Functions

Funktsiyalar - bu kompyuter yoki kompyuter tili qanday bajarishni allaqachon biladigan harakatlar.

Hello.py dasturingizda chop etish funksiyasi terminal oynasiga qanday chop etishni biladi.

`print` funksiyasi argumentlarni oladi. Bunday holda, "Hello, World!" chop etish funktsiyasi qabul qiladigan argumentlardir.

Bugs

Bugs code ning tabiiy qismidir. Bu xatolar, siz hal qilishingiz kerak bo'lgan muammolar! Tushkunlikka tushmang! Bu ajoyib dasturchi bo'lish jarayonining bir qismidir.

Tasavvur qiling-a, hello.py dasturimizda kompilyator talab qilgan print("Hello, World") tasodifan qoldirdik. Agar men bu xatoni ataylab qilsam, kompilyator terminal oynasida xatolik chiqaradi!

Ko'pincha, xato xabarlari sizni xatongiz haqida xabardor qiladi va ularni qanday tuzatish bo'yicha maslahatlar beradi. Biroq, har doim ham emas.

```
input("What's your name? ")
print("hello, world")
```

Biroq, ushbu code o'zi dasturingizga foydalanuvchi kiritgan narsalarni chiqarishga ruxsat bermaydi. Buning uchun biz sizni o'zgaruvchilar(variables) bilan tanishtirishimiz kerak.

Variables

O'zgaruvchi shunchaki o'z dasturingizdagi qiymat uchun idish dur. Dasturingizda siz o'z o'zgaruvchingizni o'qish uchun `input` orqali dasturingizga kiritishingiz mumkin

```
name = input("What's your name? ")
print("hello, world")
```

E'tibor bering, bu teng = nom o'rtasida joylashgan belgi = `input("What’s your name?")` dasturlashda alohida rol o'ynaydi. Bu teng belgi tom ma'noda o'ngdagi narsani chap tomondagiga belgilaydi. Shuning uchun, `input("What’s your name?")` orqali qaytarilgan qiymat nomga tayinlanadi.

Agar siz kodingizni quyidagi tarzda ozgartirsangiz, xatolikni sezasiz

```
name = input("What's your name? ")
print("hello, name")
```

Dastur foydalanuvchi nima yozganidan qat'iy nazar terminal oynasida **“hello, name”** qaytaradi.

```
name = input("What's your name? ")
print("hello,")
print(name)
```

Quyidagi natija bayda boladi

```
What's your name? David
hello
David
```

Biz kutgan natijaga yaqinlashmoqdamiz!

Comments

Comment lar dasturchilar uchun o'z dasturlarida nima qilayotganlarini kuzatish va hattoki kod blokiga bo'lgan malumot haqida boshqalarni xabardor qilishning bir usuli hisoblanadi. Qisqasi, ular o'zingiz va boshqalar uchun eslatmalar!

Dasturingiz nima qilayotganini ko'rish uchun dasturingizga commentlar qo'shishingiz mumkin.

```
# Ask the user for their name
name = input("What's your name? ")
print("hello,")
print(name)
```

Commentlar siz uchun vazifalar ro'yxati sifatida ham xizmat qilishi mumkin.

Pseudocode

Pseudocode - bu ishlar ro'yxatining maxsus turiga aylanadigan muhim comment turi, ayniqsa siz kodlash vazifasini qanday bajarishni tushunmasangiz. Misol uchun, kodingizda siz o'z kodingizni quyidagicha ozgartirishingiz mumkin:

```
# Ask the user for their name
name = input("What's your name? ")

# Print hello
print("hello,")

# Print the name inputted
print(name)
```

```
# Ask the user for their name
name = input("What's your name? ")

# Print hello and the inputted name
print("hello, " + name)
```

Ma'lum bo'lishicha, ba'zi funktsiyalar ko'p argumentlarni qabul qiladi.

Kodimizni quyidagi tarzda tahrirlash orqali bir nechta argumentlarni kiritish uchun verguldan foydalanishimiz mumkin:

```
# Ask the user for their name
name = input("What's your name? ")

# Print hello and the inputted name
print("hello,", name)
```

String and Parameters

Pythonda **str** sifatida tanilgan **string** harflar ketma-ketligidir.

Bizning kodimizda bir oz orqaga qaytsak, natija bir nechta satrlarda paydo bo'lishini kordik:

```
# Ask the user for their name
name = input("What's your name? ")
print("hello,")
print(name)
```

Funktsiyalar ularning xatti-harakatlariga ta'sir qiluvchi `argumentlarni` qabul qiladi. Agar biz `print` ni qollanmasini korib chiqsak chiqsak, siz `print` etish funksiyasi oladigan argumentlar haqida ko'p narsalarni bilib olishingiz mumkinligini ko'rasiz.

Ushbu qollanma ni ko'rib chiqsangiz, `print` funksiyasi avtomatik ravishda `end='\n'` kod qismini o'z ichiga olishini bilib olasiz. Bu \n, `print` funksiyasi ishga tushganda avtomatik ravishda qator uzilishini yaratishini bildiradi. `print` e funksiyasi end` deb nomlangan argumentni oladi va standart yangi qator yaratishdir.

Biroq, biz texnik jihatdan o'zimiz uchun argument keltira olamiz, shunda yangi qator yaratilmaydi!

```
# Ask the user for their name
name = input("What's your name? ")
print("hello,", end="")
print(name)
```

`end=""` ni taqdim etish orqali biz `end` ning standart qiymatini ozgartiramiz, shunda u hech qachon ushbu birinchi `print` dan keyin yangi qator yaratmaydi. Ismni "David" deb ta'minlash, terminal oynasidagi chiqish `"Hello, David"` bo'ladi.

Demak, parametrlar funksiya tomonidan qabul qilinadigan argumentlardir.

Qo'shtirnoq bilan bog'liq kichik muammo

Stringni bir qismi sifatida tirnoq belgilarini qo'shish qanchalik qiyinligiga e'tibor bering.

`print("Hello,"friend"")` ishlamaydi va kompilyator xatoga yo'l qo'yadi.

Umuman olganda, buni tuzatish uchun ikkita yondashuv mavjud. Birinchidan, siz shunchaki qo'shtirnoqlarni bitta tirnoq belgilariga o'zgartirishingiz mumkin.

Yana bir ko'proq qo'llaniladigan yondashuv bu `("Hello, \"friend\"")`. Backward slash kompilyatorga quyidagi belgi qatordagi tirnoq belgisi sifatida qaralishi va kompilyator xatosidan qochish kerakligini aytadi.

Formatted Strings

Aslida, stringlardan foydalanishning eng oqlangan usuli quyidagicha bo'ladi:

```
# Ask the user for their name
name = input("What's your name? ")
print(f"hello, {name}")
```

`print` da f ga e'tibor bering `(f"Hello, {name}")`. Bu f Python uchun maxsus ko'rsatkich.

Sizning foydalanuvchiingiz mo'ljallangan tarzda malumot kiritishini qachon kutmasligingiz kerak. Shuning uchun siz foydalanuvchi kiritgan ma'lumotlar to'g'rilangan yoki tekshirilganligiga ishonch hosil qilishingiz kerak bo'ladi.

Ma'lum bo'lishicha, string da bosh joylarni ochirib tashlagash uchun hususiyati bor.

Stringdagi `strip` metod orqali biz bosh orinlarni ochirib tashlasak boladi, buni quyidagicha amalga oshiriladi `name = name.strip()`.

```
# Ask the user for their name
name = input("What's your name? ")

# Remove whitespace from the str
name = name.strip()

# Print the output
print(f"hello, {name}")
```

`title` metodi orqali biz titul formatda yozsak boladi.

```
# Ask the user for their name
name = input("What's your name? ")

# Remove whitespace from the str
name = name.strip()

# Capitalize the first letter of each word
name = name.title()

# Print the output
print(f"hello, {name}")
```

```
# Ask the user for their name
name = input("What's your name? ")

# Remove whitespace from the str and capitalize the first letter of each word
name = name.strip().title()

# Print the output
print(f"hello, {name}")
```

```
# Ask the user for their name, remove whitespace from the str and capitalize the first
letter of each word
name = input("What's your name? ").strip().title()

# Print the output
print(f"hello, {name}")
```

Integers

Pythonda butun son int (integer) deb ataladi.

Matematika olamida biz +, -, *, / va % operatorlari bilan tanishmiz. Oxirgi % operatori yoki modul operatori sizga unchalik tanish bo‘lmasligi mumkin.

Python kodini ishga tushirish uchun kompilyatorda matn muharriri oynasidan foydalanish shart emas. Terminalingizda siz **python**-ni yolg'iz ishlatishingiz mumkin. Terminal oynasida sizga >>> ko'rsatiladi. Keyin jonli, interaktiv kodni ishga tushirishingiz mumkin. Siz 1+1 ni yozishingiz mumkin va u hisob-kitobni amalga oshiradi. Kurs davomida bu rejim odatda ishlatilmaydi.

terminalga code calculator.py kodini kiritishimiz mumkin. Bu yangi faylni yaratadi, unda biz o'z kalkulyatorimizni yaratamiz.

Birinchidan, biz bir nechta o'zgaruvchilarni yaratishimiz mumkin.

```
x = 1
y = 2

z = x + y

print(z)
```

Tabiiyki, biz python calculator.py ni ishga tushirganimizda, natijani 3 terminal oynasida olamiz. Biz input funksiyasidan foydalanib, buni yanada interaktiv qilishimiz mumkin.

```
x = input("What's x? ")
y = input("What's y? ")

z = x + y

print(z)
```

Ushbu dasturni ishga tushirib, biz chiqish 12 sifatida noto'g'ri ekanligini aniqlaymiz. Nima uchun bu bo'lishi mumkin?

Avvalroq, biz + belgisi ikkita string ni qanday birlashtirganini ko'rib chiqdik. Kompyuteringizdagi klaviaturadan kiritgan ma'lumotlar kompilyatorga matn sifatida kelganligi sababli, u string sifatida ko'rib chiqiladi. Shuning uchun biz ushbu kiritishni stringdan butun songa aylantirishimiz kerak. Biz buni quyidagicha qilishimiz mumkin:

```
x = input("What's x? ")
y = input("What's y? ")

z = int(x) + int(y)

print(z)
```

Natija endi to'g'ri. `int(x)` dan foydalanish “casting” deb ataladi, bunda qiymat vaqtincha bir turdagi o‘zgaruvchidan (bu holda string) boshqasiga (bu yerda butun son) o‘zgartiriladi.

Biz dasturimizni quyidagi tarzda yanada takomillashtirishimiz mumkin:

```
x = int(input("What's x? "))
y = int(input("What's y? "))

print(x + y)
```

Bu funksiyalarni funksiyalarda ishga tushirishingiz mumkinligini ko'rsatadi. Avval eng ichki funksiya ishga tushiriladi, keyin esa tashqi funksiya ishga tushiriladi. Birinchidan, input funktsiyasi ishga tushiriladi. Keyin int funktsiyasi.

Kodlash vazifasiga yondashuvingiz haqida qaror qabul qilayotganda, bir xil muammoga ko'plab yondashuvlar uchun asosli dalillar keltirish mumkinligini unutmang.

Dasturlash topshirig'iga qanday yondashishingizdan qat'i nazar, kodingiz o'qilishi kerakligini unutmang. O'zingizga va boshqalarga kodingiz nima qilayotgani haqida maslahatlar berish uchun comment lardan foydalanishingiz kerak. Bundan tashqari, kodni o'qilishi mumkin bo'lgan tarzda yaratishingiz kerak.

Float

Kasr son python da float nuqta qiymati - 0,52 kabi o'nli kasrga ega bo'lgan haqiqiy son.

Floatlarni ishlatish uchun kodingizni quyidagicha o'zgartirishingiz mumkin:

```
x = float(input("What's x? "))
y = float(input("What's y? "))

print(x + y)
```

Ushbu o'zgarish foydalanuvchingizga 1.2 va 3.4 ni kiritish uchun jami 4.6 ni taqdim etish imkonini beradi.

Tasavvur qilaylik, siz umumiy sonni eng yaqin butun songa yaxlitlashni xohlaysiz. Python qollanmasiga etibor bersak buning uchun round funksiyasidan foydalanamiz `round(number[n, ndigits])` . Shu bilan bir qatorda, siz quyidagi tarzda kodlashingiz mumkin:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Create a rounded result
z = round(x + y)

# Print the result
print(z)
```

Agar biz uzun raqamlarning chiqishini formatlashni xohlasak nima bo'ladi? Misol uchun, 1000 ni ko'rishdan ko'ra, siz 1,000 ni ko'rishni xohlashingiz mumkin. Kodingizni quyidagicha o'zgartirishingiz mumkin:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Create a rounded result
```

```
z = round(x + y)

# Print the formatted result
print(f"{z:,}")
```

Garchi juda sirli bo'lsa-da, bu print (f"{z:,}") stsenariy yaratadi, unda chiqarilgan z vergulni o'z ichiga oladi, natijada 1,000 yoki 2,500 ko'rinishi mumkin.

O'zgaruvchan nuqta qiymatlarini qanday qilib yaxlitlashimiz mumkin? Birinchidan, kodingizni quyidagicha o'zgartiring:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Calculate the result
z = x / y

# Print the result
print(z)
```

2 ni x va 3 ni y sifatida kiritganda, z natijasi 0,666666666, biz kutganimizdek, cheksiz bo'lib ketadi.

Tasavvur qilaylik, biz buni yaxlitlashni xohlaymiz, biz kodimizni quyidagicha o'zgartirishimiz mumkin:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Calculate the result and round
z = round(x / y, 2)

# Print the result
print(z)
```

Biz kutganimizdek, bu natijani eng yaqin ikki kasr nuqtasiga yaxlitlaydi.

Natijani quyidagi tarzda formatlash uchun `fstring` dan ham foydalanishimiz mumkin:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Calculate the result
z = x / y

# Print the result
print(f"{z:.2f}")
```

Ushbu sirli `fstring` kodi oldingi yaxlitlash strategiyamiz bilan bir xil ko'rinadi.

Def

O'z funktsiyalarimizni yaratish yaxshi emasmi?

Terminal oynasiga code hello.py kodini kiritish orqali oxirgi hello.py kodimizni qaytaramiz. Sizning boshlang'ich kodingiz quyidagicha ko'rinishi kerak:

```
# Ask the user for their name, remove whitespace from the str and capitalize the first
letter of each word
name = input("What's your name? ").strip().title()

# Print the output
print(f"hello, {name}")
```

Biz uchun "hello" deb aytadigan o'z maxsus funksiyamizni yaratish uchun kodimizni yaxshilashimiz mumkin!

barcha kodimizni o'chirib tashlab, noldan boshlaylik:

```
name = input("What's your name? ")
hello()
print(name)
```

Ushbu kodni ishga tushirishga urinayotganda, kompilyatoringiz xatoga yo'l qo'yadi. Axir, hello uchun aniq funksiya yo'q.

Hello deb nomlangan o'z funksiyamizni quyidagicha yaratishimiz mumkin:

```
def hello():
    print("hello")


name = input("What's your name? ")
hello()
print(name)
```

E'tibor bering, def hello() ostidagi hamma code bir oz ongga surilgan. Python - da shubday dasturlanadi. Yuqoridagi funktsiyaning bir qismi nima ekanligini tushunish uchun ongga surishidan foydalanadi yani python tilida indentetaion deyiladi. Shuning uchun, hello funktsiyasidagi hamma narsa indent langan bolishi kerak. Agar biror narsa indentlangan bolmasa, u hello funktsiyasi ichida bo'lmagandek muomala qiladi. Terminal oynasida python hello.py dasturini ishga tushirsangiz, natija siz xohlagandek emasligini ko'rasiz.

```
# Create our own function
def hello(to):
    print("hello,", to)


# Output using our own function
name = input("What's your name? ")
hello(name)
```

Bu erda, birinchi qatorlarda siz o'zingizning hello funksiyangizni yaratasiz. Biroq, bu safar siz kompilyatorga ushbu funktsiya bitta parametrni qabul qilishini aytasiz: o'zgaruvchiga chaqirilgan. Shuning uchun, siz `hello(name)` deb ishlatilganda, kompyuter name ni hello funktsiyasiga o'tkazadi. Mana shunday qilib biz qiymatlarni funksiyalarga yetqazamiz. Juda foydali! Terminal oynasida python hello.py dasturini ishga tushirganingizda, natija ushbu ma'ruzada ilgari taqdim etilgan idealimizga ancha yaqin ekanligini ko'rasiz.

```
# Create our own function
def hello(to="world"):
    print("hello,", to)


# Output using our own function
name = input("What's your name? ")
hello(name)
```

```
# Output without passing the expected arguments
hello()
```

Kodingizni o'zingiz sinab ko'ring. E'tibor bering, birinchi hello siz kutganingizdek o'zini tutadi va qiymat berilmagan ikkinchi hello sukut bo'yicha “Hello world” chiqaradi.

Dasturimiz boshida o'z funksiyamiz bo'lishi shart emas. Biz uni pastga siljitishimiz mumkin, lekin biz kompilyatorga asosiy funksiyamiz va alohida hello funksiyamiz borligini aytishimiz kerak.

```
def main():

    # Output using our own function
    name = input("What's your name? ")
    hello(name)

    # Output without passing the expected arguments
    hello()


# Create our own function
def hello(to="world"):
    print("hello,", to)
```

Biroq, buning o'zi bir turdagi xatolikni keltirib chiqaradi. Agar biz python hello.py ni ishga tushirsak, hech narsa bo'lmaydi! Buning sababi shundaki, ushbu koddagi hech narsa aslida asosiy funktsiyani chaqirmaydi.

Quyidagi juda kichik modifikatsiya asosiy funktsiyani chaqiradi va dasturimizni ish tartibiga qaytaradi:

```
def main():

    # Output using our own function
    name = input("What's your name? ")
    hello(name)

    # Output without passing the expected arguments
    hello()
```

```
# Create our own function
def hello(to="world"):
    print("hello,", to)


main()
```

Siz ko'p stsenariylarni tasavvur qilishingiz mumkin, bu erda siz funktsiyani nafaqat harakatni bajarishini, balki qiymatni asosiy funktsiyaga qaytarishni ham xohlaysiz. Masalan, oddiygina x + y hisobini chop etishdan ko'ra, funktsiyadan ushbu hisob qiymatini dasturingizning boshqa qismiga qaytarishni xohlashingiz mumkin. Ushbu qiymatning "return value" biz qaytish qiymati deb ataymiz.

calculator.py barcha kodlarni o'chiring. Kodni quyidagicha qayta ishlang:

```
def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))


def square(n):
    return n * n


main()
```

Samarali ravishda x kvadratga o'tkaziladi. Keyin x * x hisobi asosiy funktsiyaga qaytariladi.

Ushbu bitta ma'ruza orqali siz o'z dasturlaringizda ko'p marta foydalanadigan qobiliyatlarni o'rgandingiz. Siz o'rgandingiz ...

Creating your first programs in Python;
Functions;
Bugs;
Variables;
Comments;
Pseudocode;
Strings;

Parameters;
Formatted Strings;
Integers;
Principles of readability;
Floats;
Creating your own functions; and
Return values.