String formatting

...printf(), sprintf() equivalents & related topics.

Simplest possible output from Rust program

```rust
fn main() {
    println!("Hello stdout!");
    eprintln!("Hello stderr!");
}
```

Formatted output (using `Display` trait)

```
fn agriculture() {
    let num_animals = 42_usize;
    let animal_name = "ducks";

    println!("We have {} {} in our farm.", num_animals, animal_name);

    let s: String = format!(
        "Nothing is better than {0} {2}, except for {1} {2},",
        num_animals, num_animals + 1, animal_name
    );

    // Minimal assert.
    assert!(num_animals >= 42);
    // assert with a custom panic message.
    assert!(
        num_animals >= 42,
        "Too few animals in our farm :( - only {} {}", num_animals, animal_name
    );
}
```

Formatted output (using `Debug` trait)

```
fn agriculture() {
    let animals: &[&str] = &["Azor", "Mućka"];

    // Does not compile: &[&str] does not implement Display.
    // println!("We have the following animals in our farm: {}", animals);

    // Concise printing for debug purposes:
    println!("We have the following animals in our farm: {:?}", animals);
    // Outputs:
    // We have the following animals in our farm: ["Azor", "Mućka"]

    // Pretty-printing for debug purposes:
    println!("We have the following animals in our farm: {:#?}", animals);
    // Outputs:
    // We have the following animals in our farm: [
    //     "Azor",
    //     "Mućka"
    // ]
}
```

Memory backing considerations

```rust
fn agriculture() {
    let animals: &[&str] = &["Azor", "Mućka"];

    let animals: [&str; 2] = ["Azor", "Mućka"];
    let animals: &[&str] = &animals;

    let animals: Vec<&str> = vec!["Azor", "Mućka"];

    static ANIMALS: [&str; 2] = ["Azor", "Mućka"];
    static ANIMALS_SLICE: &[&str] = &ANIMALS;

    let animals: Vec<&str> = vec!["Azor", "Mućka"];
    let animals_slice: &[&str] = &animals;

    let animals: Vec<String> = vec!["Azor".into(), "Mućka".into()];
}
```

Memory backing considerations - with hints

```rust
fn agriculture() {
    let animals: &[&str] = &["Azor", "Mućka"]; // stack-allocated stack-backed slice.

    // stack-allocated array (of statically-allocated strings).
    let animals: [&str; 2] = ["Azor", "Mućka"];
    let animals: &[&str] = &animals; // stack-allocated stack-backed slice.

    let animals: Vec<&str> = vec!["Azor", "Mućka"]; // stack-allocated heap-backed slice.

    static ANIMALS: [&str; 2] = ["Azor", "Mućka"]; // statically-allocated array.
    static ANIMALS_SLICE: &[&str] = &ANIMALS; // statically-allocated statically-backed slice.

    let animals: Vec<&str> = vec!["Azor", "Mućka"]; // stack-allocated heap-backed Vec.
    let animals_slice: &[&str] = &animals; // stack-allocated heap-backed slice.

    // stack-allocated heap-backed Vec of heap-allocated strings.
    let animals: Vec<String> = vec!["Azor".into(), "Mućka".into()];
}
```