Wrocław University of Science and Technology

Wstęp do programowania

INP003203L

Semestr zimowy 2020/2021

Poniedziałek, 7:30 - 9:00

sala wirtualna
– zajęcia online

Sylwia Majchrowska

sylwia.majchrowska@pwr.edu.pl

https://majsylw.netlify.app/teaching/

pokój 213, budynek L-1

Wrocław University of Science and Technology

Zasady komunikacji

- Spotkania wirtualne – ZOOM
- Korespondencja – email (sylwia.majchrowska@pwr.edu.pl)
- Zaliczenia – E-portal, odpowiedź ustna
- Czy coś jeszcze?
  - Discord
  - Kahoot, formularze googla

Wrocław University of Science and Technology

Zasady zaliczenia

- Kartkówki – 3 x 18 punktów
- Aktywność – 6 punktów
- Kolokwium końcowe – 40 punktów

Warunki konieczne zaliczenia:

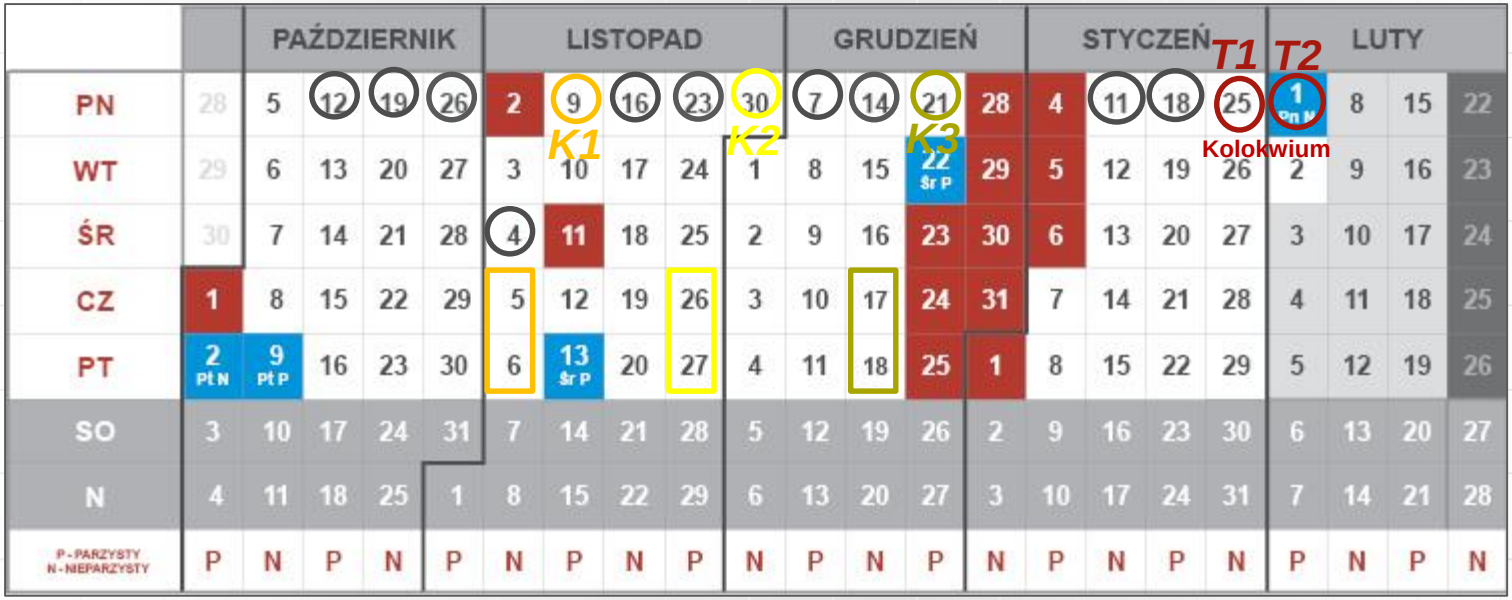
– uzyskanie co najmniej 50 punktów łącznie

– uzyskanie co najmniej 20 punktów na kolokwium

Obecność na zajęciach jest obowiązkowa
– dozwolone 2 nieusprawiedliwione nieobecności.

Wrocław University of Science and Technology

Spojrzenie na kalendarz



Poniedziałek, 7:30 - 9:00
sala wirtualna

Konsultacje:
Poniedziałki, 19:30 - 20:30
Środy, 19:30 – 20:30

Wrocław University of Science and Technology

Plan prezentacji

1. Projektowanie programu:
   - IPO – input, processing, output,
   - sposoby prezentacji programu.
2. Operacje wyjścia – funkcja **print**():
   - funkcje w życiu programisty,
   - znaki sterujące,
   - argumenty pozycyjne i nazwane.
3. Komentarze.
4. Zmienne i typy danych:
   - Łańcuchy znaków: str
   - Liczby: całkowite (int), zmiennoprzecinkowe (float)
5. Operacje wejścia – funkcja **input()**.

Wrocław University of Science and Technology

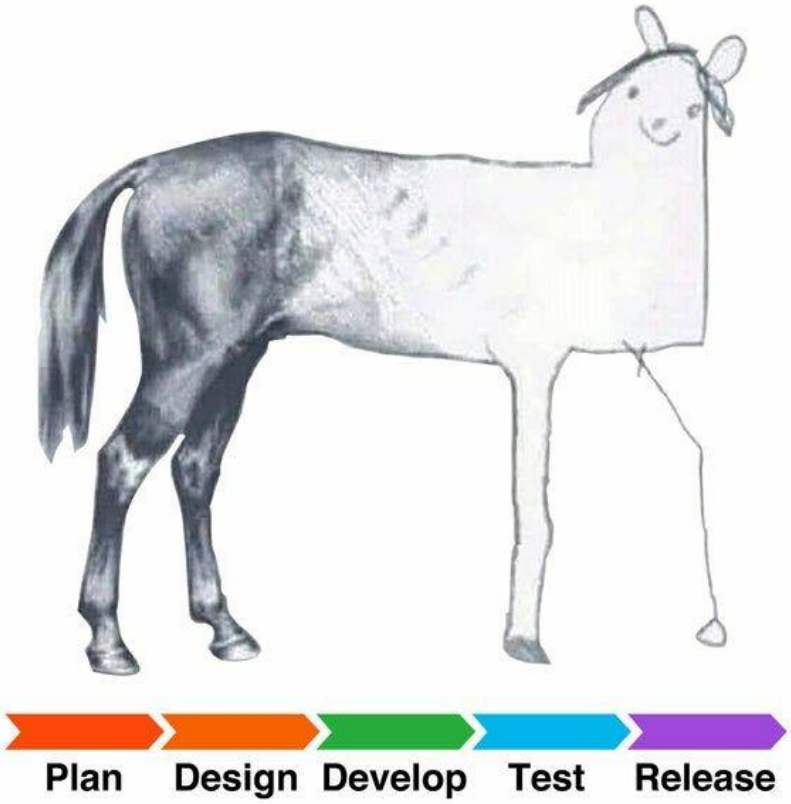
Projektowanie programu

Co musimy wiedzieć?

1. Jakie mamy dane wejściowe.
2. Co z nimi chcemy zrobić.
3. Jak przedstawiamy wynik.

Projektowanie programu można sprowadzić do dwóch kroków:

- określenie, jakie zadania ma wykonywać program,
- określenie kroków, za pomocą których program wykona to zadanie.



Wrocław University of Science and Technology

Projektowanie programu - sposoby zapisu

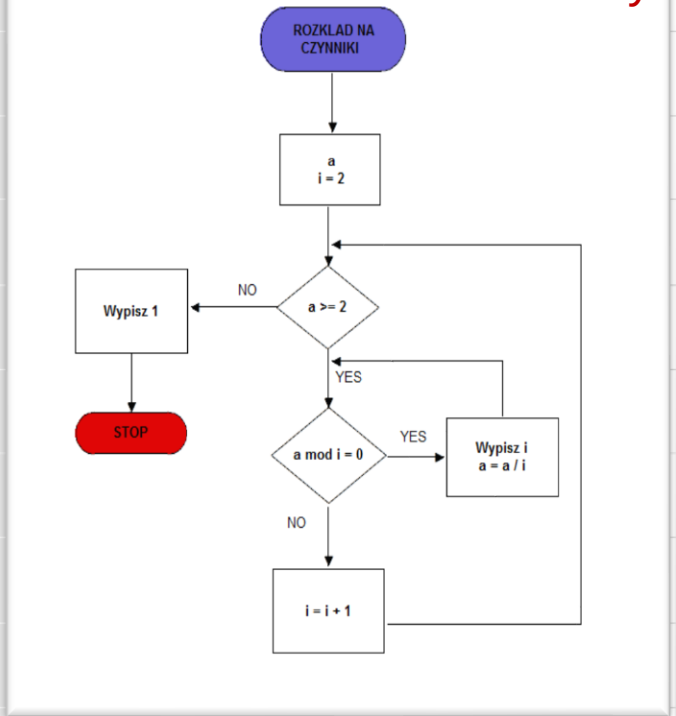
Lista kroków

1. Przypisz danej „i” wartość 2 i idź do kroku nr 2.
2. Jeżeli „a” jest większe, bądź równe liczbie 2 przejdź do kroku nr 3.
3. Jeżeli „a” jest podzielne przez „i” przejdź do kroku nr 4.
4. Wypisz „i”, a potem idź do kroku nr 5.
5. Przypisz „a” wartość równą ilorazowi „a” przez „i”, a potem wróć do kroku nr 3.
6. Jeżeli „a” nie jest podzielne przez „i” to dodaj 1 do „i” i wróć do kroku nr 2.
7. Jeżeli „a” jest mniejsze od 2 to wypisz 1 i zakończ algorytm.

Pseudokod

```
i←2
dopóki a>=2
        dopóki reszta z dzielenia „a” przez „i” równa się 0
                wypisz „i”
                a←a/i
        i←i+1
wypisz
```

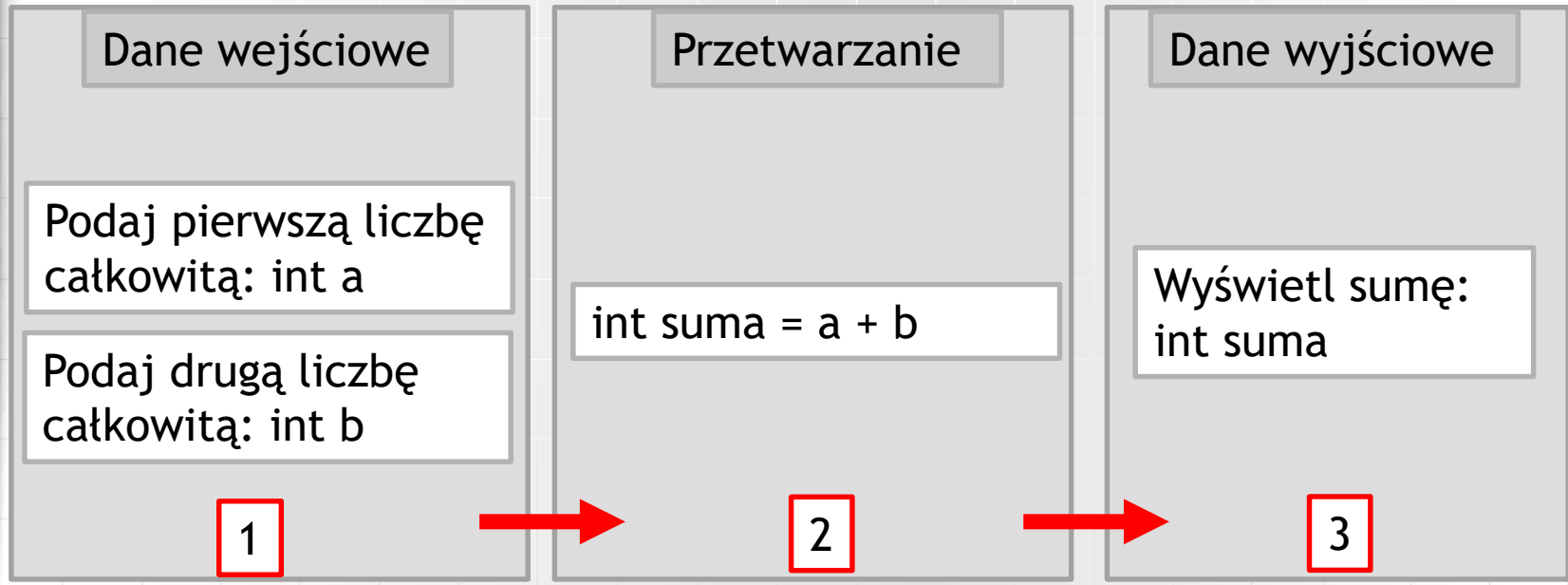
Schemat blokowy



Wrocław University of Science and Technology

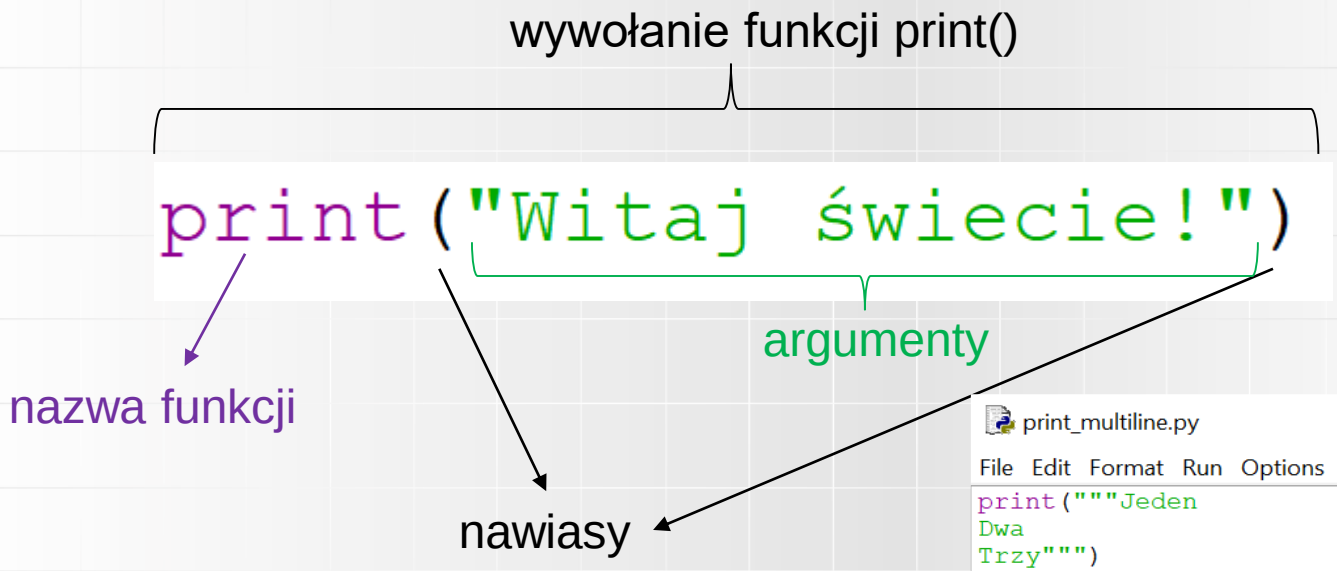
Projektowanie programu - przykład

Napisz program, który przyjmuje 2 liczby całkowite, a następnie wylicza ich sumę. Wynik powinien zostać wypisany na ekranie.



Wrocław University of Science and Technology

Wypisywanie komunikatów - funkcja print()



print_multiline.py

File Edit Format Run Options Window Help

```
print("""Jeden
Dwa
Trzy""")
```

Funkcja to oddzielna część kodu komputerowego, która może:

- **wywołać jakiś efekt** (np. wysłać tekst do terminala, stworzyć plik, narysować obrazek, odtworzyć dźwięk itp.); jest to coś zupełnie niespotykanego w świecie matematyki;
- **obliczyć jakaś wartość** lub wartości (np. pierwiastek kwadratowy z wartości lub długość danego tekstu); to właśnie sprawia, że funkcje programistyczne są krewnymi pojęć matematycznych.

Wrocław University of Science and Technology

Skąd się biorą funkcje?

- funkcje wbudowane (ang. ***build-in functions***) - taka funkcja jest wartością dodaną otrzymywaną razem z interpreterem pythona i jego środowiskiem; nie musisz robić nic specjalnego (np. prosić kogokolwiek o cokolwiek), jeśli chcesz z niej skorzystać, np. funkcja print();
- mogą pochodzić z jednego lub więcej nazwanych modułów dodatkowych języka python; niektóre moduły są dostarczane z interpreterem, inne mogą wymagać oddzielnej instalacji - w każdym przypadku wszystkie muszą być jawnie połączone z Twoim kodem (wkrótce pokażemy, jak to zrobić);
- możesz napisać je samodzielnie, umieszczając w programie tyle funkcji, ile chcesz i potrzebujesz, aby kod był prostszy, bardziej przejrzysty i elegancki.

Nazwa funkcji powinna być samotłumacząca się (nazwa funkcji print() jest oczywista).

Oczywiście, jeśli zamierzasz wykorzystać jakąkolwiek już istniejącą funkcję, nie masz wpływu na jej nazwę, ale kiedy zaczynasz pisać własne funkcje, powinieneś dokładnie przemyśleć wybór nazwy.

Wrocław University of Science and Technology

Funkcja print() - argumenty

Funkcje matematyczne zwykle przyjmują jeden argument, np. sin (x) przyjmuje x, który jest miarą kąta.

Z drugiej strony funkcje pythona są bardziej wszechstronne. W zależności od indywidualnych potrzeb mogą przyjąć dowolną liczbę argumentów - tyle, ile potrzeba do wykonania powierzonych im zadań.

Uwaga: Niektóre funkcje nie wymagają żadnych argumentów.

Funkcje w programowaniu zdecydowanie wymagają obecności pary nawiasów - odpowiednio otwierającego i zamykającego.

Wrocław University of Science and Technology

Funkcja print() - przykłady - rozwiązanie

- Użyj funkcji print(), aby wydrukować Hello, Python! na ekranie. Teraz popełni literówkę np. pint() – co się stało?

```
>>> pint("Hello, Python!")
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    pint("Hello, Python!")
NameError: name 'pint' is not defined
>>>
```

Niezdefiniowana funkcja

- Wróć do działającego kodu. Usuń cudzysłowy i uruchom go. Jaki rodzaj błędu jest zgłaszany?

```
>>> print(Hello, Python!)
SyntaxError: invalid syntax
>>>
```

Nieznane wywołanie

Wrocław University of Science and Technology

Funkcja print()
- przykłady – rozwiązanie cd.

- Usuń nawiasy, umieść z powrotem cudzysłowy i ponownie uruchom kod. Jaki rodzaj błędu jest zgłaszany tym razem?

```
>>> print"Hello, Python!"
SyntaxError: invalid syntax
>>>
```

Nieznane wywołanie

- Użyj funkcji print() bez widocznych argumentów – co się stało?

```
>>> print()

>>>
```

Wydrukowano pustą linię

- Eksperymentuj tyle, ile możesz. Zmień podwójne cudzysłowy na pojedyncze cudzysłowy, użyj wielu funkcji print() w tym samym wierszu, a następnie w różnych wierszach. Zobacz co się dzieje.

Wrocław University of Science and Technology

Funkcja print() - schemat działania

1. Najpierw interpreter sprawdza, czy podana nazwa jest prawidłowa - przegląda swoje wewnętrzne dane w celu znalezienia istniejącej funkcji tej nazwy; jeśli to wyszukiwanie nie powiedzie się, zgłaszany jest błąd – niepoprawnej nazwy;
2. W kolejnym kroku sprawdzane są, czy wymagania funkcji dotyczące liczby argumentów pozwalają na jej wywołanie, np. jeśli określona funkcja wymaga dokładnie dwóch argumentów, każde wywołanie dostarczające tylko jeden argument będzie uważane za błędne i spowoduje przerwanie jej wykonania;
3. Po trzecie, interpreter przejdzie do wnętrza funkcji, którą chcesz wywołać; oczywiście pobiera również argumenty i przekazuje je do jej wnętrza – jeśli tu pojawi się jakiś błąd przerwie działanie;
4. Po czwarte, wykonywany jest kod wewnątrz funkcji;
5. Na koniec Python wraca do kodu bazowego (do miejsca zaraz po wywołaniu) i wznawia swoje wykonanie.

Wrocław University of Science and Technology

Funkcja print() - znaki sterujące

Spróbujmy z poniższym kodem – zwróć uwagę, w jakiej kolejności pojawiają się napisy.

```
1  print("The itsy bitsy spider climbed up the waterspout.")
2  print()
3  print("Down came the rain and washed the spider out.")
```

Istnieje także inny sposób aby zasygnalizować „przerwę” – znaki sterujące:

- Znak specjalny końca linii *'\n'*,
- Znak specjalny tabulacji *'\t'*,
- Znak specjalny backspace *'\b'*,
- Znak apostrofu \',
- Znak cudzysłowu \",
- Ukośnik \\.

Wrocław University of Science and Technology

Funkcja print() - znaki sterujące - przykład

Napisz program wypisujący na ekranie tabelę przedrostków SI zawierającą w kolejnych kolumnach symbol, nazwę, wykładnik mnożnika oraz mnożnika od femto- do peta-. Efekt działania programu powinien być następujący:

symbol	nazwa	wykladnik	mnoznik
f	femto	-15	1e-015
p	piko	-12	1e-012
n	nano	-9	1e-009
mu	mikro	-6	1e-006
m	mili	-3	0.001
c	centy	-2	0.01
d	decy	-1	0.1
da	deka	1	10
h	hekto	2	100
k	kilo	3	1000
M	mega	6	1e+006
G	giga	9	1e+009
T	tera	12	1e+012
P	peta	15	1e+015

```
print('symbol\tnazwa\twykładnik\tmnoznik')
print('f\tfemto\t-15\t1e-15')
…
```

Czy jesteś w stanie bardziej upodobnić się do tabeli po lewej?

O tym będzie na kolejnych zajęciach ;-)

Wrocław University of Science and Technology

Funkcja print() - kilka argumentów

Spróbujmy z poniższym kodem – zwróć uwagę, w jakiej kolejności pojawiają się napisy.

- Do funkcji print() istnieje możliwość przekazania kilku napisów rozdzielając je przecinkiem.

```
1  print("The itsy bitsy spider" , "climbed up" , "the waterspout.")
2
3
```

- W ten sposób można też przekazywać literały różnego typu.

```
>>> print("Ala ma",2,"koty.")
Ala ma 2 koty.
```

- Wprowadzone po przecinku dane w wypisywanym komunikacie oddzielone są za pomocą spacji.

Wrocław University of Science and Technology

Funkcja print()
- sposoby na przekazanie argumentów

```
1  print("My name is", "Python.", end=" ")
2  print("Monty Python.")
3
```

- Przekazywanie argumentów względem ich pozycji:
  - Kolejność przekazywanych argumentów musi opowiadać ich kolejności w definicji funkcji; wszelkie argumenty nazwane należy umieścić po ostatnim argumencie pozycyjnym.
- Przekazywanie argumentów względem ich nazwy:
  - Przekazywany argument nazwany składa się z 3 części – nazwy argumentu, operatora przypisania (=) oraz wartości przypisywanej do tego argumentu

Funkcja print: sep, end, file

https://docs.python.org/3.8/library/functions.html#print

Wrocław University of Science and Technology

Funkcja print() – argumenty nazwane - przykład

Przekształć poniższy kod:

```
print("Programming","Essentials","in")
print("Python")
```

aby uzyskać na wyjściu ciąg:

```
Programming***Essentials***in...Python
```

```
print("Programming","Essentials","in", sep="***", end="...")
print("Python")
```

Wrocław University of Science and Technology

Funkcja print() – argumenty nazwane - przykład

Podwój strzałkę – tak aby dwie strzałki wystąpiły obok siebie. Postaraj się użyć tylko jednej funkcji print().

```
1  print("     *")
2  print("    * *")
3  print("   *    *")
4  print(" *      *")
5  print("***    ***")
6  print("   *    *")
7  print("   *    *")
8  print("   *****")
```

```
print("     *      "*2,
      "    * *     "*2,
      "   *    *   "*2,
      " *      *   "*2,
      "***    *** "*2,
      "   *    *   "*2,
      "   *    *   "*2,
      "   *****    "*2, sep = "\n")
```

Wrocław University of Science and Technology

Komentarze – po co i jak?

- Komentarz –informacja umieszczona w kodzie źródłowym programu, która objaśnia jego działanie
- Komentarze są ignorowane przez interpreter
- Komentarze ułatwiają zrozumienie kodu
- W Pythonie komentarz to fragment tekstu, który zaczyna się od znaku # (krzyżyka) i rozciąga się do końca linii.

comment01.py

File Edit Format Run Options Window Help

```
#Ten program wyświetla informację
#o polecanej literaturze
print('Tony Gaddis')
print('Python dla zupełnie początkujących')
print('Helion, 2019')
```

comment02.py

File Edit Format Run Options Window Help

```
print('Tony Gaddis')                          #Wyświetlenie informacji o autorze
print('Python dla zupełnie początkujących')   #Wyświetlenie tytułu
print('Helion, 2019')                         #Wyświetlenie wydawcy i roku wydania
```

Wrocław University of Science and Technology

Komentarze – kiedy?

- Dobrzy, odpowiedzialni programiści opisują każdy ważny fragment kodu, np. wyjaśniając rolę zmiennych; chociaż trzeba stwierdzić, że najlepszym sposobem komentowania zmiennych jest nadawanie im jednoznacznych nazw – **samo komentujące się zmienne.**

```
1  # Ten program wylicza dlugosc przeciwprostokatnej trojkata prostokatnego
2  # a i b są jego przyprostokatnymi
3  a = 3.0
4  b = 4.0
5  c = (a ** 2 + b ** 2) ** 0.5  # korzystamy z ** aby podniesc zmienne do kwadratu
6  print("c =", c)
```

- Komentarze mogą być przydatne pod innym względem - możesz ich użyć do oznaczenia fragmentu kodu, który obecnie nie jest potrzebny z jakiegokolwiek powodu, np. podczas testowania.

```
1  # To jest test programu
2  x = 1
3  y = 2
4  # y = y + x
5  print(x + y)
```

Wrocław University of Science and Technology

Czym są zmienne?

Zmienna to miejsce w pamięci komputera reprezentowane przez określoną nazwę. Zmienna posiada **nazwę** i **wartość**.

wiek = 19

wiek ⟶ 19

przypisanie

= ⟶ operator przypisania

Jeśli chcesz nadać nazwę zmiennej, musisz przestrzegać kilku ścisłych zasad:

- nazwa zmiennej może składać się z wielkich lub małych liter, cyfr i znaku _ (podkreślenia)
- nazwa zmiennej musi zaczynać się od litery;
- znak podkreślenia jest uznawany za literę;
- duże i małe litery są traktowane jako różne znaki (Alicja i ALICE to te różne napisy);
- nazwa zmiennej nie może być żadnym ze słów zastrzeżonych w Pythonie (to tak zwane słowa kluczowe).

Wrocław University of Science and Technology

Czym są zmienne?

- przykład

Wskaż poprawnie i niepoprawne nazwy zmiennych:

Adiós_Señora, 10t, Exchange Rate, Einbahnstraße, del, переменная, ćma, t34, _lambda, lambda, var, TheNameIsSoLongThatYouCanGoWrong, _, return, sûr_la_mer, counter

Słowa kluczowe

```
['False', 'None', 'True', 'and', 'as', 'assert',
'break', 'class', 'continue', 'def', 'del',
'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with',
'yield']
```

Wrocław University of Science and Technology

Zmienne
- typy danych: str

- str lub string, czyli napis, łańcuch znakowy bądź literał znakowy *<class 'str'>*
- napis poznajemy głównie po tym, że jest zapisany w apostrofach (') lub cudzysłowie (")
- wewnątrz napisu mogą pojawić się znaki specjalne – znaki sterujące rozpoczynające się od '\', np. znak końca linii '\n'
- aby zakodować apostrof lub cudzysłów w łańcuchu znaków, możesz użyć ukośnika, np. 'I\'m happy' lub wykorzystać innego typu znakowego, np. "I'm happy"

Wrocław University of Science and Technology

Zmienne
- typy danych: int

- ***int()*** lub integer to liczba całkowita, zakres dla liczb ujemnych i dodatnich, bez części dziesiętnych
  <class 'int'>

- funkcja int posiada drugi argument base – wskazujący na system zapisu liczb (domyślnie jest to system dziesiętny), ale jeśli liczba całkowita zacznie się od 0o zostanie potraktowana jako liczba w systemie ósemkowym, a 0x → szestnastkowym, np.

```
print(0o123) # to 83 w systemie dziesiętnym
print(0x123) # to 291 w systemie dziesiętnym
```

- nie wpisujemy między liczby znaków, które nimi nie są*, np.
  11,111,111 lub 11.111.111 lub 11 111 111 (skutkuje błędem)
  *od Pythona 3.6 możliwym jest zapis 11_111_111

Wrocław University of Science and Technology

Zmienne
- typy danych: float

- ***float()***, czyli zmienna zmiennoprzecinkowa/liczba rzeczywista - w Pythonie separatorem dziesiętnym jest **.**
  <class 'float'>

- same części dziesiętne możemy zapisywać również jako .4 oraz 4. (UWAGA! 4 to nie będzie to samo co 4. jeśli patrzymy na typy) → **kropka czyni liczbę zmiennoprzecinkową**

- można także posłużyć się notacją naukową używając literki e, tutaj 3e8 (lub 3E8) oznacza 3 x 10^8 (UWAGA! Wykładnik (liczba za literą e) musi być liczbą całkowitą, a podstawa (liczba przed literą e) – może być liczbą całkowitą)

```
print(0.00000000000000000001) # 1e-22
```

Wrocław University of Science and Technology

Zmienne
- typy danych: bool

- bool lub boolean, wartość logiczna, tak/nie, prawda(True)/fałsz(False)
  <class 'bool'>

- Algebra boolowska – część algebry, w której używane są tylko dwie wartości True (zapisywana jako 1) oraz False (zapisywana jako 0) – nazwa pochodzi od matematyka Georga Boole

```
>> print(True > False) # True
>> print(True < False) # False
```

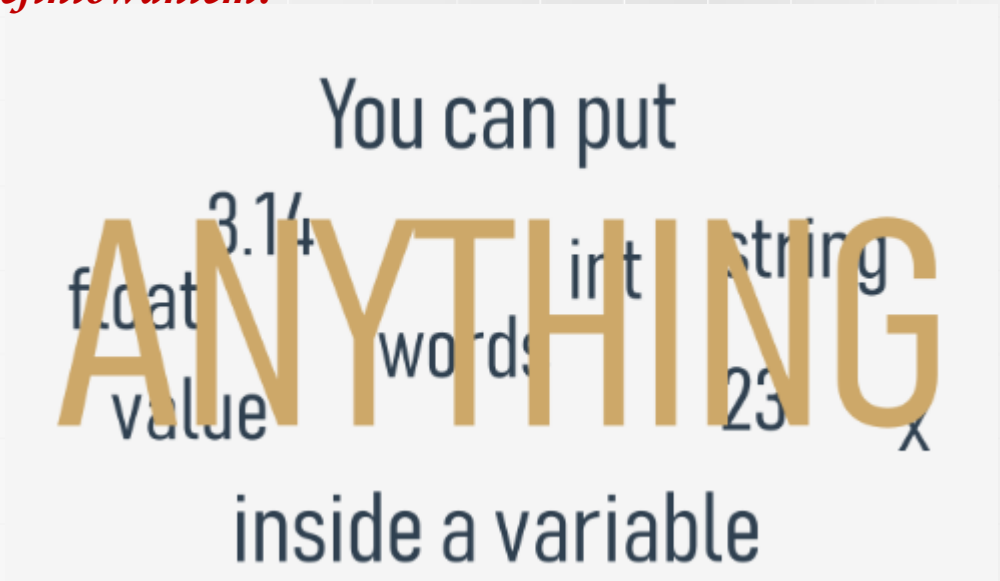
Jest jeszcze jeden specjalny literał używany w Pythonie: literał **None**. Ten literał jest tak zwanym obiektem ***NoneType*** i jest używany do reprezentowania braku wartości.

Wrocław
University
of Science
and Technology

Typy danych - przykład

```
>> a = 1
>> print(a) # teraz a jest 1
>> a = '1'
>> print(a) # tutaj a jest '1'
```

Nie można używać zmiennych przed ich zdefiniowaniem!



Wrocław
University
of Science
and Technology

Odczyt danych wejściowych - funkcja input()

→ `input()`

- Do pobierania danych wejściowych dostarczanych za pomocą klawiatury służy funkcja ***input()***
- Funkcja ***input()*** zwraca napis (literał znakowy), która aby wykorzystać go później należy przypisać do zmiennej
- Funkcja ***input()*** może pobierać dane w postaci napisu

```
>> print("Tell me anything...")
>> anything = input()
>> print("Hmm...", anything, "... Really?")
```

```
>> anything = input("Tell me anything...")
>> print("Hmm...", anything, "... Really?")
```

Wrocław
University
of Science
and Technology

Odczyt danych wejściowych - funkcja input() cd.

→ `input()`

- Aby zmienić typ danych (skonwertować dane) można wykorzystać funkcje ***int()*** oraz ***float()***

```
>> anything = float(input("Enter a number: "))
>> something = anything ** 2.0
>> print(anything, "to the power of 2 is", something)
```

Wrocław University of Science and Technology

Rzeczy do zapamiętania!

1. Funkcja ***print()*** jest funkcją wbudowaną, która drukuje określony komunikat w oknie konsoli.
2. Funkcje wbudowane, w przeciwieństwie do funkcji zdefiniowanych przez użytkownika, są zawsze dostępne i nie trzeba ich importować - pełną listę można znaleźć w porządku alfabetycznym w bibliotece standardowej (https://docs.python.org/3.8/library/functions.html).
3. Aby wywołać funkcję, musisz użyć nazwy funkcji, po której następuje nawias, wewnątrz którego wypisujemy przekazywane argumenty.
4. Ciągi znaków są ujmowane w cudzysłowu lub apostrofach, np. ”Jestem łańcuchem” lub ’Ja też jestem łańcuchem’.
5. Programy komputerowe to zbiory instrukcji. Instrukcja jest poleceniem wykonania określonego zadania, np. wydrukowania określonej wiadomości na ekranie.
6. W napisach ukośnik (\) jest znakiem specjalnym, który informuje, że następny znak ma inne znaczenie, np. \n (znak nowej linii) rozpoczyna nowy wiersz.

Wrocław University of Science and Technology

Rzeczy do zapamiętania!

7. Argumenty pozycyjne to takie, których znaczenie jest podyktowane ich pozycją, np. drugi argument jest wyprowadzany po pierwszym, trzeci po drugim itd.
8. Argumenty nazwane to takie, których znaczenie nie jest podyktowane ich lokalizacją, ale specjalnym słowem (słowem kluczowym) używanym do ich identyfikacji.
9. Argumenty *end* i *sep* mogą być używane do formatowania danych wyjściowych funkcji *print()*. Argument *sep* określa separator między argumentami wyjściowymi (np. print("H", "E", "L", "L", "O", sep = "-"), podczas gdy argument *end* określa, co wydrukować na końcu instrukcji drukowania.
10. Literały to notacje reprezentujące pewne ustalone wartości w kodzie. W pythonie mamy różne typy literałów - na przykład literał może być liczbą (literały numeryczne, np. 123) lub ciągiem znaków (literały łańcuchowe/znakowe, np. ”Jestem literałem”).
11. System binarny to system liczbowy, w którym podstawą jest 2. Dlatego liczba binarna składa się tylko z zer i jedynek, np. 1010 to 10 w systemie dziesiętnym.

Wrocław University of Science and Technology

Rzeczy do zapamiętania!

12. Podobnie, systemy numeracji ósemkowej i szesnastkowej stosują odpowiednio 8 i 16 jako podstawy systemów. System szesnastkowy używa liczb dziesiętnych i sześciu kolejnych liter z alfabetu.
13. Liczby całkowite (**int**) są jednym z typów liczbowych obsługiwanych przez Pythona. Są to liczby zapisane bez części ułamkowej, np. 256 lub -1 (liczby całkowite ujemne).
14. Liczby zmiennoprzecinkowe (**float**) to kolejny typ liczbowy obsługiwany przez Pythona. Są to liczby, które zawierają (lub mogą zawierać) składnik ułamkowy, np.1,27.
15. Aby zakodować apostrof lub cudzysłów w łańcuchu znaków, możesz użyć ukośnika, np. *'I\'m happy'* lub wykorzystać innego typu znakowego, np. *"I'm happy".*
16. Wartości logiczne to dwa stałe obiekty ***True*** i ***False*** (w kontekście liczbowym 1 to **True**, a 0 to **False**).
17. Jest jeszcze jeden specjalny literał używany w Pythonie: literał **None**. Ten literał jest tak zwanym obiektem ***NoneType*** i jest używany do reprezentowania braku wartości.