KURS PROGRAMOWANIA W JĘZYKU PYTHON

TYDZIEŃ 3 – FUNKCJE

1. Instrukcje warunkowe

Wcięcia

W Pythonie, wcięcia są kluczowe: definiują zakres pętli i funkcji, rozróżniając bloki kodu, które należą do siebie. Wcięcia w Pythonie czynią kod źródłowy czytelny i czysty, ponieważ wizualnie oddziela bloki kodu bez potrzeby stosowania dodatkowej składni, takiej jak nawiasy klamrowe używane w wielu innych językach programowania. Nieprawidłowe wcięcia mogą jednak prowadzić do błędów lub nieoczekiwanego zachowania!

If, else

Instrukcja warunkowa if-else służy do wykonywania różnych bloków kodu w zależności od spełnienia określonego warunku. Jeśli warunek jest prawdziwy (True), wykonuje się kod zawarty w bloku if, w przeciwnym razie, jeśli warunek jest fałszywy (False), wykonuje się kod zawarty w bloku else. Jest to przydatne, gdy chcemy, aby program zachowywał się różnie w różnych sytuacjach, np. wykonanie różnych czynności w zależności od wartości zmiennej.

```
1 wiek = 20
2
3 if wiek >= 18:
4     print("Możesz kupić piwo!")
5 else:
6     print("Nie możesz kupić piwa :(")
7
8
```

2. Pętle

Pętla while

Pętla while służy do wielokrotnego wykonywania bloku instrukcji tak długo, jak określony przez programistę warunek jest prawdziwy. Jest to szczególnie korzystne w sytuacjach, w których liczba iteracji (czyli ilość powtórzeń wykonania instrukcji) nie jest znana z góry, np. oczekiwanie na dane wejściowe użytkownika lub przetwarzanie danych do momentu spełnienia określonego warunku. Wadą pętli while jest ryzyko utworzenia nieskończonej pętli, jeśli warunek zakończenia nigdy nie zostanie spełniony. Typowe przykłady użycia pętli while to ponawianie prób w żądaniach sieciowych, odczytywanie ze strumienia danych, gdzie koniec nie jest z góry określony oraz menu, które działają do momentu, aż użytkownik zdecyduje się je zakończyć.

```
11 while wiek < 18:
12     print("Nie możesz kupić piwa :( Wróć jak dorośniesz")
13     wiek = int(input("Podaj swój wiek: "))
14 
15 #po wyjściu z pętli użytkownik musi mieć ukończone 18 lat
16 print("Możesz kupić piwo!")
17 
```

Pętla for

Pętla for również służy do wielokrotnego wykonywania bloku instrukcji, ale w przeciwieństwie do pętli while, określa się w niej ilość iteracji przed rozpoczęciem działania pętli. Jest to szczególnie przydatne, gdy liczba iteracji jest znana z góry lub gdy potrzebujemy wykonać określoną liczbę powtórzeń. Wadą pętli for może być ograniczenie w elastyczności, ponieważ trzeba z góry znać liczbę iteracji, co może być trudne w przypadku, gdy liczba ta jest zmienna lub zależy od warunków działania programu. Jednakże, dzięki swojej przejrzystości, pętla for jest często używana do iteracji po sekwencjach danych, takich jak listy, krotki czy słowniki. Typowe przykłady użycia pętli for to przeglądanie elementów listy, analiza danych z tablicy dwuwymiarowej, wykonywanie operacji na każdym elemencie sekwencji oraz iteracja po indeksach sekwencji w celu dostępu do ich wartości.

```
19 #zakres wieku od 0 do 17 lat
20 for wiek in range(18):
21     print(f"Masz {wiek} lat.")
22     print("Nie możesz kupić piwa.")
23 
```

3. Funkcje

Funkcje w Pythonie są definiowane za pomocą słowa kluczowego def i są używane do zamykania kodu w bloki wielokrotnego użytku, dzięki czemu programy są krótsze, bardziej czytelne i łatwiejsze do modyfikacji. Jedną z głównych zalet korzystania z funkcji jest lepsza organizacja struktury kodu przez oddzielenie różnych funkcjonalności programu od siebie. Jednak potencjalną wadą, w przypadku niewłaściwego użycia, jest zwiększone zużycie pamięci, np. jeśli do funkcji przekazywanych jest wiele dużych obiektów. Funkcje są przede wszystkim używane do wielokrotnego wykonywanie określonych obliczeń i automatyzację powtarzalnych zadań.

```
24 def sprawdz_wiek(wiek):
25     if wiek >= 18:
26         return "Jesteś pełnoletni."
27     else:
28         return "Jesteś niepełnoletni."
29
30 #Wywołanie funkcji
31 print(sprawdz_wiek(20))  #Wyświetli: Jesteś pełnoletni.
32 print(sprawdz_wiek(15))  #Wyświetli: Jesteś niepełnoletni.
```

4. Zmienne lokalne

Zmienne lokalne w Pythonie są deklarowane w ramach funkcji lub bloku i są dostępne tylko w tym określonym zakresie, przez co są "niewidoczne" dla pozostałych części programu. Taki zakres pomaga zapobiegać niezamierzonej modyfikacji danych. Ponieważ zmienne lokalne są rozpoznawane tylko wewnątrz ich funkcji lub bloku, próba uzyskania do nich dostępu poza ich zdefiniowanym zakresem spowoduje błąd!