Programowanie Python 1

(CP1S02005)

Politechnika Białostocka - Wydział Elektryczny

Cyfryzacja przemysłu, sem. II, studia stacjonarne I stopnia

Rok akademicki 2023/2024

Wykład nr 2 (06.03.2024)

dr inż. Jarosław Forenc

Plan wykładu nr 2

- Nazwy zmiennych, typy
- Operatory rozszerzonego przypisania
- Operatory porównania i logiczne
- Instrukcje if, if-else, if-elif-else
- Operator warunkowy

Python - nazwy zmiennych

- Reguły i zalecenia:
  - nazwy zmiennych mogą zawierać jedynie litery, cyfry i znak podkreślenia
  - nazwa zmiennej może rozpoczynać się od litery lub znaku podkreślenia, ale nie od cyfry

```
temp2     _variable4
```

```
2temp     my-variable
```

  - nie można stosować spacji w nazwach zmiennych (ale można zastosować podkreślenie do separacji słów)

```
my_variable     pole_koła     temp_celsjusz
```

  - unikamy stosowania w nazwach zmiennych słów kluczowych Pythona

```
print     class     input     pass
```

Python - nazwy zmiennych

- Reguły i zalecenia:
  - nazwa zmiennej powinna być krótka, ale czytelna

```
tc      tempc
```

```
temperatura_w_skali_celsjusza
```

  - ostrożnie używamy małej litery l i wielkiej litery O, bo mogą być łatwo pomylone z cyframi 1 i 0
  - nazwy zmiennych piszemy małymi literami, gdyż przyjęło się, że wielkie litery są używane do nazw stałych
- Nie ma specjalnego sposobu zapisywania / definiowania stałych

```
MAX = 9999
```

```
(constant) MAX: Literal[9999]
MAX = 9999
```

```
(variable) max: Literal[9999]
max = 9999
```

Python - typy

- Liczby całkowite:
  - stosowane systemy liczbowe: dziesiętny, dwójkowy, ósemkowy, szesnastkowy

```
w, x, y, z = 10, 0b10, 0o10, 0x10
print(w,x,y,z)
```

```
10 2 8 16
```

  - przy dzieleniu liczb całkowitych wynik jest zawsze liczbą zmiennoprzecinkową

```
x = 4/2
print(x)
```

```
2.0
```

Python - typy

- Liczby całkowite:
  - przy zapisywaniu liczb składających się z wielu cyfr można grupować je za pomocą znaków podkreślenia

```
ludnosc = 8_019_000_000     # stan na 01.01.2024
print(ludnosc)
```

```
8019000000
```

Python - typy

- Liczby zmiennoprzecinkowe:
  - w przypadku operacji na liczbie całkowitej i rzeczywistej wynik jest zawsze zmiennoprzecinkowy

```
x = 2 + 2.0
print(x)
```

```
4.0
```

  - czasem wyniki operacji na liczbach zmiennoprzecinkowych może być zaskakujący

```
x = 0.2 + 0.1
print(x)
```

```
0.30000000000000004
```

Python - typy

- Ciągi tekstowe:
  - ciąg tekstowy może być ujęty w cudzysłów lub apostrofy

```
txt1 = "Ala ma laptopa"
txt2 = 'Janek ma tablet'
```

- Wbudowana funkcja type() pozwala sprawdzić typ przekazanego argumentu

```
print(type(32))
print(type(2.5))
print(type(3.1e-4))
print(type(True))
print(type("napis"))
print(type('napis'))
```

```
<class 'int'>
<class 'float'>
<class 'float'>
<class 'bool'>
<class 'str'>
<class 'str'>
```

Python - operatory rozszerzonego przypisania

Operator	Przykład instrukcji	Instrukcja równoważna
`+=`	`x += 10`	`x = x + 10`
`-=`	`x -= 10`	`x = x - 10`
`*=`	`x *= 10`	`x = x * 10`
`/=`	`x /= 10`	`x = x / 10`
`//=`	`x //= 10`	`x = x // 10`
`%=`	`x %= 10`	`x = x % 10`
`**=`	`x **= 10`	`x = x ** 10`

Python - operatory porównania

Operator	Znaczenie	Przykład	Wynik
==	równa się	`2 == 2`	`True`
!=	nie równa się	`2 != 2`	`False`
>	większe niż	`2 > 5`	`False`
<	mniejsze niż	`2 < 5`	`True`
>=	mniejsze lub równe	`2 >= 5`	`False`
<=	większe lub równe	`2 <= 2`	`True`

- w wyniku porównania otrzymujemy wartość True (prawda) lub False (fałsz)
- wartości True i False można przypisywać zmiennym

```
koniec = True
warunek = False
```

Python - operatory porównania (przykłady)

```
wiek = 17
print(wiek == 18)
```

```
False
```

```
wiek = 17
print(wiek < 18)
```

```
True
```

```
wiek = 17
print(wiek >= 15)
```

```
True
```

Python - operatory porównania (przykłady)

```
imie = "Jan"
print(imie != "Jan")
```

```
False
```

```
imie = "Jan"
print(imie == "Jan")
```

```
True
```

```
imie = "Jan"
print(imie > "Ela")
```

```
True
```

Python - operatory porównania (przykłady)

```
imie = "Jan"
print(imie == "jan")
```

```
False
```

```
imie = "Jan"
print(imie.lower() == "jan")
```

```
True
```

- metoda lower() zwraca ciąg znaków, w którym wszystkie wielkie litery zostały zamienione na małe
- metoda ta nie zmienia wartości zmiennej imie

Python - operatory logiczne

Operator	Znaczenie	Przykład
`and`	zwraca wartość True jeśli oba warunki są prawdziwe	`x > 5 and x <= 7`
`or`	zwraca wartość True jeśli przynajmniej jeden z warunków jest prawdziwy	`x < 3 or x >= 4`
`not`	zwraca odwrotność wartości logicznej wyrażenia	`not x`

```
wiek1 = 19
wiek2 = 17
print(wiek1 >= 18 and wiek2 >= 18)
```

```
False
```

Python - operatory logiczne (przykłady)

- w warunku można zastosować dodatkowe nawiasy, ale nie jest to konieczne

```
wiek1 = 19
wiek2 = 17
print((wiek1 >= 18) and (wiek2 >= 18))
```

- przykład zastosowania operatora or

```
wiek1 = 19
wiek2 = 17
print(wiek1 >= 18 or wiek2 >= 18)
```

```
True
```

Python - instrukcja if

- Najprostsza postać instrukcji if:

```
if test_warunkowy:
    dowolna_akcja
```

  - jeśli wynikiem testu_warunkowego jest True to akcja jest wykonywana; jeśli zaś False - to akcja nie jest wykonywana
  - wcięty blok kodu określa akcje wykonywane po if

```
wiek = 19
if wiek >= 18:
    print("Jesteś pełnoletni")
```

  - blok kodu to jeden lub kilka kolejnych wierszy kodu z taką samą wielkością wcięcia
  - wcięcie to najczęściej: 4 spacje, 2 spacje, tabulator

Python - instrukcja if

- tam, gdzie kończy się wcięcie, tam też kończy się blok warunkowy:

```
wiek = 19
if wiek >= 18:
    print("Jesteś pełnoletni")
    print("Możesz iść na wybory")
print("Koniec")
```

Python - instrukcja if-else

- Składnia instrukcji if-else:

```
if test_warunkowy:
    akcja1
else:
    akcja2
```

  - jeśli wynikiem testu_warunkowego jest True to wykonywana jest tylko akcja1; jeśli zaś False - to wykonywana jest tylko akcja2

```
liczba = int(input("Podaj liczbe: "))
if liczba % 2 == 0:
    print(f"{liczba} - liczba parzysta")
else:
    print(f"{liczba} - liczba nieparzysta")
```

Python - przykład (pierwiastek kwadratowy)

```
import math

x = float(input("Podaj liczbę: "))
if x >= 0:
    y = math.sqrt(x)
    print("Pierwiastek liczby:", y)
else:
    print("Błąd! Liczba ujemna")
```

```
Podaj liczbę: 5
Pierwiastek liczby: 2.23606797749979

Podaj liczbę: -3
Błąd! Liczba ujemna
```

Python - przykład (pierwiastek kwadratowy)

```
import math

x = float(input("Podaj liczbę: "))
if x >= 0:
    y = math.sqrt(x)
    print("Pierwiastek liczby:", y)
else:
    print("Błąd! Liczba ujemna")
```

```
Podaj liczbę: a
Traceback (most recent call last):
  File "d:\MyApp.py", line 3, in <module>
    x = float(input("Podaj liczbę: "))
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: could not convert string to float: 'a'
```

Python - przykład (pierwiastek kwadratowy)

```
import math

try:
    x = float(input("Podaj liczbę: "))
    if x >= 0:
        y = math.sqrt(x)
        print("Pierwiastek liczby:", y)
    else:
        print("Błąd! Liczba ujemna")
except ValueError:
    print("Błąd! Wprowadź poprawną liczbę")
```

```
Podaj liczbę: a
Błąd! Wprowadź poprawną liczbę
```

Python - instrukcja if-elif-else

- Składnia instrukcji if-elif-else:

```
if test1:
    akcja1
elif test2:
    akcja2
else:
    akcja3
```

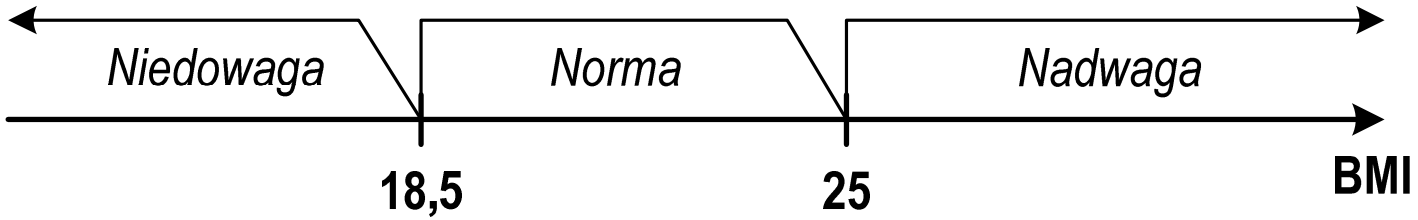
  - jeśli wynikiem test1 jest True to wykonywana jest akcja1
  - jeśli wynikiem test1 jest False to sprawdzany jest test2
  - jeśli wynikiem test2 jest True to wykonywana jest akcja2
  - jeśli wynikiem test2 jest False to wykonywana jest akcja3
  - tylko jedna akcja może być wykonana

Python - przykład (BMI)

- BMI - współczynnik powstały przez podzielenie masy ciała podanej w kilogramach przez kwadrat wzrostu podanego w metrach

$$BMI = \frac{masa}{wzrost^2}$$

- Dla osób dorosłych:
  - BMI < 18,5 - wskazuje na niedowagę
  - BMI ≥ 18,5 i BMI < 25 - wskazuje na prawidłową masę ciała
  - BMI ≥ 25 - wskazuje na nadwagę

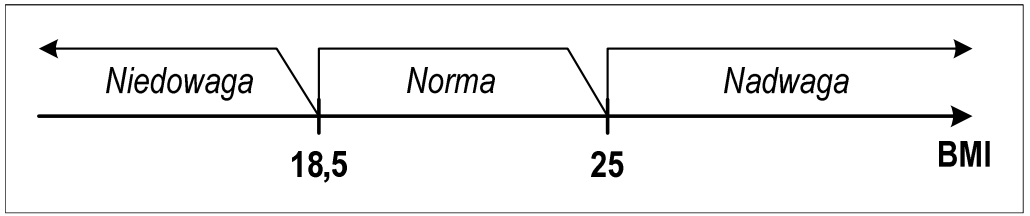


Python - przykład (BMI)

```
masa = float(input("Podaj masę [kg]: "))
wzrost = float(input("Podaj wzrost [m]: "))
bmi = masa / (wzrost * wzrost)
print("BMI:", "{:.2f}".format(bmi))

if bmi < 18.5:
    print("Niedowaga")
elif bmi >= 18.5 and bmi < 25:
    print("Norma")
else:
    print("Nadwaga")
```

```
Podaj masę [kg]: 84
Podaj wzrost [m]: 1.85
BMI: 24.54
Norma
```



Python - przykład (BMI)

```
masa = float(input("Podaj masę [kg]: "))
wzrost = float(input("Podaj wzrost [m]: "))
bmi = masa / (wzrost * wzrost)
print("BMI:", "{:.2f}".format(bmi))

if bmi < 18.5:
    print("Niedowaga")
elif 18.5 <= bmi < 25:
    print("Norma")
else:
    print("Nadwaga")
```

- warunek sprawdzający czy bmi znajduje się w przedziale można zapisać także w inny sposób

```
bmi >= 18.5 and bmi < 25
```
→
```
18.5 <= bmi < 25
```

Python - przykład (BMI)

```
masa = float(input("Podaj masę [kg]: "))
wzrost = float(input("Podaj wzrost [m]: "))
bmi = masa / (wzrost * wzrost)
print("BMI:", "{:.2f}".format(bmi))
if bmi < 18.5:
    print("Niedowaga")
else:
    if bmi < 25:
        print("Norma")
    else:
        print("Nadwaga")
```

- instrukcje if można zagnieżdżać

Python - przykład (oceny)

```
pkt = int(input("Podaj liczbę punktów: "))

if pkt < 51:
    ocena = 2.0
elif pkt < 61:
    ocena = 3.0
elif pkt < 71:
    ocena = 3.5
elif pkt < 81:
    ocena = 4.0
elif pkt < 91:
    ocena = 4.5
else:
    ocena = 5.0

print(f"Twoja ocena: {ocena:.1f}")
```

- blok elif może występować wielokrotnie
- ostatni else może być pominięty

Python - operator warunkowy

- Składnia operatora warunkowego (wyrażenia trójargumentowego):

```
wartość_prawda if warunek else wartość_fałsz
```

  - warunek jest warunkiem logicznym, który ma być sprawdzany
  - wartość_prawda jest wartością zwracaną, jeśli warunek jest spełniony
  - wartość_fałsz jest wartością zwracaną, jeśli warunek nie jest spełniony

```
x = int(input("Podaj liczbe: "))
txt = "Parzysta" if x % 2 == 0 else "Nieparzysta"
print(txt)
```

Koniec wykładu nr 2

Dziękuję za uwagę!