Wizualizacja danych - wykład 2

Operacje arytmetyczne

Operator	Opis	Składnia
+	Dodawanie	x + y
–	Odejmowanie	x – y
*	Mnożenie	x * y
/	Dzielenie	x / y
//	Dzielenie całkowite	x // y
%	Dzielenie modulo	x % y
**	Potęgowanie	x ** y

```
print(5+3)
```

```
## 8
```

```
print(4*5.2)
```

```
## 20.8
```

```
print(9-7)
```

```
## 2
```

```
print(25%7)
```

```
## 4
```

```
print(4/5)
```

```
## 0.8
```

```
print(4//5)
```

```
## 0
```

```
print(4/5.0)
```

```
## 0.8
```

```
print(4//5.0)
```

```
## 0.0
```

```
print(3**0)
```

```
## 1
```

```
print(0**0)
```

```
## 1
```

```
print(4/0)
```

```
ZeroDivisionError: division by zero
```

Przypisanie z operacją arytmetyczną

Lista zawiera wybrane operacje.

Inna nazwa to złożone operatory przypisania.

Operator	Zapis	Dłuższa wersja
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5

```
a = 5
a += 1
print(a)
```

```
## 6
```

```
a **= 2
print(a)
```

```
## 36
```

Operatory porównania

Operator	Znaczenie	Przykład
$>$	Większe niż	x $>$ y
$<$	Mniejsze niż	x $<$ y
$==$	Równe	x $==$ y
$!=$	Nie równa się	x $!=$ y
$>=$	Większe lub równe	x $>=$ y
$<=$	Mniejsze lub równe	x $<=$ y

Operatory logiczne

Operator	Znaczenie	Przykład
and	i	x and y
or	lub	x or y
not	negacja	not x

Instrukcje warunkowe

Składnia

```
if <expr>:
    <statement>
```

else

```
if <expr>:
    <statement(s)>
else:
    <statement(s)>
```

```
a = 5
if a > 0:
    print("Liczba dodatnia")
else:
    print("Liczna ujemna lub zero")
```

```
x = 0
y = 5
if x < y:
    print('yes1')

if y < x:
    print('yes2')

if x:
    print('yes3')

if y:
    print('yes4')

if x or y:
    print('yes5')

if x and y:
    print('yes6')
```

elif

```
if <expr>:
    <statement(s)>
elif <expr>:
    <statement(s)>
elif <expr>:
    <statement(s)>
    ...
else:
    <statement(s)>
```

```python
a = 5
if a > 0:
    print("Liczba dodatnia")
elif a == 0:
    print("Zero")
else:
    print("Liczna ujemna")
```

Zagnieżdżone instrukcje warunkowe:

```
if <expr>:
    <statement(s)>
    if <expr>:
        <statement(s)>
```

```
i = 21
if i > 0:
    print("liczba jest dodatnia")
    if i % 2 == 0:
        print("Liczba jest dodatkowo parzysta")
```

for - pętla

```
for i in <collection>:
    <loop body>
```

```
for i in range(5):
    print(i)
```

range

Generuje nam ciąg liczb (dedykowany typ `range`). Trzeba zamienić na listę "by podejrzeć".

Uwaga: wszystkie argumenty muszą być w typie całkowitym.

Jeden argument - to "koniec" - ciąg tworzą liczby naturalne od 0 do $n-1$. Krok domyślny to 1.

Dwa argumenty - to "początek" i "koniec". Krok domyślny to 1. Wtedy wyświetlone są liczby całkowite z przedziału lewostronnie domkniętego $[początek, koniec)$.

Trzy argumenty - to "początek", "koniec" oraz krok.

```
print(list(range(5)))
print(list(range(1, 11)))
print(list(range(0, 30, 5)))
print(list(range(0, 10, 3)))
print(list(range(0, -10, -1)))
print(list(range(0)))
print(list(range(1, 0)))
```

While - pętla

```
while <expr>:
    <statement(s)>
```

```
i = 0
while i < 5:
    print(i)
    i = i + 1
```

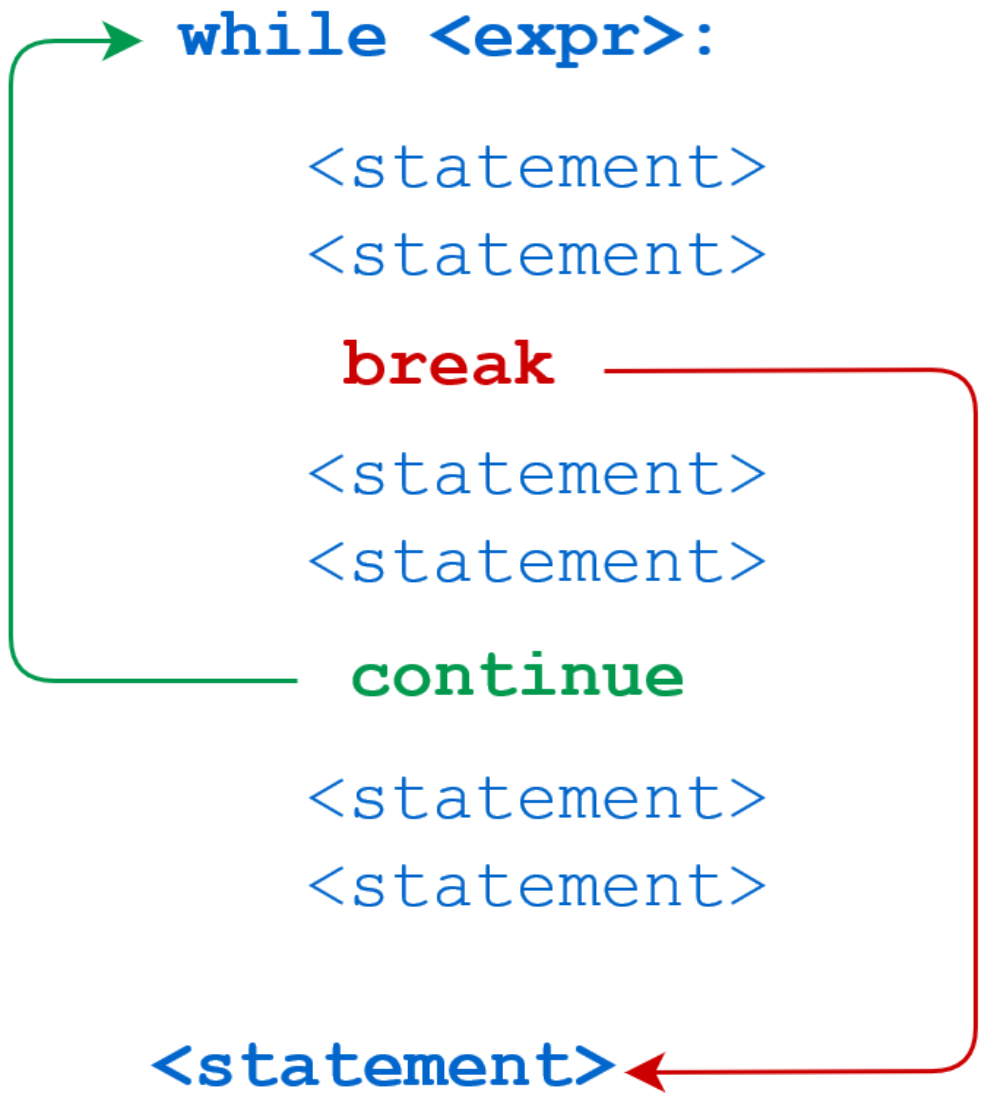
Zagnieżdżone pętle

```
for i in <collection>:
    <loop body>
    for i in <collection>:
        <loop body>
```

```
while <expr>:
    <statement(s)>
    while <expr>:
        <statement(s)>
```

```
for i in range(5):
    for j in range(5):
        print(i, "*", j, "=", i * j)
```

break/continue



break

```
n = 5
while n > 0:
    n -= 1
    if n == 2:
        break
    print(n)
```

continue

```
n = 5
while n > 0:
    n -= 1
    if n == 2:
        continue
    print(n)
```

Kolejność operatorów

Od ostatniego:

- `lambda`
- `if - else`
- `or`
- `and`
- `not x`
- `in, not in, is, is not, <, <=, >, >=, !=, ==`
- `|`
- `^`
- `&`
- `<<, >>`
- `+, -`
- `*, @, /, //, %`
- `+x, -x, ~x`

- `**`
- `await x`
- `x[index]`, `x[index:index]`, `x(arguments...)`, `x.attribute`
- `(expressions...)`, `[expressions...]`, `{key: value...},{expressions...}`

Źródło: https://docs.python.org/3/reference/expressions.html#operator-precedence.

Pytanie do przemyślenia

Co oznacza w Pythonie, że wartości przekazywane są przez referencję?

```
a = 5
b = a
b += 2
print(a)
```

```
## 5
```

```
print(b)
```

```
## 7
```

Listy

Lista w Pythonie to tzw. typ sekwencyjny umożliwia przechowywanie elementów różnych typów.

Cechy:

- zmienny (`mutable`) - umożliwia przypisanie wartości pojedynczym elementom
- do zapisu używamy nawiasów kwadratowych
- poszczególne elementy rozdzielamy przecinkami
- każdy element listy ma przyporządkowany indeks
- elementy listy są numerowane od zera
- listy są uporządkowane
- listy są dynamiczne (mogą mieć różną długość)
- listy mogą być zagnieżdżone

Uwaga!

Listy w języku Python są specyficzną strukturą danych nie zawsze dostępną w innych językach programowania. Pojęcie listy w całej informatyce "szersze". Wyróżnia się np. listy jednokierunkowe, które nie muszą mieć indeksu. Nie będziemy takich przypadków analizować.

```
nazwa = [element1, element2, ..., elementN]
```

Pusta lista:

```
a = []
b = list()
```

Lista z liczbami:

```
a = [2, 3, 4.5, 5, -3.2]
```

Lista mieszana:

```
b = ['abcd', 25+3j, True, 1]
```

Kolejność ma znaczenie

```
a = [1, 2, 3, 4]
b = [4, 3, 2, 1]
print(a == b)
```

```
## False
```

Elementy na liście nie muszą być unikalne

```
a = [1, 2, 3, 4, 2]
b = [1, 2, 3, 4]
print(a)
```

```
## [1, 2, 3, 4, 2]
```

```
print(a == b)
```

```
## False
```

Indeks - dostęp do elementów listy (od zera)

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[1])
```

```
## 3
```

```
print(a[4])
```

```
## -2.3
```

```
print(a[0])
#print(a[7])
```

```
## 1
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[-1])
```

```
## 9.3
```

```
print(a[-5])
```

```
## abc
```

```
print(a[-7])
```

```
## 1
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[1:4])
```

```
## [3, 'abc', False]
```

```
print(a[-5:-2])
```

```
## ['abc', False, -2.3]
```

```
print(a[:4])
```

```
## [1, 3, 'abc', False]
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[2:])
```

```
## ['abc', False, -2.3, 'XYZ', 9.3]
```

```
print(a[0:6:2])
```

```
## [1, 'abc', -2.3]
```

```
print(a[1:6:2])
```

```
## [3, False, 'XYZ']
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[6:0:-2])
```

```
## [9.3, -2.3, 'abc']
```

```
print(a[::-1])
```

```
## [9.3, 'XYZ', -2.3, False, 'abc', 3, 1]
```

```
print(a[:])
```

```
## [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[::2])
```

```
## [1, 'abc', -2.3, 9.3]
```

```
print(a[::-2])
```

```
## [9.3, -2.3, 'abc', 1]
```

Specjalne funkcje

Długość (rozmiar listy)

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(len(a))
```

```
## 7
```

Implementacja samodzielna długości

```
def dlugosc(lista):
    x = 0
    for i in lista:
        x += 1

    return x


a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(dlugosc(a))
```

```
## 7
```

Maksimum i minimum?

Działa wtedy gdy mamy porządek

- liczby <=
- napisy - porządek leksykograficzny (omówimy przy napisach)

```
a = [4,-5,3.4,-11.2]
print(min(a))
```

```
## -11.2
```

```
print(max(a))
```

```
## 4
```

```
b = ['abc', 'ABCd', 'krt', 'abcd']
print(min(b))
```

```
## ABCd
```

```
print(max(b))
```

```
## krt
```

Modyfikacja i wstawianie

```
a = [4,-5,3.4,-11.2]
a[2] = 'a'
print(a)
```

```
## [4, -5, 'a', -11.2]
```

```
a = [4,-5,3.4,-11.2]
a[2] = ['a','b']
print(a)
```

```
## [4, -5, ['a', 'b'], -11.2]
```

```
a = [4,-5,3.4,-11.2]
a[1:2] = ['a','b']
print(a)
```

```
## [4, 'a', 'b', 3.4, -11.2]
```

```
a = [4,-5,3.4,-11.2]
a[1:3] = ['a','b']
print(a)
```

```
## [4, 'a', 'b', -11.2]
```

Dodawanie list

```
a = [4,-5,3.4,-11.2]
b = ['a','b','c']
print(a+b)
```

```
## [4, -5, 3.4, -11.2, 'a', 'b', 'c']
```

Mnożenie listy przez liczbę całkowitą (int)

```
a = [4,-5,3.4,-11.2]
print(a*2)
```

```
## [4, -5, 3.4, -11.2, 4, -5, 3.4, -11.2]
```

```
print(2*a)
```

```
## [4, -5, 3.4, -11.2, 4, -5, 3.4, -11.2]
```

Usuwanie elementów z listy

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
del a[2]
print(a)
```

```
## [1, 3, False, -2.3, 'XYZ', 9.3]
```

```
del a[:]
print(a)
```

```
## []
```

Do poczytania

- https://docs.python.org/3.9/tutorial/introduction.html#lists
- https://docs.python.org/3.9/tutorial/datastructures.html#more-on-lists

`list.append(x)` - dodaje element na końcu listy. Równoważnie `a[len(a):] = [x]`

```
a = [1, 3, 'abc', False]
a.append(5.3)
print(a)
```

```
## [1, 3, 'abc', False, 5.3]
```

`list.extend(iterable)` - dodaje elementy z argumenty na koniec listy. Równoważnie: `a[len(a):] = iterable`

```
a = [1, 3, 'abc', False]
b = [3, -2]
a.extend(b)
print(a)
```

```
## [1, 3, 'abc', False, 3, -2]
```

Różnice?

```
a = [1, 3, 'abc', False]
b = [3, -2]
a.append(b)
print(a)
```

```
## [1, 3, 'abc', False, [3, -2]]
```

`list.insert(i, x)` - wstawia element `x` na pozycji `i`

```
a = [1, 3, 'abc', False]
a.insert(0,'w')
print(a)
```

```
## ['w', 1, 3, 'abc', False]
```

```
a.insert(4,9.0)
print(a)
```

```
## ['w', 1, 3, 'abc', 9.0, False]
```

`list.remove(x)` - usuwa element z listy (pierwszy od początku)

```
a = [1, 3, 'abc', False]
a.remove(False)
print(a)
```

```
## [1, 3, 'abc']
```

```
b = [3, 4, 5, 3]
b.remove(3)
print(b)
```

```
## [4, 5, 3]
```

`list.pop()` - usuwa i zwraca ostatni element

`list.pop(i)` - usuwa i zwraca element na pozycji `i`

```
a = [1, 3, 'abc', False]
print(a.pop())
```

```
## False
```

```
print(a)
```

```
## [1, 3, 'abc']
```

```
b = [3, -4, 6.2, 7]
print(b.pop(3))
```

```
## 7
```

```
print(b)
```

```
## [3, -4, 6.2]
```

`list.clear()` - usuwa wszystkie elementy z listy. Równoważnie: `del a[:]`

```
a = [1, 3, 'abc', False]
a.clear()
print(a)
```

```
## []
```

`list.index(x)` - zwraca indeks elementu x (o ile istnieje, inaczej błąd), w przypadku duplikatów pierwszy z lewej
`list.index(x,start)` - zwraca indeks elementu x (o ile istnieje, inaczej błąd) zaczynając od pozycji `start`, w przypadku duplikatów pierwszy z lewej `list.index(x,start,end)` - zwraca indeks elementu x (o ile istnieje, inaczej błąd) zaczynając od pozycji `start` a kończąc na `end-1`, w przypadku duplikatów pierwszy z lewej

```
a = [1, 3, 1, 4, 5, 2, 3]
print(a.index(3))
```

```
## 1
```

```
print(a.index(3, 5))
```

```
## 6
```

```
print(a.index(3, 1, 4))
```

```
## 1
```

```
a = ['abc','xyz','abc','efg']
print(a.index('abc'))
```

```
## 0
```

```
print(a.index('abc', 2))
```

```
## 2
```

```
print(a.index('abc', 1, 4))
```

```
## 2
```

`list.count(x)` - zwraca ile razy występuję element `x` na liście

```
a = ['abc','xyz','abc','efg']
print(a.count('abc'))
```

```
## 2
```

```
print(a.count(4))
```

```
## 0
```

`list.sort()` - sortuje listę (o ile elementy można posortować)

```
a = ['abc','xyz','abc','efg']
a.sort()
print(a)
```

```
## ['abc', 'abc', 'efg', 'xyz']
```

`list.reverse()` - odwraca kolejność elementów na liście (nie ma nic związku z sortowaniem!)

```
a = [4, 5, -2, 7.3, 9, -22, 23]
a.reverse()
print(a)
```

```
## [23, -22, 9, 7.3, -2, 5, 4]
```

`list.copy()` - tworzy kopię listy

Spójrzmy na przykład jak działa operator przypisania dla list.

```
a = [4, 5, -2, 7.3, 9, -22, 23]
b = a
b[2] = 100
print(b)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

```
print(a)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

Różnica z użyciem copy.

```
a = [4, 5, -2, 7.3, 9, -22, 23]
b = a.copy()
b[2] = 100
print(b)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

```
print(a)
```

```
## [4, 5, -2, 7.3, 9, -22, 23]
```

Lista jako stos

```
stack = [3, 4, 5, 8, 9]
stack.append(6)
stack.append(7)
print(stack)
```

```
## [3, 4, 5, 8, 9, 6, 7]
```

```
print(stack.pop())
```

```
## 7
```

```
print(stack)
```

```
## [3, 4, 5, 8, 9, 6]
```

Lista jako kolejka

```
from collections import deque

queue = deque(["aw", "tg", "kj"])
queue.append("gg")
print(queue)
```

```
## deque(['aw', 'tg', 'kj', 'gg'])
```

```
print(queue.popleft())
```

```
## aw
```

```
print(queue)
```

```
## deque(['tg', 'kj', 'gg'])
```

List Comprehensions

```
squares = []
for x in range(5):
    squares.append(x ** 2)

print(squares)
```

```
## [0, 1, 4, 9, 16]
```

```
squares = [x**2 for x in range(5)]
print(squares)
```

```
## [0, 1, 4, 9, 16]
```

Krotka - tuple

```
krotka = 123, 'abc', True
print(krotka[2])
```

```
## True
```

```
krotka[0] = 1
```

```
## TypeError: 'tuple' object does not support item assignme
```

Zbiór - set

```
cyfry = {'raz', 'dwa', 'raz', 'trzy', 'raz', 'osiem'}
print(cyfry)
```

```
## {'trzy', 'raz', 'dwa', 'osiem'}
```

Słownik

```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
print(tel)
```

```
## {'jack': 4098, 'sape': 4139, 'guido': 4127}
```

```
tel['jack']
```

```
## 4098
```

```
del tel['sape']
tel['irv'] = 4127
print(tel)
```

```
## {'jack': 4098, 'guido': 4127, 'irv': 4127}
```

```
print(list(tel))
```

```
## ['jack', 'guido', 'irv']
```

```
print(sorted(tel))
```

```
## ['guido', 'irv', 'jack']
```

Funkcje

```
def functionname( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

```
def printme(str):
    """Funkcja wyświetlająca string"""
    print(str)
    return


printme("abc")
```

```
## abc
```

```
print(printme.__doc__)
```

```
## Funkcja wyświetlająca string
```

Przekazywanie przez referencję

```
def changeme(lista):
    print("Przed zmianą: ", lista)
    lista[2] = 50
    print("Po zmianie: ", lista)
    return


mylist = [10, 20, 30]
changeme(mylist)
```

```
## Przed zmianą:  [10, 20, 30]
## Po zmianie:  [10, 20, 50]
```

```
print("Poza funkcją: ", mylist)
```

```
## Poza funkcją:  [10, 20, 50]
```

```
def changeme(lista):
    lista = [2, 3, 4]
    print("Wewnątrz funkcji: ", lista)
    return


lista = [10, 20, 30]
changeme(lista)
```

```
## Wewnątrz funkcji:  [2, 3, 4]
```

```
print("Poza funkcją: ", lista)
```

```
## Poza funkcją:  [10, 20, 30]
```

```
def changeme():
    global lista
    lista = [2, 3, 4]
    print("Wewnątrz funkcji: ", lista)
    return


changeme()
```

```
## Wewnątrz funkcji:  [2, 3, 4]
```

```
print("Poza funkcją: ", lista)
```

```
## Poza funkcją:  [2, 3, 4]
```

Obowiązkowy argument

```
def printme(str):
    print(str)
    return


printme()
```

```
## TypeError: printme() missing 1 required
positional argument: 'str'
```

Keyword argument

```
def kwadrat(a):
    return a*a


print(kwadrat(a=4))
```

```
## 16
```

Domyślny argument

```
def sumsub(a, b, c=0, d=0):
    return a - b + c - d


print(sumsub(12, 4))
```

```
## 8
```

```
print(sumsub(3, 4, 5, 7))
```

```
## -3
```

```
def srednia(first, *values):
    return (first + sum(values)) / (1 + len(values))


print(srednia(2, 3, 4, 6))
```

```
## 3.75
```

```
print(srednia(45))
```

```
## 45.0
```

```
def f(**kwargs):
    print(kwargs)


f()
```

```
## {}
```

```
f(pl="Polish", en="English")
```

```
## {'pl': 'Polish', 'en': 'English'}
```

Funkcje matematyczne

Link do dokumentacji https://docs.python.org/3/library/math.html

```
import math


a=0
b=math.sin(2*math.pi)
print(b)
```

```
## -2.4492935982947064e-16
```

```
print(math.isclose(a,b, rel_tol=1e-09, abs_tol=1e-09))
```

```
## True
```

Programowanie obiektowe w Pythonie

Rysunek 1: Lego jako model programowanie obiektowego

```
class Employee:
    """Common base class for all employees"""
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print("Total Employee %d" % Employee.empCount)

    def displayEmployee(self):
        print("Name : ", self.name, ", Salary: ",
              self.salary)
```

```
emp1 = Employee("John", 2000)
emp2 = Employee("Anna", 5000)
emp1.displayEmployee()
```

```
## Name :  John , Salary:  2000
```

```
emp2.displayEmployee()
```

```
## Name :  Anna , Salary:  5000
```

Wizualizacja danych

Czym zajmuje się wizualizacja danych?

Wizualizacja – ogólna nazwa graficznych metod tworzenia, analizy i przekazywania informacji. Za pomocą środków wizualnych ludzie wymieniają się zarówno ideami abstrakcyjnymi, jak i komunikatami mającymi bezpośrednie oparcie w rzeczywistości. W dzisiejszych czasach wizualizacja wpływa na sposób prowadzenia badań naukowych, jest rutynowo wykorzystywana w dyscyplinach technicznych i medycynie, służy celom dydaktycznym, a także bywa pojmowana jako środek wyrazu artystycznego.

Wizualizacja danych to zagadnienie ich obrazowego przedstawienia. Dane są rozumiane jako „informacje, które zostały zestawione w pewnej schematycznej formie, np. zmiennych lub współrzędnych". Według Friedmana jej głównym celem jest skuteczny i zrozumiały przekaz zawartych w nich treści. Jednym z najczęściej popełnianych błędów bywa przykładanie zbytniej uwagi do formy komunikatu, który przestaje spełniać swoje zasadnicze zadanie. Odmienny pogląd na sens tej dziedziny wyrażają Fernanda Viegas i Martin M. Wattenberg, akcentując rolę pozyskania uwagi potencjalnego odbiorcy. Odpowiedni sposób przedstawienia danych pozwala na poprawne i szybkie zrozumienie zależności opisanych przez dane. Nieodpowiedni sposób prezentacji prowadzi do powstawania celowych lub przypadkowych zniekształceń w postrzeganiu zależności obecnych w danych.

Analiza danych - podstawowe pojęcia

Analiza danych - podstawowe pojęcia

Współczesne znaczenia słowa “statystyka”:

- zbiór danych liczbowych pokazujący kształtowanie procesów i zjawisk np. statystyka ludności.
- wszelkie czynności związane z gromadzeniem i opracowywaniem danych liczbowych np. statystyka pewnego problemu dokonywana przez GUS.
- charakterystyki liczbowe np. statystyki próby np. średnia arytmetyczna, odchylenie standardowe itp.
- dyscyplina naukowa - nauka o metodach badania zjawisk masowych.

Zjawisko/procesy masowe - badaniu podlega duża liczba jednostek. Dzielą się na:

- gospodarcze (np. produkcja, konsumpcja, usługi reklama),
- społeczne (np. wypadki drogowe, poglądy polityczne),
- demograficzne (np. urodzenia, starzenie, migracje).

Statystyka - dyscyplina naukowa - podział:

- statystyka opisowa - zajmuje się sprawami związanymi z gromadzeniem, prezentacją, analizą i interpretacją danych liczbowych. Obserwacja obejmuje całą badaną zbiorowość.
- statystyka matematyczna - uogólnienie wyników badania części zbiorowości (próby) na całą zbiorowość.

Zbiorowość statystyczna, populacja statystyczna: zbiór obiektów podlegających badaniu statystycznemu. Tworzą je jednostki podobne do siebie, logicznie powiązane, lecz nie identyczne. Mają pewne cechy wspólne oraz pewne właściwości pozwalające je różnicować.

- przykłady:
  - badanie wzrostu Polaków - mieszkańcy Polski
  - poziom nauczania w szkołach woj. warmińsko-mazurskiego - szkoły woj. warmińsko-mazurskiego.
- podział:
  - zbiorowość/populacja generalna - obejmuje całość,
  - zbiorowość/populacja próbna (próba) - obejmuje część populacji.

Jednostka statystyczna: każdy z elementów zbiorowości statystycznej.

- przykłady:
  - studenci UWM - student UWM
  - mieszkańcy Polski - każda osoba mieszkająca w Polsce
  - maszyny produkowane w fabryce - każda maszyna

Cechy statystyczne

- właściwości charakteryzujące jednostki statystyczne w danej zbiorowości statystycznej.
- dzielimy je na stałe i zmienne.

Cechy stałe

- takie właściwości, które są wspólne wszystkim jednostkom danej zbiorowości statystycznej.
- podział:
  - rzeczowe - kto lub co jest przedmiotem badania statystycznego,
  - czasowe - kiedy zostało przeprowadzone badanie lub jakiego okresu czasu dotyczy badanie,
  - przestrzenne - jakiego terytorium (miejsce lub obszar) dotyczy badanie.
- przykład: studenci WMiI UWM w Olsztynie w roku akad. 2017/2018:
  - cecha rzeczowa: posiadanie legitymacji studenckiej,
  - cecha czasowa - studenci studiujący w roku akad. 2017/2018
  - cecha przestrzenna - miejsce: WMiI UWM w Olsztynie.

Cechy zmienne

- właściwości różnicujące jednostki statystyczne w danej zbiorowości.
- przykład: studenci UWM - cechy zmienne: wiek, płeć, rodzaj ukończonej szkoły średniej, kolor oczu, wzrost.

Ważne:

- obserwacji podlegają tylko cechy zmienne,
- cecha stała w jednej zbiorowości może być cechą zmienną w innej zbiorowości.

Przykład: studenci UWM mają legitymację wydaną przez UWM. Studenci wszystkich uczelni w Polsce mają legitymacje wydane przez różne szkoły.

Podział cech zmiennych:

- cechy mierzalne (ilościowe) - można je wyrazić liczbą wraz z określoną jednostką miary.
- cechy niemierzalne (jakościowe) - określane słownie, reprezentują pewne kategorie.

Przykład: zbiorowość studentów. Cechy mierzalne: wiek, waga, wzrost, liczba nieobecności. Cechy niemierzalne: płeć, kolor oczu, kierunek studiów.

Często ze względów praktycznych cechom niemierzalnym przypisywane są kody liczbowe. Nie należy ich jednak mylić z cechami mierzalnymi. Np. 1 - wykształcenie podstawowe, 2 - wykształcenie zasadnicze, itd...

Podział cech mierzalnych:

- ciągłe - mogące przybrać każdą wartość z określonego przedziału, np. wzrost, wiek, powierzchnia mieszkania.
- skokowe - mogące przyjmować konkretne (dyskretne) wartości liczbowe bez wartości pośrednich np. liczba osób w gospodarstwie domowych, liczba osób zatrudnionych w danej firmie.

Cechy skokowe zazwyczaj mają wartości całkowite choć nie zawsze jest to wymagane np. liczba etatów w firmie (z uwzględnieniem części etatów).

Skale

Skala pomiarowa

- to system, pozwalający w pewien sposób usystematyzować wyniki pomiarów statystycznych.
- podział:
  - skala nominalna,
  - skala porządkowa,
  - skala przedziałowa (interwałowa),
  - skala ilorazowa (stosunkowa).

Skala nominalna

- skala, w której klasyfikujemy jednostkę statystyczną do określonej kategorii.
- wartość w tej skali nie ma żadnego uporządkowana.
- przykład:

Religia	Kod
Chrześcijaństwo	1
Islam	2
Buddyzm	3

Skala porządkowa

- wartości mają jasno określony porządek, ale nie są dane odległości między nimi,
- pozwala na uszeregowanie elementów.
- przykłady:

Wykształcenie	Kod
Podstawowe	1
Średnie	2
Wyższe	3

Dochód	Kod
Niski	1
Średni	2
Wysoki	3

Skala przedziałowa (interwałowa)

- wartości cechy wyrażone są poprzez konkretne wartości liczbowe,
- pozwala na porównywanie jednostek (coś jest większe lub mniejsze),
- nie możliwe jest badanie ilorazów (określenie ile razy dana wartość jest większa lub mniejsza od drugiej).
- przykład:

Miasto	Temperatura w $^\circ C$	Temperatura w $^\circ F$
Warszawa	15	59
Olsztyn	10	50
Gdańsk	5	41
Szczecin	20	68

Skala ilorazowa (stosunkowa)

- wartości wyrażone są przez wartości liczbowe,
- możliwe określenie jest relacji mniejsza lub większa między wartościami,
- możliwe jest określenie stosunku (ilorazu) między wartościami,
- występuje zero absolutne.
- przykład:

Produkt	Cena w zł
Chleb	3
Masło	8
Gruszki	5

Rodzaje badań statystycznych

- badanie pełne - obejmują wszystkie jednostki zbiorowości statystycznej.
  - spis statystyczny,
  - rejestracja bieżąca,
  - sprawozdawczość statystyczna.
- badania częściowe - obserwowana jest część populacji. Przeprowadza się wtedy gdy badanie pełne jest niecelowe lub niemożliwe.
  - metoda monograficzna,
  - metoda reprezentacyjna.

Etapy badania statystycznego

- projektowanie i organizacja badania: ustalenie celu, podmiotu, przedmiotu, zakresu, źródła i czasu trwania badania;
- obserwacja statystyczna;
- opracowanie materiału statystycznego: kontrola materiału statystycznego, grupowanie uzyskanych danych, prezentacja wyników danych;
- analiza statystyczna.

Analiza danych zastanych

Analiza danych zastanych – proces przetwarzania danych w celu uzyskania na ich podstawie użytecznych informacji i wniosków. W zależności od rodzaju danych i stawianych problemów, może to oznaczać użycie metod statystycznych, eksploracyjnych i innych.

Korzystanie z danych zastanych jest przykładem badań niereaktywnych - metod badań zachowań społecznych, które nie wpływają na te zachowania. Dane takie to: dokumenty, archiwa, sprawozdania, kroniki, spisy ludności, księgi parafialne, dzienniki, pamiętniki, blogi internetowe, audio-pamiętniki, archiwa historii mówionej i inne. (Wikipedia)

Dane zastane możemy podzielić ze względu na (Makowska red. 2013):

- Charakter: Ilościowe, Jakościowe
- Formę: Dane opracowane, Dane surowe
- Sposób powstania: Pierwotne, Wtórne
- Dynamikę: Ciągła rejestracja zdarzeń, Rejestracja w interwałach czasowych, Rejestracja jednorazowa
- Poziom obiektywizmu: Obiektywne, Subiektywne
- Źródła pochodzenia: Dane publiczne, Dane prywatne

Analiza danych to proces polegający na sprawdzaniu, porządkowaniu, przekształcaniu i modelowaniu danych w celu zdobycia użytecznych informacji, wypracowania wniosków i wspierania procesu decyzyjnego. Analiza danych ma wiele aspektów i podejść, obejmujących różne techniki pod różnymi nazwami, w różnych obszarach biznesowych, naukowych i społecznych. Praktyczne podejście do definiowania danych polega na tym, że dane to liczby, znaki, obrazy lub inne metody zapisu, w formie, którą można ocenić w celu określenia lub podjęcia decyzji o konkretnym działaniu. Wiele osób uważa, że dane same w sobie nie mają znaczenia – dopiero dane przetworzone i zinterpretowane stają się informacją.

Proces analizy danych

Analiza odnosi się do rozbicia całości posiadanych informacji na jej odrębne komponenty w celu indywidualnego badania. Analiza danych to proces uzyskiwania nieprzetworzonych danych i przekształcania ich w informacje przydatne do podejmowania decyzji przez użytkowników. Dane są zbierane i analizowane, aby odpowiadać na pytania, testować hipotezy lub obalać teorie. Istnieje kilka faz, które można wyszczególnić w procesie analizy danych. Fazy są iteracyjne, ponieważ informacje zwrotne z faz kolejnych mogą spowodować dodatkową pracę w fazach wcześniejszych.

Zdefiniowanie wymagań

Przed przystąpieniem do analizy danych, należy dokładnie określić wymagania jakościowe dotyczące danych. Dane wejściowe, które mają być przedmiotem analizy, są określone na podstawie wymagań osób kierujących analizą lub klientów (którzy będą używać finalnego produktu analizy). Ogólny typ jednostki, na podstawie której dane będą zbierane, jest określany jako jednostka eksperymentalna (np. osoba lub populacja ludzi. Dane mogą być liczbowe lub kategoryczne (tj. Etykiety tekstowe). Faza definiowania wymagań powinna dać odpowiedź na 2 zasadnicze pytania:

- co chcemy zmierzyć?
- w jaki sposób chcemy to zmierzyć?

Gromadzenie danych

Dane są gromadzone z różnych źródeł. Wymogi, co do rodzaju i jakości danych mogą być przekazywane przez analityków do "opiekunów danych", takich jak personel technologii informacyjnych w organizacji. Dane ponadto mogą być również gromadzone automatycznie z różnego rodzaju czujników znajdujących się w otoczeniu - takich jak kamery drogowe, satelity, urządzenia rejestrujące obraz, dźwięk oraz parametry fizyczne. Kolejną metodą jest również pozyskiwanie danych w drodze wywiadów, gromadzenie ze źródeł internetowych lub bezpośrednio z dokumentacji.

Przetwarzanie danych

Zgromadzone dane muszą zostać przetworzone lub zorganizowane w sposób logiczny do analizy. Na przykład, mogą one zostać umieszczone w tabelach w celu dalszej analizy - w arkuszu kalkulacyjnym lub innym oprogramowaniu. Oczyszczanie danych Po fazie przetworzenia i uporządkowania, dane mogą być niekompletne, zawierać duplikaty lub zawierać błędy. Konieczność czyszczenia danych wynika z problemów związanych z wprowadzaniem i przechowywaniem danych. Czyszczenie danych to proces zapobiegania powstawaniu i korygowania wykrytych błędów. Typowe zadania obejmują dopasowywanie rekordów, identyfikowanie nieścisłości, ogólny przegląd jakość istniejących danych, usuwanie duplikatów i segmentację kolumn. Niezwykłe istotne jest też zwracanie uwagi na dane których wartości są powyżej lub poniżej ustalonych wcześniej progów (ekstrema).

Właściwa analiza danych

Istnieje kilka metod, które można wykorzystać do tego celu, na przykład data mining, business intelligence, wizualizacja danych lub badania eksploracyjne. Ta ostatnia metoda jest sposobem analizowania zbiorów informacji w celu określenia ich odrębnych cech. W ten sposób dane mogą zostać wykorzystane do przetestowania pierwotnej hipotezy. Statystyki opisowe to kolejna metoda analizy zebranych informacji. Dane są badane, aby znaleźć najważniejsze ich cechy. W statystykach opisowych analitycy używają kilku podstawowych narzędzi - można użyć średniej lub średniej z zestawu liczb. Pomaga to określić ogólny trend aczkolwiek nie zapewnia to dużej dokładności przy ocenie ogólnego obrazu zebranych danych. W tej fazie ma miejsce również modelowanie i tworzenie formuł matematycznych - stosowane są w celu identyfikacji zależności między zmiennymi, takich jak korelacja lub przyczynowość.

Raportowanie i dystrybucja wyników

Ta faza polega na ustalaniu w jakiej formie przekazywać wyniki. Analityk może rozważyć róże techniki wizualizacji danych, aby w sposób wyraźnym i skuteczny przekazać wnioski z analizy odbiorcom. Wizualizacja danych wykorzystuje formy graficzne jak wykresy i tabele. Tabele są przydatne dla użytkownika, który może wyszukiwać konkretne rekordy, podczas gdy wykresy (np. wykresy słupkowe lub liniowe) dają spojrzenie ilościowych na zbiór analizowanych danych.

Skąd brać dane?

Darmowa repozytoria danych:

- Bank danych lokalnych GUS - link
- Otwarte dane - link
- Bank Światowy - link

Przydatne strony:

- https://astrafox.pl/10-darmowych-baz-danych-na-wyciagniecie-reki/
- https://www.nature.com/sdata/policies/repositories
- https://medium.freecodecamp.org/https-medium-freecodecamp-org-best-free-open-data-sources-anyone-can-use-a65b514b0f2d

“Tidy data”

Koncepcja

Koncepcja czyszczenia danych (ang. tidy data):

- WICKHAM, Hadley . Tidy Data. Journal of Statistical Software, [S.l.], v. 59, Issue 10, p. 1 - 23, sep. 2014. ISSN 1548-7660. Available at: https://www.jstatsoft.org/v059/i10. Date accessed: 25 oct. 2018. doi:http://dx.doi.org/10.18637/jss.v059.i10.

Zasady “czystych danych”

Idealne dane są zaprezentowane w tabeli:

Imię	Wiek	Wzrost	Kolor oczu
Adam	26	167	Brązowe
Sylwia	34	164	Piwne
Tomasz	42	183	Niebieskie

Na co powinniśmy zwrócić uwagę?

- jedna obserwacja (jednostka statystyczna) = jeden wiersz w tabeli/macierzy/ramce danych
- wartości danej cechy znajdują się w kolumnach
- jeden typ/rodzaj obserwacji w jednej tabeli/macierzy/ramce danych

Przykłady nieuporządkowanych danych

Imię	Wiek	Wzrost	Brązowe	Niebieskie	Piwne
Adam	26	167	1	0	0
Sylwia	34	164	0	0	1
Tomasz	42	183	0	1	0

Nagłowki kolumn muszą odpowiadać cechom, a nie wartościom zmiennych.

Kod do analizy

https://gist.github.com/pjastr/309281eedf2ca5d0425b26e8d12eaa6f

Jak tworzyć wykresy?

http://smarterpoland.pl/

https://www.kdnuggets.com/2021/02/telling-great-data-story-visualization-decision-tree.html ## Bibliografia

- https://pl.wikipedia.org/wiki/Python, dostęp online 12.02.2019.
- https://bulldogjob.pl/news/264-java-php-ruby-jak-wlasciwie-wymawiac-nazwy-technologii. dostęp online 12.02.2019.
- https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html, dostęp online 14.02.2019.
- K. Ropiak, Wprowadzenie do języka Python, http://wmii.uwm.edu.pl/~kropiak/wd/Wprowadzenie%20do%20j%C4%99zyka%20Python.pdf, dostęp online 14.02.2019.
- B. Slatkin, Efektywny Python. 59 sposobów na lepszy kod, Helion 2015.
- https://www.python.org/dev/peps/pep-0008/, dostęp online 14.02.2019.

Bibliografia - cd2

- https://www.flynerd.pl/2017/05/python-4-typy-i-zmienne.html, dostęp online 14.02.2019.
- http://pytolearn.csd.auth.gr/p0-py/01/print.html, dostęp online 15.02.2019.
- https://www.tutorialspoint.com/python3/python_lists.htm, dostęp online 17.02.2019.

Bibliografia - cd3

- https://realpython.com/python-data-types/, dostęp online 5.01.2022
- https://www.w3schools.com/python/python_variables.asp, dostęp online 5.01.2022
- https://www.w3schools.com/python/python_variables_multiple.asp, dostęp online 5.01.2022
- https://realpython.com/python-print/, dostęp online 5.01.2022
- https://www.programiz.com/python-programming/operators, dostęp online 5.01.2022
- https://realpython.com/python-conditional-statements/, dostęp online 5.01.2022
- https://realpython.com/python-for-loop/, dostęp online 5.01.2022
- https://realpython.com/python-while-loop/, dostęp online 5.01.2022

- https://pl.wikipedia.org/wiki/Wizualizacja
- https://mfiles.pl/pl/index.php/Analiza_danych, dostęp online 1.04.2019.
- Walesiak M., Gatnar E., Statystyczna analiza danych z wykorzystaniem programu R, PWN, Warszawa, 2009.
- Wasilewska E., Statystyka opisowa od podstaw, Podręcznik z zadaniami, Wydawnictwo SGGW, Warszawa, 2009.