierszy 2, 3, 4

`print (data.iloc[2,1])` wyświetlenie 2go pola w 3cim wierszu

`print (data.loc[data['Type 1'] == 'Fire'])`
wyświetlenie pozycji, posiadających Fire w Type 1

`print(data.sort_values('Name'))`
sortowanie rosnąco po Name

`print(data.sort_values('Name', ascending=False))`
posortowanie malejąco po Name

Obsługa plików biblioteka Pandas

Przykłady użycia biblioteki Pandas



Biblioteka
Pandas

Zapis danych do csv:

```
data.to_csv('C:\\Users\\kamusia\\Desktop\\modified.csv', sep=',';')
```

lub

```
data.to_csv('C:\\Users\\kamusia\\Desktop\\modified.csv',  
index=False, sep=';')
```

Zapis danych do txt:

```
data.to_csv('C:\\Users\\kamusia\\Desktop\\modified.txt', sep='\\t')
```

lub

```
data.to_csv('C:\\Users\\kamusia\\Desktop\\modified.txt',  
index=False, sep='\\t')
```

`index=False`

usuwa pierwszą kolumnę z indeksami

Wykresy - pyplot

Przykłady użycia biblioteki matplotlib



Biblioteka
Matplotlib

- Przykład:
- `import matplotlib.pyplot as plt`
- `numbers = [5,10,15,3,20]` # kolejne dane
- `plt.plot(numbers)` # oś x to indeksy
- `plt.show()`

Wykresy - pyplot

Przykłady użycia biblioteki matplotlib



Biblioteka
Matplotlib

Można określić rodzaj punktów, kolor linii i punktów oraz kształt linii:

- `plt.plot(numbers, 'o')` # niepołączone punkty
- `plt.plot(numbers, '.')` # niepołączone kropki
- `plt.plot(numbers, 's')` # niepołączone kwadraty
- `plt.plot(numbers, 'ro')` # czerwone punkty
- `plt.plot(numbers, 'g^')` # zielone trójkąty
- `plt.plot(numbers, 'k*')` # czarne gwiazdki
- `plt.plot(numbers, 'r-')` # punkty połączone czerwonymi liniami
- `plt.plot(numbers, 'bD:')` # niebieskie diamenty połączone kropkami
- `plt.plot(numbers, 'mp--')` # różowe pięciokąty - przerywane linie

Wykresy - pyplot

Przykłady użycia biblioteki matplotlib



Biblioteka Matplotlib

- Przykład:
- `import matplotlib.pyplot as plt`
- `X = [i+1 for i in range(-10,10)]`
- `Y1 = [6*i - 1 for i in X]`
- `Y2 = [-4*i + 3 for i in X]`
- `Y3 = [3*i + 4 for i in X]`
- `plt.axhline()` # linia pozioma osi
- `plt.axvline()` # linia pionowa osi
- `plt.plot(X,Y1,'ro-')`
- `plt.plot(X,Y2,'b^--')`
- `plt.plot(X,Y3,'gs-')`
- `plt.xlabel("punkty x")` # opis osi X
- `plt.ylabel("wartosci y")` # opis osi Y
- `plt.title("Wykresy funkcji")` # tytuł wykresu
- `plt.show()`

Wykresy - pyplot

Przykłady użycia biblioteki matplotlib – wykres słupkowy



Biblioteka
Matplotlib

- `import matplotlib.pyplot as plt import random`
- `names = ["Ania", "Kasia", "Pawel", "Mariusz", "Roman"]`
- `ages = [random.randint(18,70) for name in names]`
- `plt.bar(names, ages, color=['red', 'green', 'blue'])`
- `plt.xticks(names) plt.yticks(ages) plt.show()`

Wykresy - pyplot

Przykłady użycia biblioteki matplotlib – wykres kołowy



Biblioteka
Matplotlib

- `import matplotlib.pyplot as plt`
- `expenses = ["mieszkanie", "media", "jedzenie", "rozrywka", "transport", "inne"]`
- `values = [3000, 300, 1000, 500, 200, 1500] plt.pie(values, labels=expenses, autopct=False, shadow=True, explode=(0.1, 0.1, 0.1, 0.1, 0.1, 0.1), plt.show()`

Wykresy - pyplot

Przykłady użycia biblioteki matplotlib – kilka wykresów



Biblioteka
Matplotlib

- **Kilka wykresów w jednym oknie (subplot)**
- Przykład - dwa wykresy sąsiadujące poziomo:
- `import matplotlib.pyplot as plt`
- `import random`
- `import math`
- `X = [x for x in range(0, 360+1, 10)]`
- `Y1 = [math.cos(math.radians(i)) for i in X]`
- `Y2 = [random.random() for i in X]`
- `plt.subplot(1,2,1) # poziomy,piony,index`
- `plt.plot(X,Y1,"r.-")`
- `plt.subplot(1,2,2) # poziomy,piony,index`
- `plt.plot(X,Y2,"bs:")`
- `plt.show()`

Obliczenia - NumPy

Przykłady użycia biblioteki NumPy



Biblioteka
NumPy

```
Import numpy as np
```

```
a = np.array([1, 2, 5, 20, 40])
```

```
print(a)
```

- odwołanie do elementu: `a[1]` -> 2,
- slice (wycinek tablicy): `a[1:3]` -> [2 , 5]
- zastąpienie elementu: `a[1] = 50` -> [1, 50, 5, 20, 40]
- długość: `len(a)` -> 5
- iteracja po tablicy jednowymiarowej: `for i in a: print(a)`
- **Elementy w ndarray muszą być tego samego typu!**

Obliczenia - NumPy

Przykłady użycia biblioteki NumPy



Biblioteka
NumPy

- **`np.zeros(N)`** -> tablica o długości N wypełniona zerami
- **`np.ones(N)`** -> tablica o długości N wypełniona jedynkami
- **`np.empty(N)`** -> tablica o długości N z "zerowymi" wartościami
- **`np.linspace(1,10,5)`** -> 5 elementów z zakresu od 1 do 10 z równym krokiem
- **`np.arange(0, 0.5, 0.1)`** -> zakres od 0 do 0.5 z krokiem 0.1 (uwaga! standardowe **`range`** działa tylko dla liczb całkowitych!)
- **`np.random.randint(-10, 10, size=5)`** -> tablica 5 losowych elementów od -10 do 10

Obliczenia - NumPy

Przykłady użycia biblioteki NumPy



Biblioteka
NumPy

- Numpy pozwala na proste tworzenie tablic wielowymiarowych:
- 0-wymiarowa (skalar): `x = np.array(10)`
- 1-wymiarowa (wektor): `a = np.array([1,5,10,15])`
- 2-wymiarowa (macierz): `m = np.array([[2,3,4], [5,6,7]])`
- 3-wymiarowa: `t = np.array([[[2,3,4],[5,6,7]],
[[12,13,14],[15,16,17]]])`
- Sprawdzenie wymiaru tablicy: `t.ndim`, rozmiar tablicy: `t.shape`

Obliczenia - NumPy

Przykłady użycia biblioteki NumPy



Biblioteka
NumPy

- **Operacje na elementach tablicy**
- Operatory dodawania, odejmowania, mnożenia itp. wykonywane na obiektach ndarray powodują wykonanie operacji na konkretnych elementach tablic (w odróżnieniu np. od dodawania list!):
- `a = np.array([1,2,5,11,20])`
- `b = np.array([4,5,6,10,40])`
- `a + b -> [5, 7, 11, 21, 60]`
- `a - b -> [-3, -3, -1, 1, -20]`
- `a * b -> [4, 10, 30, 110, 800]`
- Możemy też wykonywać inne operacje na każdym elemencie:
- `a + 100 -> [101, 102, 105, 111, 120]`
- `a * 10 -> [10, 20, 50, 110, 200]`

Obliczenia - NumPy

Przykłady użycia biblioteki NumPy



Biblioteka
NumPy

Zaokrąglania:

- **`np.around(a, n)`** - zaokrąglenie elementów do n liczb po przecinku **`np.floor(a)`** - zaokrąglenie elementów w dół
- **`np.ceil(a)`** - zaokrąglenie elementów w górę

np.

- **`a = np.array([-5.5, -1, 0, 0.2, 10.123])`** **`np.around(a, 2)`** -> **`[-5.5 -1. 0. 0.2 10.12]`** **`np.floor(a)`** -> **`[-6. -1. 0. 0. 10.]`** **`np.ceil(a)`** -> **`[-5. -1. 0. 1. 11.]`**

Obliczenia - NumPy

Przykłady użycia biblioteki NumPy

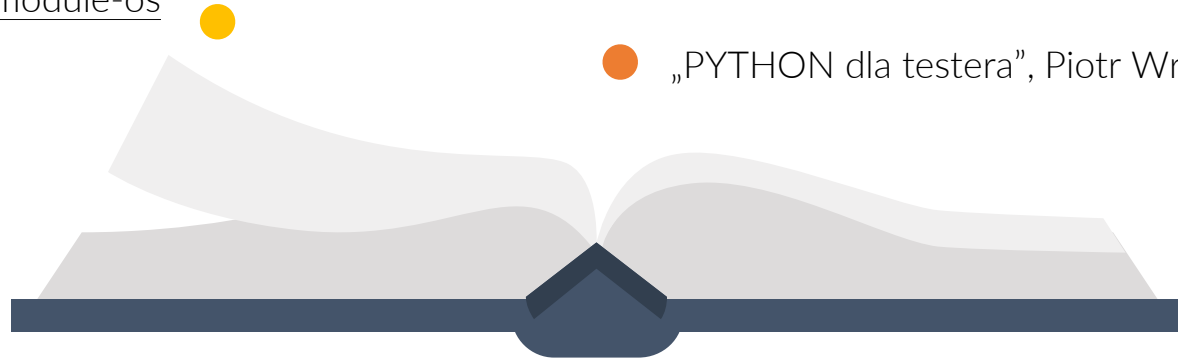


Biblioteka NumPy

- **Funkcje statystyczne**
- **`np.sum(a)`** - suma elementów tablicy **`a`** **`np.prod(a)`** - iloczyn elementów tablicy **`a`** **`np.mean(a)`** - średnia elementów tablicy **`a`** **`np.median(a)`** - mediana elementów tablicy **`a`**
- **`np.std(a)`** - odchylenie standardowe elementów tablicy **`a`** **`np.var(a)`** - wariancja elementów tablicy **`a`**
- **`np.min(a)`**, **`np.max(a)`** - minimum i maksimum z elementów tablicy **`a`** **`np.argmin(a)`**, **`np.argmax(a)`** - indeks wartości minimalnej i maksymalnej w tablicy **`a`**
- **Rozkłady statystyczne**
- NumPy udostępnia również moduł **`random`** - definiuje wiele funkcji pozwalających na generowanie zbioru
- danych według konkretnego rozkładu statystycznego, np. rozkład Gaussa (dla danego **`mu`** i **`sigma`**):
- ```
import numpy as np
```
- ```
mu = 3
```
- ```
sigma = 0.5
```
- ```
N = 10
```
- ```
X = np.random.normal(mu, sigma, N) print(np.mean(X), np.median(X), np.std(X), np.var(X), np.min(X), np.max(X))
```

# Materiały

Biblioteka os:  
<https://docs.python.org/3/library/os.html?highlight=os#module-os>



● „PYTHON dla testera”, Piotr Wróblewski

● „Python 3. Projekty dla początkujących i pasjonatów”, Adam Jurkiewicz



## Pandas – csv

[https://pandas.pydata.org/docs/reference/api/pandas.read\\_csv.html](https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html)



## Pandas kurs

<https://www.youtube.com/watch?v=vmEHCJofslg>



## OS kurs

<https://www.youtube.com/watch?v=tJxcKyFMTGo>

# Koniec

**Dziękuję za  
uwagę**

