• Laboratorium Wstępu do Programowania: 2

• Temat: Instrukcja warunkowa if elif else, praca z plikami

• Wiktoria Kurowska

• Numer albumu: 73146

• Grupa: IN. INP L6

• Prowadzący: mgr inż. Przemysław Skubel

ZADANIE 1

Przebieg zadania:

Celem było napisanie programu, który na podstawie liczby zdobytych punktów określa wynik egzaminu. W zależności od progu punktowego student otrzymuje inny komunikat.

Kod programu:

Lab2_zad1.py > ...

```
1  wynik = int(input("Podaj zdobytą liczbę punktów: "))
2
3  if wynik > 80:
4      print("Zaliczyłeś egzamin")
5  elif wynik <= 80 and wynik >= 50 :
6      print("Możesz poprawić egzamin")
7  elif wynik < 50:
8      print("Musisz poprawić egzamin")
```

Przykładowy wynik programu:

Dla wartości 72

```
Podaj zdobytą liczbę punktów: 72
Możesz poprawić egzamin
```

Analiza działania:

Program pobiera wartość punktów, po czym instrukcja warunkowa sprawdza zakres. Kolejno porównywane są warunki: > 80, 50-80 oraz <50. Program wypisuje odpowiednią informację.

Wnioski:

Zastosowanie instrukcji if-elif-elif pozwala szybko i czytelnie określić wynik egzaminu na podstawie jednego parametru.

ZADANIE 2

Przebieg zadania:

Należało uporządkować trzy liczby od najmniejszej do największej bez korzystania z funkcji wbudowanych.

Kod programu:

Lab2_zad2.py > x

```python
x = int(input("Podaj pierwszą liczbę: "))
y = int(input("Podaj drugą liczbę: "))
z = int(input("Podaj trzecią liczbę: "))

if x > y:
    x, y = y, x
if x > z:
    x, z = z, x
if y > z:
    y, z = z, y

print(x, y, z)
```

Przykładowy wynik programu:

```
Podaj pierwszą liczbę: 5
Podaj drugą liczbę: 2
Podaj trzecią liczbę: 9
2 5 9
```

Analiza działania:

Program sprawdza, która liczba jest najmniejsza, a następnie porównuje pozostałe, aby ustalić ich kolejność. Zagnieżdżone instrukcje if pozwalają ustawić liczby rosnąco.

Wnioski:

Chociaż Python ma funkcje sortujące, ręczne porównanie liczb dobrze pokazuje logikę instrukcji warunkowych.

ZADANIE 3

Przebieg zadania:

Celem było sprawdzenie czy nazwa pliku ma rozszerzenie .xlsx.

Kod programu:

Lab2_zad3.py > Nazwa_pliku

```python
Nazwa_pliku = str(input("Podaj pełną nazwę pliku: "))

wynik = Nazwa_pliku.endswith(".xlsx")

if wynik == True:
    print("Tak")
elif wynik == False:
    print("Nie")
```

Przykładowy wynik programu:

```
Podaj pełną nazwę pliku: zadania.xlsx
Tak
```

Analiza działania:

Funkcja endswith() sprawdza zakończenie ciągu znaków. Warunek zwraca True lub False, co decyduje o wypisanym komunikacie.

Wnioski:

Metoda endswith() jest szybkim sposobem na weryfikację rozszerzeń plików.

ZADANIE 4

A)

Przebieg zadania:

Zadaniem było obliczyć wynik drużyny na podstawie liczby bramek i bonusów, przy czym bonus nie sumuje się (5 punktów po >5 golach, 10 po >10).

Kod programu:

```python
#a
gol = int(input("Podaj liczbę zdobytych bramek: "))

bonus = gol * 10

if gol > 5 and gol < 10:
    bonus = bonus + 5
if gol > 10:
    bonus = bonus + 10

print(bonus)
```

Przykładowy wynik programu:

```
Podaj liczbę zdobytych bramek: 7
75
```

Analiza działania:

Najpierw obliczane są punkty za gole, następnie sprawdzane są progi bonusowe. Opcje są rozłączne dzięki if.

Wnioski:

Program poprawnie działa dla różnych zakresów goli, a logika warunków jest czytelna.

B)

Przebieg zadania:

W tej wersji bonusy po >5 i >10 golach sumują się.

Kod programu:

```python
#b
gol = int(input("Podaj liczbę zdobytych bramek: "))

bonus = gol * 10

if gol > 5 and gol < 10:
    bonus = bonus + 5
if gol > 5 and gol > 10:
    bonus = bonus + 15

print(bonus)
```

Przykładowy wynik programu:

```
Podaj liczbę zdobytych bramek: 12
135
```

Analiza działania:

Dwa niezależne warunki umożliwiają dodawanie obu bonusów.

Wnioski:

Program poprawnie działa dla różnych zakresów goli, a logika warunków jest czytelna.

ZADANIE 5

A)

Przebieg zadania:

Należało odczytać plik liniami i wyświetlić jego zawartość.

Kod programu:

```python
#a
with open("notowania_gieldowe.txt", "r") as plik:
    print(plik.read())

```

Przykładowy wynik programu:

```
KGHM, 123
Tauron, 150
Orange, 45
PGE, 24
PKN Orlen, 70
PKO BP, 56
```

Analiza działania:

Blok with automatycznie zamyka plik. Iteracja po pliku pozwala na odczyt każdej linii osobno.

Wnioski:

To najbezpieczniejszy sposób obsługi plików w Pythonie.

B)

Przebieg zadania:

Trzeba było dopisać nową linię do pliku i ponownie go odczytać.

Kod programu:

```
5   #b
6   with open("notowania_gieldowe.txt", "a") as plik:
7       plik.write("\nALR, 113")
8
9   with open("notowania_gieldowe.txt", "r") as plik:
10      print(plik.read())
```

Przykładowy wynik programu:

```
KGHM, 123
Tauron, 150
Orange, 45
PGE, 24
PKN Orlen, 70
PKO BP, 56
ALR, 113
```

Analiza działania:

Tryb a dopisuje zawartość na koniec. Ponowne otwarcie pliku umożliwia odczyt zaktualizowanej treści.

Wnioski:

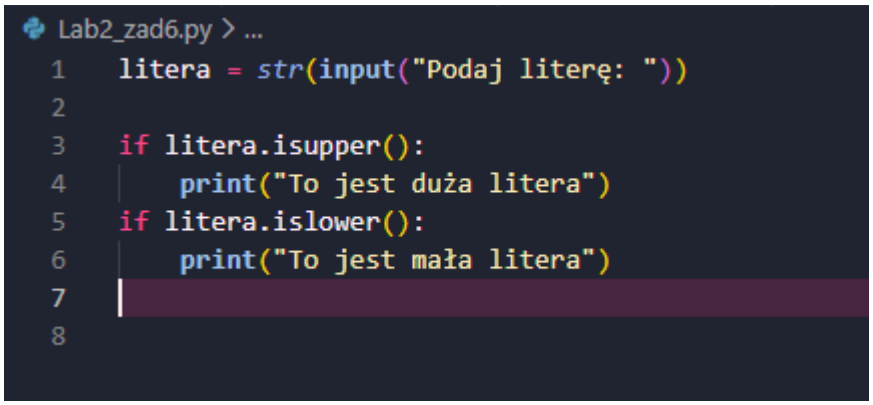
Zapis i odczyt pliku w Pythonie są intuicyjne i bazują na prostych trybach pracy.

ZADANIE 6

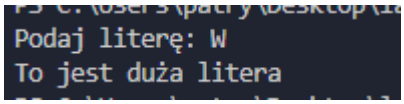
Przebieg zadania:

Program miał sprawdzić czy wprowadzona litera jest duża czy mała.

Kod programu:



Przykładowy wynik programu:



Analiza działania:

Metody isupper() i islower() pozwalają łatwo określić wielkość liter.

Wnioski:

Program poprawnie reaguje także na znaki nie będące literami.

ZADANIE 7

Przebieg zadania:

Należało sprawdzić, czy hasło:

- ma długość 11 znaków
- zawiera znak !

Kod programu:

```python
Haslo = 'pk47!jy0893'

if len(Haslo) == 11 and '!' in Haslo:
    print("Hasło jest poprawne")
else:
    print("Hasło jest niepoprawne")

```

Przykładowy wynik programu:

```
Hasło jest poprawne
```

Analiza działania:

Warunek korzysta z operatora logicznego and, co wymusza spełnienie obu kryteriów.

Wnioski:

To podstawowy przykład walidacji hasła w Pythonie.

ZADANIE 8

Przebieg zadania:

Celem było pobranie pierwszych trzech oraz ostatnich dwóch znaków tekstu.

Kod programu:

```python
text = 'Studiuję-Informatykę'

pierwsze_trzy = text[:3]
ostatnie_dwa = text[-2:]

print("Pierwsze trzy znaki:", pierwsze_trzy)
print("Ostatnie dwa znaki:", ostatnie_dwa)
```

Przykładowy wynik programu:

```
Pierwsze trzy znaki: Stu
Ostatnie dwa znaki: kę
```

Analiza programu:

Wykorzystano operator wycinania [start:stop] oraz indeksy ujemne.

Wnioski:

Slicing jest bardzo prosty i pozwala łatwo operować na ciągach znaków.

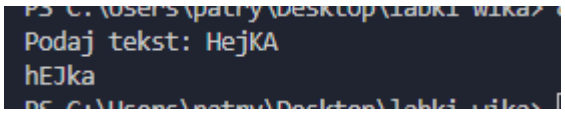
ZADANIE 9

Przebieg zadania:

Należało zamienić małe litery na duże i odwrotnie.

Kod programu:

Przykładowy wynik programu:



Analiza programu:

Metoda swapcase() zamienia każdą literę na przeciwną wersję.

Wnioski:

Gotowa metoda znacząco upraszcza takie operacje na tekście.