**Temat 2**

**Zadanie 1.** Pole koła

**import** math   
  
r\_input = input(**'Podaj promień: '**)  
r\_value = float(r\_input)  
result = math.pi \* r\_value \*\* 2  
print(result)

Importuję moduł **math** zawierającą zdefiniowaną zmienną pi.

*Więcej o importowaniu modułów (*[*https://pl.python.org/docs/tut/node8.html*](https://pl.python.org/docs/tut/node8.html)*)*

Przypisuje zmiennej **r\_input** wartość podaną przez użytkownika.

Uwaga! input(„Zachęta: ”) zwraca napis. **r\_input** to string

Rzutuje **r\_input** na typ zmiennoprzecinkowy (float). nowyTyp(zmienna) „zmienia” typ zmiennej na nowyTyp.

Wyliczam **result** jako iloczyn pi (**math.pi** = 3.141592653589793) i zmiennej typu float podniesionej do kwadratu. Python wykonuje obliczenia w poprawnej kolejności, nie bezmyślnie od lewej do prawej.

*Można pominąć import math i zamiast math.pi wpisać 3.14.*

Wypisuje zmienną result funkcją **print().** Funkcja print przyjmuje jako argument dowolny obiekt

**Zadanie 2.** Suma kwadratów

sum=0  
**for** number **in** range(1,11):  
 sum += number \*\* 2  
print(sum)

Definiuje zmienną **sum** w której będę przechowywał wartości sumy n kolejnych kwadratów.

Tworzę pętlę **for**, number przyjmuje wartości całkowite z zakresu **[1;11)**

*(range(a,b) zwraca wartości całkowite z przedziału [a,b). range(a) zwraca wartości całkowite z przedziału [0;a), w tym przypadku wystarczyłaby konstrukcja range(11) ponieważ wliczenie kwadratu liczby zero nie wpływa na wynik)*

W każdym obiegu pętli do zmiennej **sum** dodaje zmienną całkowitą **number** podniesioną do kwadratu.

Wypisuje zmienną **sum** – sumę kwadratów dziesięciu pierwszych liczb naturalnych.

**Zadanie 3.** Minimum

minimum = float(input(**'Podaj liczbe: '**))  
**for** \_ **in** range(9):  
 number = float(input(**'Podaj liczbe: '**))  
 **if** number < minimum:  
 minimum = number   
print(minimum)

Definiuję zmienną **minimum** i wczytuje jej wartość z klawiatury rzutując ją ze stringa na float.

Tworzę pętlę **for**, nie potrzebujemy informacji o numerze obiegu pętli, stąd **for \_**, ponieważ wczytaliśmy pierwszą z 10 liczb wcześniej, pętla wykonuje 9 obiegów. [0,9)

W każdym obiegu pętli:

* Wczytuje zmienną number z klawiatury rzutując ją na float.
* Sprawdzam czy podana liczba jest mniejsza od **minimum**. Jeśli jest mniejsza, oznacza to że jest ona nową wartością minimalną, więc przypisuje ją do zmiennej **minimum**. W przypadku kiedy jest równa lub większa nic się nie zmienia.

Po 9 obiegach mamy pewność, że zmienna **minimum** wynosi tyle ile najmniejsza z podancyh liczb przez użytkownka. Wypisuje.

**Zadanie 4.** Zgaduj Zgadula

**import** random  
  
correct\_answer = random.randint(1,10)  
**for** \_ **in** range(3):  
 guess = int(input(**"Podaj liczbe: "**))  
 **if** guess == correct\_answer:  
 print(**"Wygraleś!"**)  
 **break  
 elif** guess > correct\_answer:  
 print(**"Za duża!"**)  
 **else**:  
 print(**"Za mala!"**)  
  
**else**:  
 print(**"Przegrałeś!"**)

Importuje moduł **random** służący do generowania liczb pseudolosowych.

Definiuje zmienną **correct\_answer** przypisując jej losową liczbę całkowitą z przedziału **[1;10]**.

*random.randint(a,b) zwraca liczbę całkowitą z przedziału [a;b]*

*Więcej o module random:* [*https://docs.python.org/3/library/random.html*](https://docs.python.org/3/library/random.html)

Tworzę pętlę **for** działającą 3 razy **[0,3)**, informacja o obiegu zbędna, dlatego **for \_**. Pętla zawiera konstrukcje **else**. Kod w else wykona się tylko wtedy gdy pętla „naturalnie” zakończy swoje działanie. W przypadku użycia **break** kod w **else** nie wykona się.

***break*** *natychmiast kończy działanie najbliższej pętli, w przypadku użycia* ***break*** *kod w* ***else****: nie wykona się*

**continue** przerywa obieg pętli i rozpoczyna następny

Zmiennej **guess** przypisuje wartość podaną przez użytkownika i rzutowaną na typ całkowity.

Sprawdzam:

Jeżeli **guess == correct\_answer**, oznacza że użytkownik podał szukaną wartość. Wypisuję komunikat o wygranej i przerywam działanie pętli.

Instrukcje pod **break** i **continue** nie wykonają się.

Jeżeli **guess > correct\_answer** wypisuje komunikat, że podana liczba jest *za duża*.

W pozostałych przypadkach (czyli **guess < correct\_answer**) wypisuje że podana liczba jest *za mała*

Jeśli w trzech obiegach pętli program nie wczyta z klawiatury poprawnej odpowiedzi pętla zakończy działanie samoistnie (bez użycia **break**) i przejdzie do kodu zawartego w klauzuli **else**. Wypisujemy *komunikat o przegranej*.

**Zadanie 5.** Sekundy na czas

number = int(input(**"Podaj liczbe"**))  
hours = number // 3600  
minutes = (number // 60) % 60  
seconds = number - hours \* 3600 - minutes \* 60  
print(hours, minutes, seconds, sep=**':'**)

Definiuje zmienną **number** i przypisuje jej wartośc liczbową podaną przez użytkownika, jest to liczba sekund.

Godzina to 3600 sekund. Wyliczam ilość pełnych godziny (**hours**) za pomocą dzielenia całkowitego (zaokrąglane w dół do liczb całkowitych)

Wyliczam ilość minut. Dzielę **number** bez reszty przez 60 (bo tyle sekund ma minuta), ale każde 60 minut stanowi pełne godziny, które już policzylismy. Uwzględniając to dzielę wartość modularnie przez 60.

Wyliczam ilość pozostałych sekund. Od początkowo podanej liczby sekund odejmuje godziny przeliczone na sekundy oraz minuty przeliczone na sekundy.

Wypisuje wynik.

***print(x1,x2,…,xn)*** *wypisuje zmienne dowolnego typu x1,x2,…,xn dzielone spacjami (domyślny separator).*

***print(x1,x2,…,xn, sep=”:”)*** *argument* ***sep=”:”*** *definiuje separator jako dwukropek. Można podać dowolny.*