Zasady zaliczania
https://moodle.mimuw.edu.pl/mod/page/view.php?id=144251

Dzisiejsze zajęcia to wprowadzenie do narzędzi z których będziemy korzystali.

1) Wstęp – ciekawostki

a) Guido van Rossum, był fanem Latającego Cyrku Monty Pythona, stąd inspiracja dla nazwy.

b) Version 0.9.0 – luty 1991, version 1.0 – styczeń 1994 z narzędziami do programowania funkcyjnego; Obecnie Python 3.x

c) The Zen of Python (początek):
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.

d) popularność Pythona
- https://pypl.github.io/PYPL.html
- https://madnight.github.io/githut/#/pull_requests/2023/2

2) Python Interactive Shell – uruchamianie

a) otwieramy terminal „python" + Enter i w ten sposób uruchamiamy interpreter Pythona

```
Python 3.7.7 (default, Mar 26 2020, 15:48:22)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

b) aby wyjść ctrl+D
c) w terminalu „python" + tab +tab widzimy dostępne w systemie wersje pythona

```
python            python3.6-config   python3-config
python2           python3.6m         python3-jsonschema
python2.7         python3.6m-config  python3m
python2.7-config  python3.7          python3m-config
python2-config    python3.7-config   python-config
```

python3 python3.7m pythontex
python3.11 python3.7m-config pythontex3
python3.11-config python3.8
python3.6 python3.9

d) możemy uruchomić konkretną wersję np. pisząc w terminalu „python3.9"+Enter

3) Python Interactive Shell – hello world

a) uruchamiamy interpreter – w terminalu „python"+Enter
b) > hello world
zwróci nam błąd
c) > 'hello world'
i mamy zwrócony napis 'hello world'
d) > 2
podobnie jest dla liczb
e) > print('hello world')
to już wykorzystanie składni Pythona
f) > 2+2 > 2**50
działa jak kalkulator
g) > 'Nie będę się spóźniał' *50
można mnożyć napisy
h) > _ * 3
underscore to poprzednia pamiętana wartość
i) > licznik = 2
jak chcemy pamiętać
j) > mianownik = 4
k) > licznik / mianownik
zmiany od wersji 3.0 „The division of two integers returns a float instead of an integer."
l) > exit()
lub ctrl+D do wyjścia

4) Jupyter – hello world

Jupyter to system do tworzenia i edycji dokumentów on-line, które mają w miarę ładnie wyglądać i zawierać uruchamialne fragmenty programów (w szczególności w Pythonie). Do pracy z dokumentem używa się przeglądarki, ale "pod spodem" działa odpowiedni program (kernel), który robi robotę. Oczywiście jest możliwość skonfigurowania serwera http, aby udostępniał takie notebooki "na zewnątrz", patrz np. https://jupyter.org/try-jupyter

Niewykluczone, że część scenariuszy naszych zajęć będzie przygotowana właśnie w formie notebook'ów.

Jupyter lab to takie IDE przeglądarkowe / online do pracy z (wieloma naraz) notebookami oraz z możliwością importu i wizualizacji plików z danymi.

Jupyter lab to nowsza (w pełni wstecznie zgodna) wersja Jupyter notebooka (który z kolei powstał na bazie IPythona). Dla nas Jupyter Lab i Notebook są równie dobre (i właściwie nierozróżnialne :)). W przeciwieństwie do IPythona, który jest tekstowy i jego nie pokazujemy na naszych zajęciach.

Oczywiście dodatkowo Jupyter Notebook oznacza też dokument tworzony przez te narzędzia -- jak zwykle w informatyce terminologię mamy jeszcze mocno nieprecyzyjną :).

a) w terminalu~ **jupyter notebook**
otwieramy interpreter w przeglądarce
b) wybieramy new→python3
c) w okienku wpisujemy „2" i klikamy z górnego menu „run"
pojawia się „Out[1] : 2" podobnie jak wcześniej
d) plusem możemy stworzyć nową krotkę
e) "Nie będę się spóźniał " *50
f) podwójne kliknięcie z lewej strony krotki pozwoli schować wynik
g) plusem tworzymy nową krotkę
h) próbujemy 2**1000, 2**10000 … 2**10000000
i) przerwać obliczenia możemy przyciskiem stop (kwadracik)
j) plusem tworzymy nową krotkę i wpisujemy
licznik = 2
mianownik = 4
licznik / mianownik
k) plusem tworzymy nową krotkę
l) za pomocą strzałki możemy przenieść ją wyżej

Markdown to jeden z najpopularniejszych języków znaczników (ang. markup language), zawierających w tekście dodatkowe informacje, które go opisują. Inne popularne to m.in. latex, HTML. Więcej o Markdown: https://www.markdownguide.org/getting-started/

m) zamieniamy w menu typ krotki z „code" na „markdown" i wpisujemy

Ważne obliczenia

Oto nasz wzór:

licznik oznacza wartość `licznika`

mianownik to wartość `mianownika`

A wzór może być w latex-u: $\frac{x}{y}$

n) po naciśnięciu „run" mamy elegancki opis, podwójne kliknięcie i znów jesteśmy w trybie edycji
o) zamykamy okienko w przeglądarce
p) ctrl+c w terminalu a potem 'y'
r) w terminalu~ **jupyter lab**
s) wybieramy jądro „Python3" - na naszym kursie nauczymy się później tworzyć wirtualne środowiska, tak aby w tym miejscu móc wybrać własną konfigurację
t) Zauważmy, że w jupyter lab klikając w niebieski pasek po lewej stronie krotki, można zwijać nie tylko odpowiedzi, ale też i krotki z kodem. Co może być bardzo przydatne później w pracy.
u) wyjście analogiczne

5) plik.py – Hello world
a) uruchamiamy ulubiony edytor i tworzymy plik
~nano hw.py
b) cat hw.py
„Hello world"
c) ~python hw.py
i nic się nie wyświetla, to już nie interpreter
d) ~cat hw.py
print('Hello world')
e) ~python hw.py
Hello world
Zadziałał nasz pierwszy skrypt. Jak widzimy nie trzeba było wykonywać dodatkowych kroków, jak np. kompilacja programu.

W rzeczywistości Python jest jednocześnie 'an interpreted and a compiled language'. Kod Pythona jest tłumaczony na pośredni kod, wykonywalny przez maszynę wirtualną, the Python Virtual Machine (PVM). Jest to podobne podejście jak w Javie. Jython umożliwia tłumaczenie programów w Pythonie na byte code Javy dla Java Virtual Machine (JVM).

f) ls
g) python -m py_compile hw.py
h) ls
pojawiły się nowe pliki
i) cd __pycache__/

j) cat hw.cpython-37.pyc
tak wygląda byte code dla PVM

Wiedząc to, ustalmy, że nie musimy, a nawet nie powinniśmy myśleć o kompilacji kodu w Pythonie. Nawet jak Python nie ma dostępu (write) do katalogu, to program i tak się wykona. Byte code zostanie stworzony, ale po skończeniu programu zostanie usunięty.

Kiedy uruchamiamy program w Pythonie, sprawdza on czy skompilowana wersja .pyc istnieje, oraz czy jest to nowszy plik niż plik .py. Jeśli tak, Python wczyta byte code, co przyspieszy start programu. Jeśli nie, Python stworzy byte code przed wykonaniem programu.

6) import hw
Chcemy wykorzystać już napisany kod w pliku hw.py w innym skrypcie.
a) nano hello.py
b) cat hello.py
import hw
c) python hello.py
Hello world
Przy imporcie wykonał nam się cały kod z importowanego skryptu, co może prowadzić do problemów.
d) edytujemy
e) cat hw.py
```
def greet():
    print("Hello world")
```
def – oznacza funkcję, w Pythonie ważne są wcięcia (więcej potem)
~cat hello.py
```
import hw

hw.greet()
#Jest to jedna z możliwości odwołania się do funkcji z innego pliku. Więcej potem.
```
f) teraz wciąż 'python hello.py' zwraca Hello world, ale 'python hw.py' już nie
g) cat hw.py
```
def greet():
    print("Hello world")
greet()
```
Ponieważ w Pythonie nie ma funkcji main(), kiedy uruchamiamy program kod na poziomie wcięcia 0 jest wykonywany.

Teraz 'python hw.py' zwraca „Hello world" ale 'python hello.py' zwraca „Hello world" dwukrotnie.

h) cat hw.py

```
def greet():
    print("Hello world")

if __name__ == '__main__':
    greet()
```

Teraz oba uruchomienia działają bez zarzutu.
I mamy wprowadzenie do Hello World generowanego przez IDE.

__name__ to tzw metoda specjalna, nazywana też magiczną lub dunder (od double underscore)

Ponieważ w Pythonie nie ma funkcji main(), kiedy uruchamiamy program kod na poziomie wcięcia 0 jest wykonywany. Jednak wcześniej kilka specjalnych zmiennych zostaje zdefiniowanych, __name__ jest jedną z nich. Jeśli plik źródłowy jest wykonywany jako główny program, interpreter ustawi zmienną __name__ na wartość "__main__". Natomiast jeśli plik jest importowany z innego modułu, zmienna __name__ przyjmuje wartość będącą nazwą modułu.

7) Hello world w IDE - PyCharm

a) Uruchamiamy PyCharm
File→New Project→ Pure Python → Create

Uruchomić nasz projekt możemy m.in.
- trójkątem w górnym pasku
- trójkątem w linii 13
- wybierając na dole okienko terminal i wpisując 'python main.py'

b) W 15 linii wpisujemy 4 spacje i „x=1"
W zakładce 'Problems' znajdziemy teraz → PEP 8: E225 missing whitespace around operator

Specyfikacja Python Enhancement Proposal #8 to styl określający sposób formatowania kodu utworzonego w Pythonie. Nasza wersja działa, ale ustandaryzowana wersja ma wiele zalet m.in. ułatwia współpracę w dużych projektach.

W linii 16 wpisujemy 4 spacje i „prit"
W zakładce problems mamy teraz czerwone oznaczenie błędu.

c) konfiguracja VCS
https://www.jetbrains.com/help/pycharm/version-control-integration.html

Małe zadanie 1:
oszacować wartość Pi metodą Monte Carlo.

Użytkownik podaje ilość iteracji (N) i krok (K). Program wypisuje wartość przybliżenia dla kolejnych kroków (K, 2K, … < N). Na koniec porównuje wartość przybliżenia dla N z wartością math.pi.

Wskazówka. Okręg o środku w 0,0 i promieniu r=1. Losujemy wartości współrzędnych x i y n razy i sprawdzamy, ile wylosowanych punktów jest w ćwiartce okręgu.